LifeGraph: a Knowledge Graph for Lifelogs

Rossetto, Luca ; Baumgartner, Matthias ; Ashena, Narges ; Ruosch, Florian ; Pernisch, Romana ; Bernstein, Abraham

DOI: https://doi.org/10.1145/3379172.3391717

LifeGraph: a Knowledge Graph for Lifelogs

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

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

Narges Ashena
University of Zurich
ashena@ifi.uzh.ch

Florian Ruosch
University of Zurich
ruosch@ifi.uzh.ch

Romana Pernischová
University of Zurich
pernischova@ifi.uzh.ch

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

ABSTRACT

The data produced by efforts such as life logging is commonly multi modal and can have manifold interrelations with itself as well as external information. Representing this data in such a way that these rich relations as well as all the different sources can be leveraged is a non-trivial undertaking. In this paper, we present the first iteration of LifeGraph, a Knowledge Graph for lifelogging data. LifeGraph aims at not only capturing all aspects of the data contained in a lifelog but also linking them to external, static knowledge bases in order to put the log as a whole as well as its individual entries into a broader context. In the Lifelog Search Challenge 2020, we show a first proof-of-concept implementation of LifeGraph as well as a retrieval system prototype which utilizes it to search the log for specific events.

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, Narges Ashena, Florian Ruosch, Romana Pernischová, and Abraham Bernstein. 2020. LifeGraph: a Knowledge Graph for Lifelogs. In *Proceedings of the Third Annual Workshop on the Lifelog Search Challenge (LSC '20), June 9, 2020, Dublin, Ireland*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379172.3391717

1 INTRODUCTION

With the increase in both capability and availability of mobile computation and sensing technologies, the means for capturing a growing fraction of the human experience become increasingly available. While a personal diary historically may have only consisted of the manual and subjective textual recording of ones daily activities or thoughts, maybe occasionally accompanied by a photograph or a memento of a particular occasion, modern sensors add many layers of more objective data. Such sensor data can consist of visual information about a person's surroundings, represented as a series of images or videos, augmented by various spacial and contextual information, as well as the current state of the person itself in the form of bio-metrics. The data generated this way is inherently multi-modal and might have a complex structure of interrelation between its various aspects. Another inherent property of this kind of data is that it continues to grow over time as more information about the recent past gets added. This leads to a need for some retrieval mechanisms in order for the data to remain useful for several application, such as memory augmentation. The Lifelog Search Challenge [8] aims at addressing this problem by providing an environment for the evaluation of retrieval systems capable of handling such data which might be produced by the process of life-logging.

In order to utilize the different modalities of the data effectively for retrieval, it is important to capture the relationships between them. Along a similar vein, it could be beneficial to associate the lifelog data, which is of a temporal nature, with static facts about the world in order to enrich the semantic context of the whole. The most appropriate data structure to meet these requirements is a graph.

In this paper, we present *LifeGraph*, a Knowledge Graph for Lifelog Data. LifeGraph is designed in such a way that it captures the internal relations of the various data modalities contained within a lifelog while simultaneously linking to large static knowledge bases in order to enrich the semantic context of the lifelog.

The remainder of this paper is structured as follows: Section 2 gives a brief overview of related work on both Knowledge Graphs and lifelog retrieval. Section 3 discusses the construction of LifeGraph while Section 4 outlines how it can be used for retrieval. Section 5 then provides some details on the system prototype which is used in the Lifelog Search Challenge before Section 6 concludes.

2 RELATED WORK

In the comparatively short time the Lifelog Search Challenge offers a common basis for the evaluation of retrieval approaches geared towards lifelog data, participating teams have already begun to explore several aspects of the retrieval process. Multiple teams started out adapting their retrieval approaches originally developed for the Video Browser Showdown [25], an interactive video retrieval campaign, to the similarly structured lifelog retrieval task, which lead to a strong focus on the visual aspects of the lifelog data [16, 18,

LSC '20, June 9, 2020, Dublin, Ireland

ACM ISBN 978-1-4503-7136-0/20/06…$15.00
https://doi.org/10.1145/3379172.3391717

22]. Other systems have experimented with novel user interaction approaches. The Exquisitor system [13] uses an interactive learning approach to formulate and refine queries based on a visual relevance feedback scheme where a user can mark retrieved results as either relevant or not, thereby iteratively approaching the desired result. The system presented in [5] meanwhile successfully uses a virtual reality user interface for the efficient interaction with retrieval results as well as the presentation of them. So far however, little work has been done on the enrichment of the lifelog data itself by the use of suitable data representations. We aim to start the exploration of this direction by the introduction of Knowledge Graphs and Semantic Web technologies into the area of lifelog retrieval.

The core idea of the Semantic Web is to have a Web that is machine processible [1], which was the driving motivation behind the development of the Resource Description Framework (RDF) [4]. In RDF, facts are stated as subject-predicate-object triples, wherein the subject and object are resources and the predicate expresses a relationship between them. Since the Web is distributed, resources and predicates are identified globally by means of prefixes. Furthermore, the Web Ontology Language (OWL) permits to define constraints on triple elements. In combination, RDF and OWL provide the means to build ontologies and tools for compliance checking and logical reasoning [11]. In recent years, the idea of Knowledge Graphs (KG) was popularized. Although there is no prevalent definition of the term, KGs are mostly an interpretation of RDF triples as a networks of facts, whereas nodes (resources) represent real-world entities. Contrasting ontologies, they rarely employ constraints, meaning that reasoning tools cannot be applied [12]. Instead, they are often a product of a (semi-)automated information extraction process. For example, DBpedia was constructed by processing Wikipedia info boxes [15]. The capability to access information in a KG is provided by the SPARQL query language which allows to define graph patterns that are matched against [9].

3 KNOWLEDGE-GRAPH CONSTRUCTION

This section outlines the construction procedure of the LifeGraph as well as the data sources and annotation methods used.

In this context, a knowledge graph (KG) is a set of triples (s, p, o), where s and o are two resources connected by a predicate/relation p. As an example, (`:entry :image ^^xsd:string`) is the triple describing the schema, where an entry has the relation image and the object is a string of the image identifier. A KG consist of two sets – TBox and ABox [2]. The TBox (Terminology Box) consists of the schema: range and domain restrictions as well as definitions of classes and relations. The ABox (Assertion Box) holds the individuals or population as defined in the TBox. Therefore, the TBox corresponds to a database schema, whereas the ABox can be compared to the tables within the same database.

3.1 Data Sources

In order to maximize intelligence, we fuse information from multiple sources that will be briefly described. Naturally, our starting point is the core image data set provided by the organizers (and obtainable from the LSC website) [7]. In the following, we will explain how we linked it to two additional data sources which will be described here first: the "Classification of Everyday Living" (COEL) [3] and Wikidata [26].

COEL is part of our graph schema and represents events that may occur in our everyday lives in a four-level hierarchical structure. More than 5'000 elements are organized in just over 1'000 subclasses which are in turn classified in almost 200 categories. On the top-most level, everything is divided into 32 clusters. COEL follows some design principles aiming to make sure that the model is complete (exhaustively so) as well as discretely hierarchical and also distinctive at the same time.

Wikidata is a sister project of Wikipedia and contains its information in a "Web of Data" following several design principles. While continuously evolving due to its open editing approach, Wikidata serves as an easily accessible source for secondary information (i.e. facts published elsewhere). In providing an API (for SPARQL queries) and by not only connecting its elements among themselves but also to other knowledge bases (e.g. WordNet, Encyclopædia Britannica Online) it enables enriching keywords with additional semantic information.

3.2 Graph Schema

Meta-data and detectables. For defining a graph structure, we considered different aspects. First of all, the lifelog images and their meta-data had to be structured in a meaningful way to enable an intuitive search over the entries. An additional challenge posed the fact, that entries in the meta-data are in the interval of a minute while pictures were taken in regular but different intervals. Therefore, multiple pictures can be associated with the same meta-data entry. To model this in a meaningful way, we consider every image to be an instance of class `:entry`. Attributes, such as time, day, month, time of day, activity, etc., which are entered in the meta-data are associated with an `:entry`. The relation `:image` then connects the image id with the entry. In this simple way, we associate the meta-data with the images. Because of the difference in timing of meta-data entries and images, there will be multiple images associated with one meta-data entry. We duplicate the meta-data entry so that each image has the corresponding information available. This choice of design arose from the goal of the challenge, which is the retrieval of images and not meta-data. Therefore, the images are the center point of the schema.

Besides the meta-data, we need to associate objects detected in the images with the corresponding images in the graph. This will enable us to search for entries in the lifelog showing specific `:detectables`. A detectable is anything we can *detect* in the images, such as physical objects or specific scenes or settings. Using the predicate `:detected`, we associate the detectable with the entry. Therefore, the schema triple looks the following way: `(:entry :detected :detectable)`. Each detectable will be modeled as an instance of the class `:detectable`.

COEL. As mentioned above, the second part of the schema is COEL [3], which is a hierarchical structure of activities a person potentially does. Since this is a terminology and not an ontology, we first convert COEL into the Resource Description Format (RDF) graph. During this conversion, we model the terms in COEL as classes and keep the hierarchy intact by defining sub- and superclasses with the predicate `owl:SubClassOf`. The lowest level of

concepts, called elements, are added as individuals of the subclass they are part of. In triple format, this looks like the following: `:e1 rdf:type :c1`, with `:e1` being a specific element and `:c1` denoting a subclass which the element is part of.

At first, these two structures are not connected with each other. The connection between the meta-data and COEL will inherently happen though the images.

3.3 Linking

We extend COEL by linking it to Wikidata entities in two ways. First, we create a mapping between COEL concepts and Wikidata entities. Such a mapping allows us to use both knowledge bases in conjunction and to expand the number of classes. Second, we populate the resulting schema with instances from Wikidata. This increases the number of detectable objects as it links activities with objects typically involved in the activity.

Since the two data sources have not yet been linked, we employ a semi-automatic mapping procedure to derive a mapping between them. The mapping procedure is based on text literals which are either COEL concept names or labels attached to Wikidata entities (via rdfs:label and skos:altLabel predicates). For each COEL concept, we retrieve all entities from Wikidata that have a similar label, ignoring special entities (e.g. Wikidata disambiguation pages). If a single entity with a perfectly matching label is found, that entity is linked to the COEL concept automatically. We review the link manually in cases where multiple entities match the same COEL concept or where the labels match approximately. Similarly, we review COEL concepts for which no entitiy was retrieved. Such cases mostly occur at subclass level where COEL describes activities ("drying your hair") which have very limited coverage in Wikidata. We leave cases without corresponcence unlinked.

The set of linked Wikidata entities is further expanded with classes along Wikidata's class hierarchy (wdt:P279) and Wikipedia's categories (wdt:P373, wdt:P910). Furthermore, we populate the resulting schema with instances. For this, we collect all Wikidata entities that are an instance (wdt:P31), a part (wdt:P361), or a use (wdt:P2283) of a class in our schema.

It is worth noting that the resulting structure is a graph rather than a hierarchy, since concepts from different locations in the COEL tree map to the same Wikidata entity if they have the same label. For example, Hair is mapped to the same entity whether it is part of COEL:Personalcare or COEL:Childcare.

3.4 Image annotation

In addition to some freely available pretrained object localizers [20, 21] we construct an array of binary classifiers to detect the presence of a large number of semantic concepts in the images of the provided dataset. These classifiers use the output of the last hidden layer of a ResNet50 [10] instance which has been pretrained on ImageNet. For their training, we use a combination of several image datasets [14, 17, 27] in order to obtain a diverse set of labelled semantic concepts which can be meaningfully linked to other nodes in the graph structure.

3.5 Temporal Interpolation

Since the temporal aspects of the data might contain gaps due to incomplete manual annotation, missing detections, or similar reasons, we perform a temporal interpolation step after automatically annotating the images. This process looks for two sequences of entries which are associated with the same concept but interrupted by a sufficiently small gap. In such cases, we assume that, in reality, there is actually only one continuous sequence of entries to which the concept should be associated rather than two following temporally closely to each other. Consequently, the two adjacent sequences are merged by associating the relevant concept to the entries comprising the gap. For sake of completeness, we not only apply this process to the annotations generated by the process described in Section 3.4 but also to the meta-data which was provided with the competition dataset.

4 RETRIEVAL

This section provides some details on how retrieval is performed in the LifeGraph.

4.1 Query formulation and expansion

To specify a query, a user can select an arbitrary number of tags they would deem to be relevant. Each of these tags is associated to a node in the graph. However, since not all tags are directly associates with individual *log entries*, the graph is traversed starting out from each of the selected tags until either a maximum number of directly associated nodes or a maximum expansion depth is reached. Nodes in the graph which can be directly associated with log entries are predominantly instances of *detectable* objects or concepts extracted from the visual part of the lifelog dataset.

4.2 SPARQL Queries and Templates

SPARQL [19] is a graph based query language for retrieval of parts of an RDF graph and based on pattern matching. A pattern is a construct of triples where any part of the triple can be replaced by a variable. They can be of varying length while also defining optional parts. The triple store then searches for triples or sets thereof that correspond to the specified pattern. In the current case, we are interested in retrieving the images, and the pattern will need to be defined in relation to the images. Therefore, the SELECT statement holds the impact identifier. The WHERE section contains the triple pattern that will be matched by the triple store. An example can be seen in Listing 1. If we want to search for images where either a cup or coffee was detected, we can use the shown template. We would then substitute `?thing1` with `:cup` and `?thing2` with `:coffee`. The keyword UNION means, that the query will return results that either match the first, the second, or both statements.

The query in Listing 1 is one of many templates we are going to construct which will allow the user to choose from different types of tags (detectables, activities, meta-data). Depending on the selection, different templates are filled and executed to return images. Our goal is to keep the templates as simple as possible and rank the returned images based on occurrence within the results. Images that are returned by multiple queries will be ranked higher than those returned by only one.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lsc: <http://www.ddis.ch/lsc-schema#>
SELECT ?img
WHERE {
    ?entry      lsc:image       ?img.
    {
        ?entry      lsc:detected    ?thing1.
        ?thing1     rdf:type        lsc:detectable.
    }
    UNION
    {
        ?entry      lsc:detected    ?thing2.
        ?thing2     rdf:type        lsc:detectable.
    }
}
```

Listing 1: Example of a SPARQL Query returning image identifiers for ?thing1 = :cup and ?thing2 = :coffee. Returned images contain a cup or coffee tag available for the image.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lsc: <http://www.ddis.ch/lsc-schema#>
SELECT ?img
WHERE {
    ?entry      lsc:image       ?img.
    ?entry      lsc:detected    ?thing1.
    ?thing1     rdf:type        lsc:detectable.
    OPTIONAL
    {
        ?entry      lsc:detected    ?thing2.
        ?thing2     rdf:type        lsc:detectable.
    }
}
```

Listing 2: Example of a SPARQL Query returning image identifiers for ?thing1 = :cup and if available also for ?thing2 = :coffee

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lsc: <http://www.ddis.ch/lsc-schema#>
SELECT ?img
WHERE {
    ?entry      lsc:image       ?img.
    ?entry      lsc:detected    ?thing1.
    ?thing1     rdf:type        lsc:detectable.
    ?entry      lsc:detected    ?thing2.
    ?thing2     rdf:type        lsc:detectable.
}
```

Listing 3: Example of a SPARQL Query returning image identifiers for ?thing1 = :cup and for ?thing2 = :coffee. Both tags need to be present for the image to be returned.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX lsc: <http://www.ddis.ch/lsc-schema#>
SELECT ?img
WHERE {
    ?entry      lsc:image       ?img.
    ?entry      lsc:detected    ?thing.
    ?thing      rdf:type        lsc:detectable.
    FILTER ( ?thing IN (:cup, :coffee, :tee, :hotdrink))
}
```

Listing 4: Example of a SPARQL Query returning image identifiers for a list of tags.

Since it is better to return more images rather than excluding some, all images which have matched at least one of the selected tags will be shown to the user. The user also has the possibility to search for two tags, of which both need to be associated with an image, for example plant (detectable) and afternoon (time of day). However, when the user wants matches to more than one tag, the types of the tags need to be different from each other. Tags within one category will always be matched with the previously described UNION keyword.

There are more possibilities of queries, which could be applied in this context. For example, letting the user choose how tags are combined, we also envision the usage of OPTIONAL (Query 2) or simply only graph patterns without additional keywords (Query 3). With the OPTIONAL keyword, the first graph pattern has to match for the query to return the results, but the second part would be considered optional. Therefore, results would be returned, if they match the first part and the additional matches would also be returned, if available. In case no match is found for the OPTIONAL part, it will remain empty in the response. Lastly, when using no keywords all patterns need to be matched.

The last query presented in 4 is the simplest but still very powerful. It allows to query for images that contain one or more detected items from a list. An image and the detected item will be returned for every list of tags, if the image is tagged with any of them. In the case of cup and coffee, there would be two results returned for an image that has both the tag cup and coffee. On the other hand, the image would be returned once, if it has one tag, either cup or coffee. By returning not just the image identifier (?img) but also the tag (?thing), we are able to distinguish the returned images based on how many of the tags of the list it contained. This would enable a ranking of the images for display in the front-end.

We can summarize that from Query 1, to Query 2, and Query 3, the first is the least strict and the third is the most restraining. The last example, Query 4, is the most expressive and allows for a wider range of applications and also number of tags.

4.3 Filtering

Once an initial set of images has been found, ranked, and returned to the user, filtering comes into play. The filtering does not touch the underlying architecture anymore but rather happens in the front end directly, using previously introduced functionality [22, 24]. Filtering over the meta-data but also over the previously queried tags is possible.

5 IMPLEMENTATION

Our retrieval system prototype consists of several components. A triple store is responsible for storing the LifeGraph and executing SPARQL queries on it. The query engine sits on top of the triple store and performs the translation between the queries specified by the user interface and the types of queries that can be evaluated by the triple store. The browser-based user interface used for this prototype is a modified version of *vitrivr-ng* [6] which we re-purpose from the *vitrivr* [23] content-based multimedia retrieval stack, since it has been shown to be useful for effective retrieval of lifelog data [22]. The user interface as well as all the visual content is served by a local web server. The entire system stack is illustrated in Figure 1.

6 CONCLUSION

In this paper, we introduced LifeGraph, a first attempt to represent complex interrelated data captured in the context of life logging

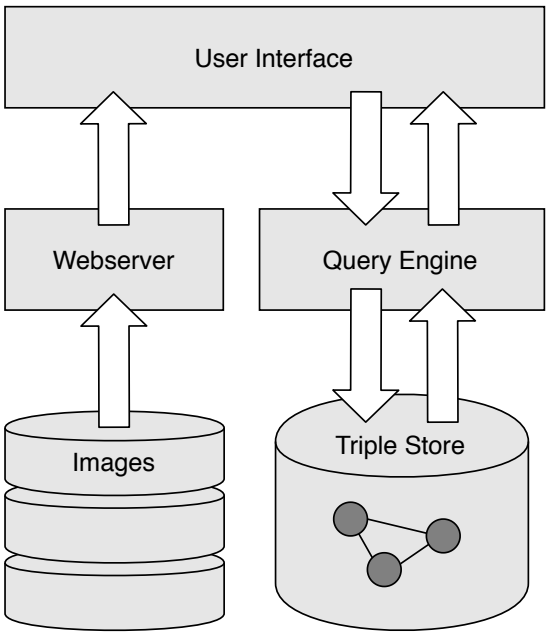

Figure 1: System diagram

using a knowledge graph, as well as some methods how it can be used for retrieval of lifelog data. We see the current iteration of LifeGraph as more of a proof-of-concept rather than a complete solution, usable by life loggers in general. This work could however lay the foundation for the construction of a specialized ontology which contains everything necessary to describe not only individual lifelog entries and their interrelations but also their context. Assuming the support of the growing life logging community such data could enrich future lifelog data-based applications.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for helpful feedback and suggestions. The authors would also like to thank the Swiss National Science Foundation (SNSF) for their generous partial support under grant numbers 184994 and 167177 (http://www.nfp75.ch). We would also like to thank the Swiss Re Institute financial support.

REFERENCES

[1] Grigoris Antoniou and Frank Van Harmelen. 2004. *A semantic web primer*. MIT press.
[2] Franz Baader, Carsten Lutz, and Sebastian Brandt. 2008. Pushing the EL Envelope Further. In *OWLED (Spring) (CEUR Workshop Proceedings)*, Vol. 496. CEUR-WS.org.
[3] 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.
[4] Richard Cyganiak, David Wood, and Markus Lanthaler. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C. https://www.w3.org/TR/rdf11-concepts/
[5] Aaron Duane, Cathal Gurrin, and Wolfgang Huerst. 2018. Virtual reality lifelog explorer: lifelog search challenge at ACM ICMR 2018. In *Proceedings of the 2018 ACM Workshop on The Lifelog Search Challenge*. 20–23.
[6] Ralph Gasser, Luca Rossetto, and Heiko Schuldt. 2019. Towards an all-purpose content-based multimedia information retrieval system. *arXiv preprint arXiv:1902.03878* (2019).
[7] Cathal Gurrin, Tu-Khiem Le, Van-Tu Ninh, Duc-Tien Dang-Nguyen, Björn Þór Jónsson, Jakub Lokoč, Wolfgang Hurst, Minh-Triet Tran, and Klaus Schoeffmann. 2020. An Introduction to the Third Annual Lifelog Search Challenge, LSC'20. In *ICMR '20, The 2020 International Conference on Multimedia Retrieval*. ACM, Dublin, Ireland.
[8] Cathal Gurrin, Klaus Schoeffmann, Hideo Joho, Andreas Leibetseder, Liting Zhou, Aaron Duane, Dang Nguyen, Duc Tien, Michael Riegler, Luca Piras, et al. 2019. Comparing approaches to interactive lifelog search at the lifelog search challenge (LSC2018). *ITE Transactions on Media Technology and Applications* 7, 2 (2019), 46–59.
[9] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C. http://www.w3.org/TR/sparql11-query/
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[11] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. 2012. *OWL 2 Web Ontology Language Primer (Second Edition)*. W3C Recommendation. W3C. http://www.w3.org/TR/owl-primer
[12] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2020. Knowledge Graphs. arXiv:cs.AI/2003.02320
[13] Omar Shahbaz Khan, Björn Þór Jónsson, Jan Zahálka, Stevan Rudinac, and Marcel Worring. 2019. Exquisitor at the lifelog search challenge 2019. In *Proceedings of the ACM Workshop on Lifelog Search Challenge*. 7–11.
[14] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. 2017. OpenImages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages* (2017).
[15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Soren Auer, and Christian Bizer. 2012. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. (2012).
[16] Andreas Leibetseder, Bernd Münzer, Manfred Jürgen Primus, Sabrina Kletz, Klaus Schoeffmann, Fabian Berns, and Christian Beecks. 2019. lifeXplore at the lifelog search challenge 2019. In *Proceedings of the ACM Workshop on Lifelog Search Challenge*. 13–17.
[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
[18] Jakub Lokoč, Tomáš Souček, Premysl Čech, and Gregor Kovalčík. 2019. Enhanced VIRET tool for lifelog data. In *Proceedings of the ACM Workshop on Lifelog Search Challenge*. 25–26.
[19] 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/.
[20] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.
[21] Joseph Redmon and Ali Farhadi. 2018. YOLOv3 on the Open Images dataset. https://pjreddie.com/darknet/yolo/.
[22] Luca Rossetto, Ralph Gasser, Silvan Heller, Mahnaz Amiri Parian, and Heiko Schuldt. 2019. Retrieval of structured and unstructured data with vitrivr. In *Proceedings of the ACM Workshop on Lifelog Search Challenge*. 27–31.
[23] 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 24th ACM international conference on Multimedia*. 1183–1186.
[24] Loris Sauter, Mahnaz Amiri Parian, Ralph Gasser, Silvan Heller, Luca Rossetto, and Heiko Schuldt. 2020. Combining Boolean and Multimedia Retrieval in vitrivr for Large-Scale Video Search. In *International Conference on Multimedia Modeling*. Springer, 760–765.
[25] Klaus Schoeffmann. 2019. Video Browser Showdown 2012-2019: A Review. In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE, 1–4.
[26] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.
[27] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2 (2014), 67–78.