KeySearchWiki: An Automatically Generated Dataset for Keyword Search over Wikidata

Leila Feddoul[1,2], Frank Löffler[3,1] and Sirko Schindler[2]

[1] *Heinz Nixdorf Chair for Distributed Information Systems, Friedrich Schiller University Jena, Jena, Germany*
[2] *Institute of Data Science, German Aerospace Center DLR, Jena, Germany*
[3] *Competence Center for Digital Research, Michael Stifel Center, Jena, Germany*

Abstract
Keyword search is an intuitive method to access knowledge graphs without requiring technical expertise or knowledge of the underlying data schema. In this context, various methods for keyword search over knowledge graphs have been developed. However, only few evaluation datasets have been created, mostly based on a time-consuming manual generation. We present KeySearchWiki, an automatically generated dataset for keyword search over Wikidata, containing over 16 thousand queries and their relevant results. It is based on Wikidata and Wikipedia set categories which are refined and combined to derive more complex queries. We explain the dataset generation workflow, highlight some dataset characteristics, present experiments using baseline retrieval methods, and evaluate the accuracy of relevant results.

Keywords
Keyword Search, Knowledge Graph, Wikidata, Wikipedia, Dataset

1. Introduction

Knowledge graphs (KGs) have become an undisputed source of semantic knowledge for various tasks, e.g., Question Answering (QA) or Entity Linking [1, 2]. Hence, techniques that simplify access for end-users are in great demand. Keyword Search over Knowledge Graphs (KSKG) is a familiar method enabling information retrieval. KSKG systems generally attempt to answer a user query by retrieving graph connections between query keywords. In general, the output of interest of KSKG systems is a set of uniquely identified relevant entities. Recently, KSKG research resulted in a wide range of methods developed [3, 4, 5, 6, 7, 8, 9, 10].

Benchmarks for effectiveness evaluation play a key role in enhancing systems and enabling inter-system comparison. They provide a target KG, user queries, and corresponding results together with their relevance judgments (RJs), e.g., using binary[1] or 3-point scales. Queries are often either manually crafted [11] or manually selected from search engine query logs [12], which results in small datasets. Relevant results are generally provided by pooling a subset

Wikidata'23: Wikidata workshop at ISWC 2023
leila.feddoul@uni-jena.de (L. Feddoul); frank.loeffler@uni-jena.de (F. Löffler); sirko.schindler@dlr.de (S. Schindler)
0000-0001-8896-8208 (L. Feddoul); 0000-0001-6643-6323 (F. Löffler); 0000-0002-0964-4457 (S. Schindler)

[1] Relevant (1), non-relevant (0), or provide only relevant results and consider anything else as non-relevant.

of system's top results and judged via crowd-sourcing [13, 11, 14]. This approach is time-consuming and depends on systems' results. To the best of our knowledge, there is only one dataset specifically for KSKG [15]. Its focus was not on creating queries, but on mapping relevant results from previous evaluation campaigns to DBpedia [16] entities.

In this paper, we present KeySearchWiki, an automatically generated dataset for keyword search over Wikidata [17]. We focus on Type Search (TS) [18] with queries retrieving entities of a specific type (target), e.g., *Paul Auster novels* with *novels* as a target. This relates to common real-world scenarios: (1) users explicitly mentioning the target in traditional search engines, (2) users selecting a target category, e.g., books in an online shop, or (3) search systems providing access to only a single type of results, e.g., portals offering access to datasets. To the best of our knowledge, this is the first automatically generated, large-scale, and diverse dataset that also includes complex queries. The general idea is to leverage *Wikipedia set categories*[2] that are mapped to Wikidata (e.g., *Category:American television directors (Q8032156)*) as a source of queries and their members as relevant entities. KeySearchWiki is more closely related to human-curated datasets than purely synthetic ones. The queries represent an actual information need as witnessed by the manually maintained, corresponding Wikipedia categories. Furthermore, Our approach is both multilingual (all Wikipedia languages are considered) and hierarchical (exploiting the Wikipedia category hierarchy). We summarize the key contributions as follows:

- We present a workflow for the automatic generation of the KeySearchWiki dataset. The source code for the dataset generation is publicly available.
- We introduce KeySearchWiki, a diverse dataset consisting of 16, 605 queries of different complexity levels together with their relevant entities.
- We provide for each query an annotated version that tags each query term with its corresponding Wikidata identifier. Mappings between natural language queries and corresponding Wikidata entities can, e.g., be used to evaluate entity linking systems.

2. Related Work

In addition to challenges, several datasets have been created to support the evaluation of KSKG approaches and related topics such as QA. Table 1 summarizes major existing datasets/challenges. They vary with respect to the task, the target data and its format, size, query creation method, source of relevant entities, RJs types, and RJs source. We distinguish between four types of tasks:

- Entity Search (ES) [18]: finding a specific entity, e.g., *University of Phoenix* in SemSearch Challenge 2010's Entity Search Track (SemSearch2010 ES).
- Type Search (TS) [18]: finding a (ranked) list of entities having a specific type, e.g., *Paul Auster novels* from the Entity Ranking Task of The INitiative for the Evaluation of XML retrieval (INEX2009 ER).
- Ad-Hoc Search (AHS): finding a (ranked) list of entities that are described with a set of random keywords (could include ES and TS style queries but also other random ones),

[2]https://en.wikipedia.org/wiki/Wikipedia:Categorization#Set_categories

Table 1
Overview of existing datasets/challenges. (*QC:* Query Creation, *RE source:* Relevant Entities source, *MH:* multi-hop, *Anot.*: Annotation of query terms with KG entities, *WP:* Wikipedia, *WD:* Wikidata, *DBp:* DBpedia, *WPLOD:* Wikipedia-LOD v1.1, *sem. XML:* semantically annotated XML, *rel.*: present only relevant entities, *collec.:* collected)

Dataset	Year	Task	Data	Format	QC	#Queries	RE source	RJs type	RJs source	MH	Anot.
INEX2009 ER [19]	2009	TS	WP/YAGO	sem. XML	collec.	55	runs	binary	participant	-	N
SemSearch2010 ES [12]	2010	ES	BTC2009 [20]	RDF	log	92	runs	3-point	crowd	-	N
SemSearch2011 LS [13]	2011	TS	BTC2009	RDF	log	50	runs	3-point	crowd	-	N
INEX2012 LD [11]	2012	AHS	WPLOD	sem. XML	manual	140	runs	-	crowd	-	N
INEX2013 LD [14]	2013	AHS	WP/DBp	XML/RDF	manual	144	runs	-	crowd	-	N
DBpedia-Entity v2 [15]	2017	all	DBp	RDF	collec.	467	runs/SPARQL	3-point	crowd	-	N
QALD-9 [21]	2018	QA	DBp	RDF	log/collec.	558	SPARQL (manual)	rel.	-	Y	N
LC-QuAD 2.0 [22]	2019	QA	DBp/WD	RDF	semi-auto	$30,000$	SPARQL (auto)	rel.	-	Y	N
KeySearchWiki	2022	TS	WD	RDF	auto	$16,605$	WP (auto)	rel.	-	Y	Y

e.g., *invented telescope* from the Ad-Hoc Search Task of the INEX2012 Linked Data track (INEX2012 LD).

- QA: finding entities that answer a natural language question, e.g., *Which people were born in Heraklion?* from Question Answering over Linked Data challenge 9 (QALD-9).

ES, TS, and AHS are directly related to KSKG. We include datasets for QA over KGs, as they share characteristics with KSKG datasets and have been adapted to evaluate KSKG systems (e.g., in [3]). KSKG systems require a target KG represented as triples. Most of the described datasets (cf. Table 1) provide underlying data in RDF format. INEX campaigns usually focus on XML retrieval, but organized a Linked Data (LD) Track to close the gap with the Semantic Web. The target data provided by INEX2012 LD was in a semantically annotated XML format: Wikipedia-LOD v1.1[3] using DBpedia and YAGO [23] annotations. In INEX2013 LD different dataset collections allowed for various retrieval techniques: XML + RDF (English Wikipedia + DBpedia and YAGO), semantically annotated XML (Wikipedia-LOD v2.0[4]), and text (extracted from Wikipedia-LOD v2.0). However, INEX LD tracks mainly target textual data, while KSKG systems work on data represented by KGs.

Queries are often either manually crafted by humans [11] or collected from previous campaigns where source queries were also manually created [15]. The manual approach is time-consuming and requires effort not only for query creation but also for finding interested volunteers. This impacts the size of the dataset, since it results in a small number of queries, usually fifty[5] to one hundred queries per dataset. Furthermore, if the dataset is created in the context of a project that also aims at developing a KSKG system, manual query creation increases the risk of designing biased queries that favor ones own approach, especially if this is done by researchers directly related to the project. Other works select queries manually from search engine query logs [12]. They argue that log queries are more realistic and representative to user needs. However, users often try to overcome the limitations of a search engine by adapting to its capabilities and by avoiding complex queries that involve relations between different entities [24]. On the other hand, query logs are often not in-line with the

[3]https://inex.mmci.uni-saarland.de/tracks/lod/2012/
[4]https://inex.mmci.uni-saarland.de/tracks/lod/2013/
[5]Established minimum for evaluating retrieval systems [24].

specified underlying data (e.g., KGs). This shortcoming requires additional efforts for selecting queries to have at least some answers within the considered data. Datasets should also contain queries with unambiguous intentions. This is important for judging whether a potential entity is relevant to the query or not. Thus, another step is the selection of queries whose intentions could be derived. LC-QuAD 2.0 [22] is, to the best of our knowledge, the only dataset that applies a semi-automatic approach for query creation and thus has the largest size ($30,000$ queries). SPARQL queries are automatically generated, transformed to template questions, and finally verbalized into natural language questions. However, QA datasets are not initially geared towards KSKG tasks. Hence, their usage requires pre-processing and selection of suitable queries. Another approach to evaluate KSKG systems is the use of *randomly* generated queries (arbitrary combinations of keywords appearing in the data source). This is generally not a good practice, since resulting queries would not reflect real information needs [25].

All challenges (besides QA) use runs[6] submitted by participants as source of relevant entities. They pool a subset of top results and assess them either via crowd-sourcing (e.g., MTurk[7]) or by the participants themselves [19]. This depends on the participating systems, though, and provides no independent list of relevant entities. QA datasets use either automatically or manually created SPARQL queries to generate the list of relevant entities. Here, results do not need to be judged, and only relevant entities are presented. However, we believe that relevant entities could originate from different SPARQL queries depending on the underlying KG. Using a single SPARQL query may omit some potentially relevant entities.

KSKG datasets should also contain complex queries. We distinguish between two degrees of complexity. *Multi-keyword*: queries that contain more than one keyword, and *multi-hop (for TS)*: there is no direct relation between target and keywords[8]. Most of the listed datasets contain multi-keyword queries. However, none of them explicitly claims to provide multi-hop queries. For QA datasets this could be verified since SPARQL queries are provided.

All datasets provide queries as mere strings. Systems thus have to deal with keyword/target to knowledge graph entity mapping. Providing semantically annotated queries is also useful and allows for using the queries also to evaluate entity linking systems. Another significant quality for such datasets is diversity, which means avoiding similar queries (e.g., *cities in Germany* and *cities in France*). It is difficult to programmatically verify the diversity of a dataset, so researches have to rely on the claims made by its creators (e.g., LC-QuAD [26] claims to avoid generating similar queries). Methods for constructing synthetic benchmarks are proposed in [27, 28]. They follow exactly the steps that a KSKG/QA system would perform to solve the task. In our view, this self-reference (evaluating one system with the output of another) defeats the idea of an objective evaluation and is not based on human judgment.

We conclude that there is a lack of established evaluation datasets dedicated to KSKG. We overcome the previously described shortcomings by proposing the first (1) automatically generated, (2) complex, (3) large-scale, (4) diverse dataset to evaluate KSKG systems on the TS Task over Wikidata, and providing semantically annotated queries. We propose an innovative approach by using Wikidata/Wikipedia set categories, existing human-edited sources of relevant

[6]Output (ranked) list of relevant entities produced by the participating systems.

[7]https://www.mturk.com/

[8]The corresponding SPARQL query consists of two connected triples (e.g., *?target ns:relation1 ?iri2 . ?iri2 ns:relation2 ?keyword).*

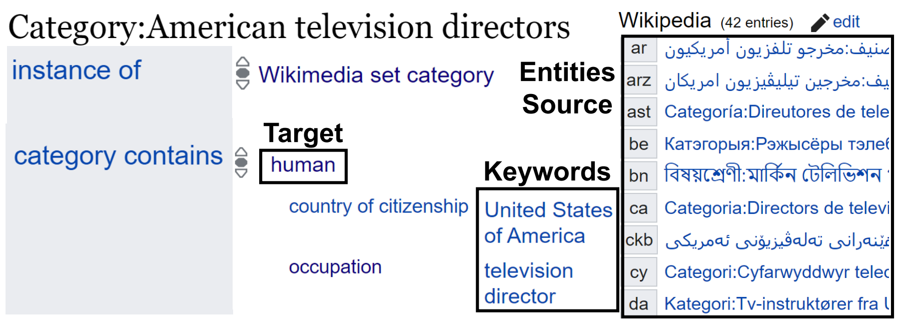

Figure 1: *Category:American television directors (Q8032156).*

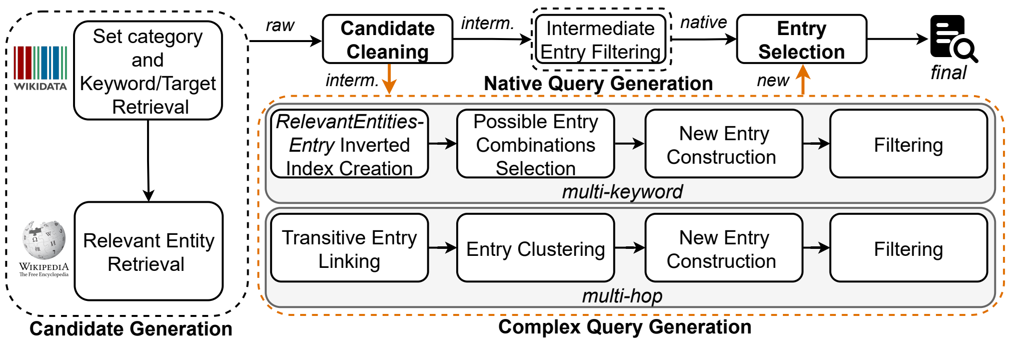

Figure 2: Dataset generation workflow.

entities and TS queries. We leverage the multilingual and hierarchical nature of Wikipedia set category pages to improve the completeness of relevant entities. Our automated approach does not mirror the steps of KSKG systems, but automatically extracts and combines information from manually curated resources to reduce the additional manual effort. Consequently, we consider KeySearchWiki more closely related to human-curated datasets than purely synthetic ones.

3. Dataset Generation Workflow

KeySearchWiki specifically focuses on the TS Task. Our primary goal is to lower manual effort and propose a fully automated workflow for dataset generation. A further goal is the creation of complex and diverse queries. We notice that Wikidata set categories exhibit TS-like characteristics (cf. Figure 1). Most set categories have a property *category contains (P4224)* providing the type (target) of entities contained and additional qualifiers (keywords). This provides the building blocks to construct TS-like queries. Links to corresponding Wikipedia pages can provide relevant entities, as they represent human-curated collections of Wikipedia articles (Wikidata entities) for these categories. Figure 2 depicts the dataset generation workflow whose details will be described in the following.

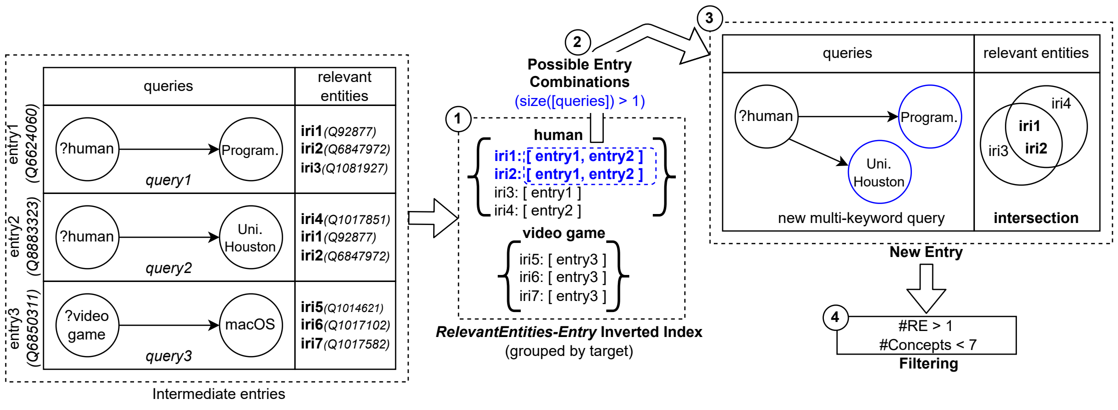

Figure 3: Example pipeline for multi-keyword entries generation.

3.1. Candidate Generation and Cleaning

The pipeline starts by generating candidate entries (queries and their relevant entities). Set categories are retrieved from Wikidata in one of two ways: (1) sending SPARQL queries to the Wikidata public endpoint[9] or (2) parsing a Wikidata JSON dump[10]. Both options retrieve set categories with additional information such as *category contains (P4224)* and its qualifiers used to determine target and keywords respectively. Next, for each available language, the Wikipedia subcategory hierarchy is explored in a Breadth-First-Search-manner to retrieve member pages and their corresponding Wikidata entities. Again, these may be retrieved online using the MediaWiki API[11] or offline from a local database built from SQL Dumps for all needed languages[12]. For each subcategory, we perform a *type check*: if fewer than 50% of its members are instances of the target or any of its subclasses[13], traversal in this branch will be stopped. The output of this phase is a list of *raw entries*. In a cleaning phase, we then remove entries without target or keywords, with more than one keyword/target (ambiguous), without relevant entities, or with a keyword having either an unknown value or no label. The resulting *intermediate entries* act as input for the two following branches.

3.2. Native Query Generation

This phase contains a single operation, *Intermediate Entry Filtering*. We define two criteria: (1) number of relevant entities (#RE) of intermediate entries and (2) number of keywords/target (#Concepts) of corresponding queries[14]. We keep entries whose #RE is at least equal to two, since TS aims at retrieving a list of entities that could be ranked afterwards. Furthermore, we

[9]https://query.wikidata.org/
[10]https://dumps.wikimedia.org/wikidatawiki/entities/
[11]https://www.mediawiki.org/wiki/API:Main_page
[12]Three SQL dumps for each language: categorylinks, page, and page_props.
[13]SPARQL: *?entity wdt:P31/wdt:P279* ?target*
[14]The #Concepts is always equal or lower than #Words (e.g., for query : *"University of Houston" "human"*, $\#Concepts = 2$ and $\#Words = 4$).

only keep queries having #Concepts below 7. The rationale is to reflect real-world user behavior, where generally a small number of keywords is used [29, 30].

3.3. Complex Query Generation

At this stage, two types of complex queries are constructed: multi-keyword and multi-hop queries[15].

Multi-keyword. The process for multi-keyword entries generation is illustrated in Figure 3 based on an example iteration from the actual generation pipeline. The pipeline takes the intermediate entries as input. For the sake of simplicity, consider having three intermediate entries. Each entry corresponds to a set category given by a Wikidata IRI (e.g., *Category:Computer programmers (Q6624060)*) and contains a query and a set of relevant entities. Similarly, relevant entities are represented by Wikidata IRIs. For convenience, we use simple identifiers in the illustration (e.g., *entry1, iri1*). Each query is given by a target and a set of keywords (e.g., *query1: "Programmer" "human"* corresponds to instances of *human* that are described by the keyword *Programmer*). Multi-keyword queries combine a number of queries that have the same target to create a new query (e.g., *query1* and *query2* result in a new query *"programmer" "University of Houston" "human"*). To detect possible combinations, a *RelevantEntities-Entry Inverted Index* is created in step ①. This index aggregates entries that share at least one relevant entity. Separate indices are created for each target to group only compatible queries. From those indices, we select elements containing at least two different entries as *Possible Entry Combinations* in step ②. The *New Entry Construction* step ③ involves the creation of new queries and the new relevant entity sets. New queries are created by merging the keywords and maintaining the shared target: *"programmer" "University of Houston" "human"*. The new relevant entity set is the intersection of relevant entities of the involved entries: *(iri1, iri2)*. Multi-keyword entries are then *filtered* in step ④ based on the criteria defined in Subsection 3.2.

Multi-hop. The steps for multi-hop entry generation are explained in Figure 4, also using a real iteration. The pipeline takes the intermediate entries as input. We consider four intermediate entries as example and use simple identifiers for the sake of convenience (e.g., *entry1, CH (first two letters of entity label), or iri1*). The algorithm traverses all intermediate entries by applying steps ① to ④. We consider one iteration (*entry1*). Multi-hop queries (2 hops for now) link two entries where a relevant entity of one query is equal to a keyword of another query. For example, from *query1 "World Music Awards" "human"* and *query2 "EL" "album"*, we can derive a new query *"World Music Awards" "album"*. In contrast to *query1* and *query2*, in the new query there is no direct relation between target and keywords, i.e., *album* and *World Music Awards.* A system needs to use another intermediate entity (e.g., *EL*, an artist that won a *World Music Awards*) to connect keyword(s) and target. The *Transitive Entry Linking* ① links Relevant Entities of the Current Entry (CERE) to other entries using one of those relevant entities as keyword. Then an *Entry Clustering* ② groups linked entities by target and keywords different from the CERE. With this, new entries can be constructed. The *new multi-hop query* is built in step ③ by merging cluster keys (*album*) and current entry keywords (*World Music Awards, album*).

[15]Based on Figure 1, native queries could be seen as 1-hop queries since they include keywords that are directly related to the target (e.g., entities of type *human (Q5)* directly related to *television director (Q2059704)* via the property *occupation (P106)*).

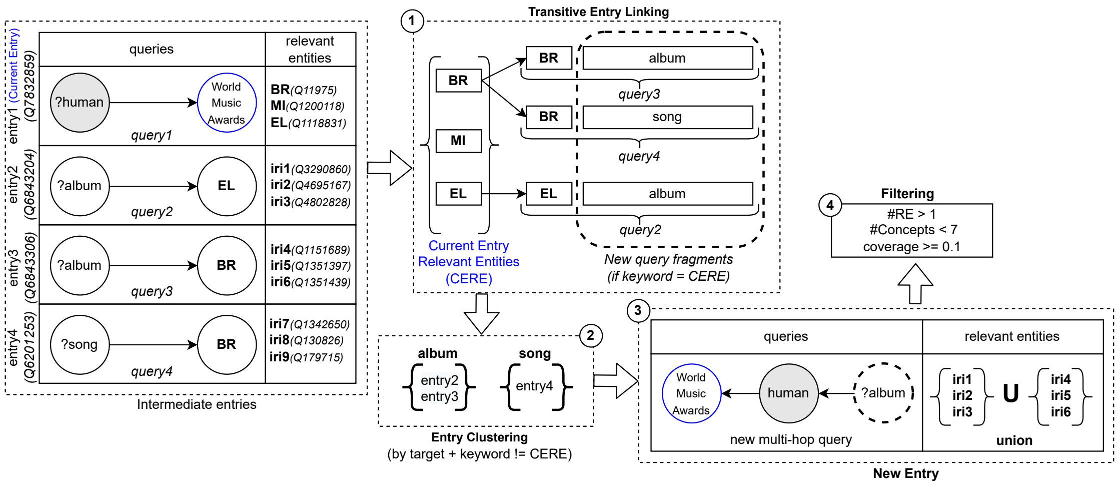

Figure 4: Example pipeline for multi-hop entries generation.

Clusters with the same target as the current entry are removed since they generate the same query (if the cluster has no keywords), or multi-keyword queries. Relevant entities of the new entry are derived from the union of relevant entities in the corresponding cluster. For example, *"World Music Awards" "albums"* are albums from all human artists that won the *World Music Awards* (here, *CH, MI and EL*). Applying the same algorithm recursively, yields queries of more than two hops. For now, we limit the number of hops to two, though. The last step is *Filtering* ④ using the criteria of Subsection 3.2 (#RE and #Concepts) as well as *coverage*. We define the *coverage* as $\frac{cluster\ size}{CERE\ size}$, where $cluster\ size$ is the number of entries in the respective cluster and $CERE\ size$ is the number of relevant entities of the current entry. This metric represents the completeness of the relevant entity set with regard to the new query. In the example of Figure 4, the coverage of the new query is $0.66 \sim \frac{2}{3}$ (in the actual dataset, the query *World Music Awards album* has $coverage = 0.44$ with $cluster\ size = 109$ and $CERE\ size = 250$). A coverage of 1 is not reached here as no linking with relevant entity *MI* was found, i.e., the entry *MI album* does not exist among the input entries. In general, this indicates a missing set category for this combination. We empirically derived a minimum coverage requirement of 0.1 after analyzing the distribution for all multi-hop queries.

3.4. Entry Selection

This step aims to ensure the *diversity* of generated entries. From sets of structurally highly similar queries, one representative is chosen while others are discarded (e.g., *"California State University, Fullerton" "human"* is semantically highly similar to *"University of Houston" "human"*). We define the *query signature* as: *<Target> <Keyword-Types>* (e.g., signature of *"University of Houston" "human"* is *<human (Q5)> <university (Q3918), public educational institution of US (Q23002039)>*). The three types of entries are merged (native/multi-keyword/multi-hop) and grouped by their signature. From each group, one representative of each entry type is selected.

For native/multi-keyword entries, a pseudo-random[16] selection is performed, whereas from multi-hop entries, the one with the highest *coverage* is selected.

4. Dataset Characteristics and Availability

The current dataset version is generated using Wikidata JSON dump and Wikipedia SQL dumps of 2021-09-20[17]. The final KeySearchWiki dataset consists of 16, 605 *final entries (native:* 1, 138*, multi-keyword:* 15, 354*, multi-hop:* 113*)* with 3, 899, 135 unique relevant entities. It involves 73 different targets, 2, 797 unique keywords, and 739 different keyword types. *Human (Q5)* is the most frequent target (13, 260), followed by *album (Q482994)* (1, 815), *video game (Q7889)* (757), and *song (Q7366)* (303). Other insights about the dataset are documented on GitHub[18].

The source code for KeySearchWiki is publicly available [31, 32] under an MIT License, including a description of the dataset, its usage and characteristics, examples, and steps needed to reproduce it. We publish our data on Zenodo [33] under a CC-BY 4.0 License to ensure persistent and public access to all resources. The current dataset is provided in both TREC[19] and JSON formats. The TREC format represents relevant entities and their RJs as follows[20]: *<queryID> 0 <RelevantEntityIRI> <judgment>*. Queries are in a separate text file, following the format in DBpedia-Entity v2 [15]: *<queryID> <query>*. We provide two types of query files: one where queries are given by labels (e.g., *<MK79540> <programmer University of Houston human>* and a second with entity IRIs (e.g., *<MK79540> <Q5482740 Q1472358 Q5>*). The latter can be directly used by systems that omit a preceding entity linking step. We also provide an additional list of queries that was partially adjusted (*naturalized*) to better reflect natural language query formulation. For example, by transforming the query *diplomat Germany 20th century human* into *diplomat Germany 20th century.* This is done by removing the target from the query if one of its keywords is a descendant of the target via *subclass of (P279).* In the previous example, *diplomat* is in the subclass hierarchy of *human.* Following this process, 1, 826 queries were adjusted and the whole list was provided using the same format: *<queryID> <query>*. The provided data contains:

- **KeySearchWiki-JSON** - the final dataset in JSON format.
- **KeySearchWiki-queries-label** - a text file containing the 16, 605 queries, each line containing space-separated queryID and query text (labels).
- **KeySearchWiki-queries-iri** - a text file containing the 16, 605 queries, each line containing space-separated queryID and IRIs of query elements.
- **KeySearchWiki-queries-naturalized** - a text file with all 16, 605 queries, including 1, 826 adjusted queries, each line containing space-separated queryID and query text (labels).

[16]Entries in a group are sorted by queryID. Then the first element is selected to ensure a deterministic behavior for reproducibility.

[17]A version where the Wikidata "Wikimedia set categories (Q59542487)" were not yet merged with their initially superclasses "Wikimedia categories (Q4167836)". https://github.com/fusion-jena/KeySearchWiki/blob/master/README.md#remark

[18]https://github.com/fusion-jena/KeySearchWiki/tree/master/docs#dataset-characteristics

[19]https://trec.nist.gov/data/qrels_eng/

[20]The second field is unused and set to 0 according to the TREC qrels format.

Table 2
Experiment results of the baseline methods.

Method	**Native**		**Multi-Keyword**		**Multi-hop**	
	MAP	P@10	MAP	P@10	MAP	P@10
BM25	0.211	**0.225**	**0.025**	**0.039**	0.014	**0.032**
DFR	0.209	0.211	0.023	0.029	0.015	0.024
LM Dirichlet	0.182	0.180	0.020	0.025	0.015	0.018
LM Jelinek-Mercer	**0.212**	0.215	0.023	0.029	**0.018**	0.022

- **KeySearchWiki-qrels-trec** - a text file containing relevant entities in TREC format.
- **KeySearchWiki-cache** [34] - a collection of SQLite database files containing all the data retrieved from Wikidata JSON Dump and Wikipedia SQL Dumps of 2021-09-20.

Users can update KeySearchWiki anytime by running our code on a new dump of Wikidata and Wikipedia. We plan to periodically publish new dataset releases.

5. Experiments and Evaluation

We use KeySearchWiki to evaluate Elas4RDF [35], a system based on indexing the textual information of entities. Elas4RDF relies on a triple-based indexing using Elasticsearch[21]. We use the best-performing approach where each triple is represented by a document with the following fields: subject/predicate/object keywords[22], description of IRI object/subject, and label of IRI object/subject. Object fields are given higher weight. We use the Elas4RDF-index Service[23] to create an index of Wikidata entity triples. We perform our experiments on a subset of queries. Considering all dataset queries would imply indexing triples involving all Wikidata entities. We avoid that to keep the indexing time reasonable by selecting queries with one of the top-10 targets and thus index triples involving Wikidata entities that are either instances of the target itself or any of its subclasses. This way we keep 99% (only 112 queries discarded) of the queries from all the types *(native:* $1,037$*, multi-keyword:* $15,343$*, multi-hop: 113).* $146,211,253$ triples were indexed with an index size of 16.8 GB. We evaluate four ranking methods provided by Elasticsearch with default settings as in [35]: BM25 [36], DFR [37], LM Dirichlet [38], and LM Jelinek-Mercer [38]. For each baseline (run), the Elas4RDF-search Service[24] is used to retrieve the results which are then written in TREC format: <queryID> Q0 <RetrievedEntityIRI> <rank> <score> <runID>. The second column is unused and should always be “Q0”.

Table 2 summarizes the experiment results. We use Mean Average Precision (MAP) and Precision at rank 10 (P@10) (considering the top-1000 results). We notice that the different query types reflect various degrees of difficulty. This corresponds to our intention of adding complex queries. Native queries are less challenging and thus achieve better results across

[21]https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

[22]Represented by the literal value. If a triple component is not a literal, the IRI’s namespace part is removed and the remainder is tokenized into keywords.

[23]https://github.com/SemanticAccessAndRetrieval/Elas4RDF-index (adapted to Wikidata)

[24]https://github.com/SemanticAccessAndRetrieval/Elas4RDF-search (adapted to Wikidata)

retrieval methods. These queries usually involve keywords that are directly related and hence their textual information is mostly occurring within the triples of the same entity. Complex queries are more difficult and show poor performance in general. Even though multi-keyword queries still involve directly related keywords, they tend to be longer and thus seem more challenging. Here, the performance has dropped by ~ 0.18 points for both MAP and P@10 compared to native queries. Multi-hop queries are more difficult than their multi-keyword counterparts as they involve keywords not directly related and thus their textual information does not occur within triples of the same entity. This lowers the performance by ~ 0.19 points compared to native queries and by ~ 0.01 points compared to the multi-keyword ones. The results reveal no noticeable difference between the different retrieval methods. We only notice an improvement between $\sim 0.01 - 0.05$ points of the P@10 using BM25 for native queries. A more detailed investigation is out of the scope for this paper. Overall, the performance of the ranking methods over KeySearchWiki is in line with other published results (e.g., in [15] and [39]). Further details about experiment data preparation, indexing, and the experimental setup are provided in the dataset's GitHub repository [31]. Runs, experiment results, queries, and relevance judgments are published on Zenodo [40].

Evaluation. We evaluate the *accuracy* of relevant entities in KeySearchWiki by using existing SPARQL queries as a baseline and comparing KeySeachWiki's relevant entities with the results of these queries. Evaluation scripts and results are available on GitHub. Some Wikidata set categories have associated SPARQL queries using the property *Wikidata SPARQL query equivalent (P3921)*. These queries retrieve results corresponding to the set category and are handcrafted by humans. They can be considered as another source of relevant entities (baseline) that can be used for comparison and verification of the relevant entities provided by KeySearchWiki. We extract native entries that contain such SPARQL queries (67 native entries) and manually verify whether the corresponding SPARQL queries correctly represent the information need expressed by the set category. One query was excluded which results in 66 queries used in this evaluation. For the selected queries, we calculate the *Precision* and *Recall* of KeySearchWiki entities $\{RE_{WIKI}\}$ with respect to SPARQL query results $\{RE_{SPARQL}\}$ [41]: $Precision = \frac{|\{RE_{SPARQL}\} \cap \{RE_{WIKI}\}|}{|\{RE_{WIKI}\}|}$, $Recall = \frac{|\{RE_{SPARQL}\} \cap \{RE_{WIKI}\}|}{|\{RE_{SPARQL}\}|}$. Results reveal that KeySearchWiki is capable of catching most of the relevant entities retrieved by SPARQL resulting in an *Average Recall* of ~ 0.70 and an *Average Precision* of ~ 0.54. A more detailed analysis of the results is provided in GitHub[25].

6. Limitations

In the following we describe the limitations of KeySearchWiki.

Matching between data sources and the target KG. In general, even though each Wikipedia page has its corresponding Wikidata entity, the two sources do not always match with respect to the knowledge contained. This is due to the fact that both are mostly independently maintained by volunteers. Despite the overlap and collaboration between both communities, the information in both projects will probably continue to differ in the foreseeable future. Other benchmarks also suffer from this – especially those that collect queries independently from the actual KG

[25] https://github.com/fusion-jena/KeySearchWiki/tree/master/docs#evaluation-results

(e.g., from logs). We attempt to mitigate the effects by using closely related sources (Wikipedia and Wikidata) for queries and relevant entities.

Completeness of relevant results. Depending on the approach, completeness is rather hard to achieve for benchmarks of reasonable size. Human relevance judgments may be feasible for smaller datasets, but fail for larger ones that are built using pooling [42]. We try to increase the completeness by considering all Wikipedia languages and traversing its hierarchy. Furthermore, KeySearchWiki uses Wikipedia as relevant entity source to include also Wikidata entities with missing semantic description.

Evaluation of approaches exploiting Wikipedia categories. Systems following KeySearch-Wiki's strategy of exploiting Wikipedia categories may achieve close to perfect scores. However, the task of KSKG assumes only two inputs: a query and a target KG. Additional sources alter this task and result in systems heavily depending on a particular KG. In such scenarios (system using Wikipedia categories), the dataset should not be used to avoid any bias.

7. Conclusion and Future Work

We introduced KeySearchWiki - a fully automatically generated, complex, large-scale, and diverse dataset for evaluating keyword search systems over Wikidata. We leverage Wikidata and Wikipedia set categories as data sources for both relevant entities and queries. We gather relevant entities by carefully navigating the Wikipedia set categories hierarchy in all available languages. In the future, we plan to extend the dataset by also generalizing to *Wikimedia categories (Q4167836)* that are superclasses of the currently used set categories. This will allow us to increase the number of dataset entries and to also generate more high-coverage multi-hop entries.

8. Acknowledgments

This work has been funded by the German Aerospace Center (DLR). We thank Prof. Dr. Birgitta König-Ries for guidance and feedback.

References

[1] K. Balog, Entity-Oriented Search, volume 39 of *The Information Retrieval Series*, Springer, 2018. doi:10.1007/978-3-319-93935-3.

[2] P. Wu, X. Zhang, Z. Feng, A survey of question answering over knowledge base, in: Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding, Springer Singapore, Singapore, 2019, pp. 86–97. doi:10.1007/978-981-15-1956-7_8.

[3] D. Dosso, G. Silvello, Search text to retrieve graphs: A scalable RDF keyword-based search system, IEEE Access 8 (2020) 14089–14111. doi:10.1109/ACCESS.2020.2966823.

[4] L. Feddoul, Semantics-driven keyword search over knowledge graphs, in: Proceedings of the Doctoral Consortium at ISWC 2020 co-located with 19th International Semantic Web Conference (ISWC 2020), Athens, Greece, November 3rd, 2020, volume 2798 of *CEUR*

Workshop Proceedings, CEUR-WS.org, 2020, pp. 17–24. URL: https://ceur-ws.org/Vol-2798/paper3.pdf.

[5] A. Ghanbarpour, K. Niknafs, H. Naderi, Efficient keyword search over graph-structured data based on minimal covered r-cliques, Frontiers Inf. Technol. Electron. Eng. 21 (2020) 448–464. doi:10.1631/FITEE.1800133.

[6] Y. Shi, G. Cheng, E. Kharlamov, Keyword search over knowledge graphs via static and dynamic hub labelings, in: Proceedings of The Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 235–245. doi:10.1145/3366423.3380110.

[7] E. S. Menendez, M. A. Casanova, L. A. P. Paes Leme, M. Boughanem, Novel node importance measures to improve keyword search over RDF graphs, in: Database and Expert Systems Applications, Springer International Publishing, Cham, 2019, pp. 143–158. doi:10.1007/978-3-030-27618-8_11.

[8] M. Rihany, Z. Kedad, S. Lopes, Keyword search over RDF graphs using WordNet, in: Proceedings of the 1st International Conference on Big Data and Cyber-Security Intelligence, BDCSIntell 2018, Hadath, Lebanon, December 13-15, 2018, volume 2343 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 75–82. URL: https://ceur-ws.org/Vol-2343/paper15.pdf.

[9] S. Han, L. Zou, J. X. Yu, D. Zhao, Keyword search on RDF graphs - a query graph assembly approach, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 227–236. doi:10.1145/3132847.3132957.

[10] Y. Shan, M. Li, Y. Chen, Constructing target-aware results for keyword search on knowledge graphs, Data & Knowledge Engineering 110 (2017) 1–23. doi:https://doi.org/10.1016/j.datak.2017.02.001.

[11] Q. Wang, J. Kamps, G. R. Camps, M. Marx, A. Schuth, M. Theobald, S. Gurajada, A. Mishra, Overview of the INEX 2012 linked data track, in: CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17-20, 2012, volume 1178 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2012. URL: https://ceur-ws.org/Vol-1178/CLEF2012wn-INEX-WangEt2012.pdf.

[12] H. Halpin, D. M. Herzig, P. Mika, R. Blanco, J. Pound, H. Thompon, D. T. Tran, Evaluating ad-hoc object retrieval, in: Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010), Shanghai, China, November 8, 2010, volume 666 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2010. URL: https://ceur-ws.org/Vol-666/paper9.pdf.

[13] R. Blanco, H. Halpin, D. Herzig, P. Mika, J. Pound, H. Thompson, D. Tran, Entity search evaluation over structured web data, in: Proceedings of the 1st International Workshop on Entity-Oriented Search at SIGIR 2011, 28.07.2011, Beijing, China, TU Delft, 2011, pp. 65 – 71.

[14] P. Bellot, A. Doucet, S. Geva, S. Gurajada, J. Kamps, G. Kazai, M. Koolen, A. Mishra, V. Moriceau, J. Mothe, M. Preminger, E. SanJuan, R. Schenkel, X. Tannier, M. Theobald, M. Trappett, Q. Wang, Overview of INEX 2013, in: Information Access Evaluation. Multilinguality, Multimodality, and Visualization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 269–281. doi:10.1007/978-3-642-40802-1_27.

[15] F. Hasibi, F. Nikolaev, C. Xiong, K. Balog, S. E. Bratsberg, A. Kotov, J. Callan, DBpedia-Entity v2: A test collection for entity search, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1265–1268. doi:10.1145/3077136.3080751.

[16] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia, Semantic Web 6 (2015) 167–195. doi:10.3233/sw-140134.

[17] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative knowledgebase, Commun. ACM 57 (2014) 78–85. doi:10.1145/2629489.

[18] J. Pound, P. Mika, H. Zaragoza, Ad-hoc object retrieval in the web of data, in: Proceedings of the 19th International Conference on World Wide Web, WWW '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 771–780. doi:10.1145/1772690.1772769.

[19] G. Demartini, T. Iofciu, A. P. de Vries, Overview of the INEX 2009 entity ranking track, in: Focused Retrieval and Evaluation, Springer, 2010, pp. 254–264. doi:10.1007/978-3-642-14556-8_26.

[20] A. Harth, Billion Triples Challenge data set, Downloaded from http://km.aifb.kit.edu/projects/btc-2009/, 2009.

[21] R. Usbeck, R. H. Gusmita, A. N. Ngomo, M. Saleem, 9th challenge on question answering over linked data (QALD-9) (invited paper), in: Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018, volume 2241 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2018, pp. 58–64. URL: https://ceur-ws.org/Vol-2241/paper-06.pdf.

[22] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia, in: The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 69–78. doi:10.1007/978-3-030-30796-7_5.

[23] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, G. Weikum, YAGO: A multilingual knowledge base from Wikipedia, Wordnet, and Geonames, in: The Semantic Web – ISWC 2016, Cham, 2016, pp. 177–185. doi:10.1007/978-3-319-46547-0_19.

[24] J. Coffman, A. C. Weaver, A framework for evaluating database keyword search strategies, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 729–738. doi:10.1145/1871437.1871531.

[25] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, USA, 2008.

[26] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, LC-QuAD: A corpus for complex question answering over knowledge graphs, in: The Semantic Web – ISWC 2017, Springer International Publishing, Cham, 2017, pp. 210–218. doi:10.1007/978-3-319-68204-4_

22.
[27] A. B. Neves, L. A. P. P. Leme, Y. T. Izquierdo, M. A. Casanova, Automatic construction of benchmarks for RDF keyword search systems evaluation, in: Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS 2021, SCITEPRESS, 2021, pp. 126–137. doi:10.5220/0010519401260137.
[28] A. Orogat, A. El-Roby, Maestro: Automatic generation of comprehensive benchmarks for question answering over knowledge graphs, Proc. ACM Manag. Data 1 (2023) 177:1–177:24. doi:10.1145/3589322.
[29] A. Spink, D. Wolfram, M. B. J. Jansen, T. Saracevic, Searching the web: The public and their queries, J. Am. Soc. Inf. Sci. Technol. 52 (2001) 226–234. doi:10.1002/1097-4571(2000)9999:9999<::AID-ASI1591>3.0.CO;2-R.
[30] G. Pass, A. Chowdhury, C. Torgeson, A picture of search, in: Proceedings of the 1st International Conference on Scalable Information Systems, InfoScale '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 1–es. doi:10.1145/1146847.1146848.
[31] L. Feddoul, S. Schindler, fusion-jena/KeySearchWiki, 2022. URL: https://github.com/fusion-jena/KeySearchWiki.
[32] L. Feddoul, S. Schindler, fusion-jena/KeySearchWiki v1.2.2, 2023. doi:10.5281/zenodo.8016819.
[33] L. Feddoul, F. Löffler, S. Schindler, KeySearchWiki, 2022. doi:10.5281/zenodo.6010301.
[34] L. Feddoul, F. Löffler, S. Schindler, KeySearchWiki-cache, 2021. doi:10.5281/zenodo.5752018.
[35] G. Kadilierakis, P. Fafalios, P. Papadakos, Y. Tzitzikas, Keyword search over RDF using document-centric information retrieval systems, in: The Semantic Web, Springer International Publishing, Cham, 2020, pp. 121–137. doi:10.1007/978-3-030-49461-2_8.
[36] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389. doi:10.1561/1500000019.
[37] G. Amati, C. J. Van Rijsbergen, Probabilistic models of information retrieval based on measuring the divergence from randomness, ACM Trans. Inf. Syst. 20 (2002) 357–389. doi:10.1145/582415.582416.
[38] C. Zhai, J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01, Association for Computing Machinery, New York, NY, USA, 2001, p. 334–342. doi:10.1145/383952.384019.
[39] K. Balog, R. Neumayer, A test collection for entity search in DBpedia, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 737–740. doi:10.1145/2484028.2484165.
[40] L. Feddoul, F. Löffler, S. Schindler, KeySearchWiki-experiments, 2022. doi:10.5281/zenodo.6010349.
[41] L. Feddoul, F. Löffler, S. Schindler, Analysis of consistency between Wikidata and Wikipedia categories, in: Proceedings of the 3rd Wikidata Workshop 2022 co-located with the 21st International Semantic Web Conference (ISWC2022), Virtual Event, Hanghzou, China,

October 2022, volume 3262 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3262/paper4.pdf.

[42] C. Buckley, E. M. Voorhees, Retrieval evaluation with incomplete information, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 25–32. doi:10.1145/1008992.1009000.