AFRL-RI-RS-TR-2023-111

KGTK: KNOWLEDGE GRAPH TOOLKIT

UNIVERSITY OF SOUTHERN CALIFORNIA

JUNE 2023

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2023-111 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
NANCY A. SEPULVEDA
Work Unit Manager

/ S /
MATTHEW J. KOCHAN
Technical Advisor,
Intelligence Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
JUNE 2023	FINAL TECHNICAL REPORT	MAY 2020	DECEMBER 2022

4. TITLE AND SUBTITLE

KGTK: Knowledge Graph ToolKit

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-20-2-1002	N/A	63760E

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
		R30E

6. AUTHOR(S)
Filip Ilievski

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California, Information Sciences Institute
4676 Admiralty Way Ste. 1001
Marina del Rey CA 90292

8.PERFORMING ORGANIZATIONREPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10.SPONSOR/MONITOR'S ACRONYM(S)	11.SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RIEA 525 Brooks Road Rome NY 13441-4505	AFRL/RI	AFRL-RI-RS-TR-2023-111

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
KGTK is a comprehensive framework for the creation and exploitation of large knowledge graphs, designed for simplicity, scalability, and interoperability. Its key quality attributes are: 1) native support for reading and writing knowledge graphs in many formats; 2) extensibility, by seamless import and export to popular data science tools like Pandas and ElasticSearch; 3) modularity, i.e., pipeline-friendly design to create workflows with multiple components. 4) speed, i.e., to be comparably fast to SQL databases; and 5) scale to billions of statements, e.g., handle all Wikidata on a laptop. KGTK represents graphs in tables and leverages popular libraries developed for data science applications, enabling a wide audience of developers to easily construct knowledge graph pipelines for their applications. KGTK has dozens of commands, covering a wide range of imports and exports to popular formats, highly scalable querying and storage functionality, a rich and diverse suite of transformation functions, and modern analytics powered by machine learning and graph algorithms. KGTK provides several services, namely a text search interface, a user-friendly browser, a similarity interface, and a SPARQL endpoint. All services of KGTK can be readily customized to arbitrary graphs, thus closing the loop and enabling users to use KGTK seamlessly with their own data in a variety of formats.

15. SUBJECT TERMS
knowledge graphs, large-scale network analysis, wikidata

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	34

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
NANCY A. SEPULVEDA	**N/A**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

Tasks involving knowledge graphs (KGs) are difficult because they frequently require complex workflows that span different tools and algorithms, consume data in different forms, and use diverse systems. For example, linking entities to a reference KG might involve steps such as building an index over KG entities and their types, learning an embedded vector representation of each entity, computing entity centrality (e.g., PageRank), and developing a statistical model for ambiguous entity names. Composing such a pipeline today requires knowledge of multiple tools that are not designed to work together, making it costly to implement these pipelines for large KGs and hindering massive adoption of knowledge graphs by AI developers and researchers. Guided by our belief that AI researchers and developers should be able to easily exploit the wealth of knowledge available in KGs with minimal effort, we have developed KGTK: an integrated knowledge graph toolkit, which provides a wide range of knowledge graph manipulation and analysis functionality, and supports common use cases such as large-scale network analysis, data enrichment, and quality estimation. Other available tools such as kglab and graphy address similar use-cases as KGTK, but contain a small subset of the tools available in KGTK and provide no support for working with hypergraphs (edges with qualifiers) such as Wikidata.

KGTK is a comprehensive framework for the creation and exploitation of large knowledge graphs, designed for simplicity, scalability, and interoperability. Its key quality attributes are: 1) native support for reading and writing knowledge graphs in many formats; 2) extensibility, by seamless import and export to popular data science tools like Pandas and ElasticSearch; 3) modularity, i.e., pipeline-friendly design to create workflows with multiple components. 4) speed, i.e., to be comparably fast to SQL databases; and 5) scale to billions of statements, e.g., handle all Wikidata on a laptop. KGTK represents graphs in tables and leverages popular libraries developed for data science applications, enabling a wide audience of developers to easily construct knowledge graph pipelines for their applications. KGTK has dozens of commands, covering a wide range of imports and exports to popular formats, highly scalable querying and storage functionality, a rich and diverse suite of transformation functions, and modern analytics powered by machine learning and graph algorithms. KGTK provides several services, namely a text search interface, a user-friendly browser, a similarity interface, and a SPARQL endpoint. All services of KGTK can be readily customized to arbitrary graphs, thus closing the loop and enabling users to use KGTK seamlessly with their own data in a variety of formats.

KGTK has a proven track record across many use cases, including KGs for financial transactions, scientific publications, geopolitical events, internet memes, and commonsense knowledge. We have used KGTK to analyze the quality of hundreds of dumps of Wikidata, to devise large-scale algorithms for entity summarization and detection of surprising facts, and to enrich knowledge graphs based on alignment with other linked data sources. KGTK has enabled published applications by academic peers on analyzing trust in KGs and mining emerging patterns of reuse in Wikidata. KGTK is open-source, regularly updated, and well-documented. The KGTK team has been proactively reaching out to the AI community, by organizing tutorials about KGTK's toolkit (ISWC'21, The Web Conference '22, Knowledge Graph Conference '22) and a use case-driven lab session at AAAI'23. The community events of KGTK are based on extensive materials, including Google Colab notebooks, comprehensive slides, and documentation pages.

2.0 INTRODUCTION

Knowledge graphs (KGs) have become the preferred technology for representing, sharing and using knowledge in applications. A typical use case is building a new knowledge graph for a domain or application by extracting subsets of several existing knowledge graphs, combining these subsets in application-specific ways, augmenting them with information from structured or unstructured sources, and computing analytics or inferred representations to support downstream applications. For example, during the COVID-19 pandemic, several efforts focused on building KGs about scholarly articles related to the pandemic starting from a dataset provided by the Allen Institute for AI. Enhancing these data with KGs such as DBpedia [1] and Wikidata [2] to incorporate gene, chemical, disease and taxonomic information, and computing network analytics on the resulting graphs, requires the ability to operate these KGs at scale. Many tools exist to query, transform and analyze KGs, including graph databases, triple stores, RDF tools, entity linking tools, libraries to compute graph embeddings, and libraries for graph analytics (Table 1) [3]. There are three main challenges when using these tools together. First, tools may be challenging to set up with large KGs (e.g., the Wikidata RDF dump takes over a week to load into a triple store) and often need custom configurations that require significant expertise. Second, interoperating between tools requires developing data transformation scripts, as some of them may not support the same input/output representation. Third, composing two or more tools together (e.g., to filter, search, and analyze a KG) includes writing the intermediate results to disk, which is time and memory consuming for large KGs.

Table 1. KG application pipelines require a diverse set of tools.

Operation	Tool	Format(s)	Language
graph analytics	graph-tool	GML, GT, TSV/CSV	python
	NetworkX	GML, JSON, TSV/CSV	python
graph database	Neo4J	TSV/CSV	*various*
RDF operations	graphy	RDF	javascript
	Jena	RDF	java
	RDFLib	RDF	python
graph embeddings	PyTorch-BigGraph	TSV/CSV	python
entity resolution	RLTK	TSV/CSV	python
entity linking	AGDISTIS	XML	python
	WAT	JSON	python

To address these pain points, we developed Knowledge Graph Toolkit (KGTK), a novel framework for manipulating, validating, and analyzing large-scale KGs. Our work is inspired by popular toolkits for machine learning and natural language processing that have had a vast impact by making these technologies accessible to data scientists and software developers. The goal of KGTK was to build a comprehensive library of tools and methods to enable easy composition of KG operations (validation, filtering, merging, centrality, text embeddings, etc.) to build knowledge-based AI applications. The contributions of KGTK are:

- The KGTK file format, which allows representing KGs as hypergraphs. This format unifies the Wikidata data model [2] based on items, claims, qualifiers, and references; property graphs that support arbitrary attributes on nodes and edges; RDF-Schema-based graphs such as DBpedia [1]; and general-purpose RDF graphs with various forms of reification. The KGTK format uses tab-separated values (TSV) to represent edge lists, making it easy to process with many off-the-shelf tools.
- A comprehensive validator and data cleaning module to verify compliance with the KGTK format, and normalize literals like strings and numbers.
- Import modules to transform different formats into KGTK, including N-Triples and Wikidata qualified terms.
- Graph manipulation modules for bulk operations on graphs to validate, clean, filter, join, sort, and merge KGs. Several of these are implemented as wrappers of common, streaming Unix tools like awk, sort, and join.
- Graph querying and analytics modules to compute centrality measures, connected components, and text-based graph embeddings using state-of- the-art language models. Common queries, such as computing the set of nodes reachable from other nodes, are also supported.
- Export modules to transform KGTK format into diverse standard and commonly used formats, such as RDF (N-Triples) and property graphs in Neo4J format.
- A framework for composing multiple KG operations, based on Unix pipes. The framework uses the KGTK file format on the standard input and output to combine tools written in different programming languages.
- Services for user-friendly access to its graphs via similarity queries, text search, faceted browsing, and SPARQL systematic queries.

KGTK provides an implementation that integrates all these methods relying on widely used tools and standards, thus allowing their composition in pipelines to operate with large KGs like Wikidata on an average laptop. KGTK has enabled us to support a long list of use cases, ranging from estimating surprise in knowledge graphs to integrating and consolidating internet memes.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

The work on the KGTK toolkit can be broadly split into three directions: development of the core toolkit, development of customizable services, and use cases. We describe our work on each of these three directions in turn.

3.1 Motivating scenario

The 2020 coronavirus pandemic led to a series of community efforts to publish and share common knowledge about COVID-19 using KGs. Many of these efforts use the COVID-19 Open Research Dataset (CORD-19) [4], compiled by the Allen Institute for AI. CORD-19 is a free resource containing over 44,000 scholarly articles, including over 29,000 with full text, about COVID-19 and the coronavirus family of viruses. Having an integrated KG would allow easy access to information published in scientific papers, as well as to general medical knowledge on genes, proteins, drugs, and diseases mentioned in these papers, and their interactions.

We integrated the CORD-19 corpus with gene, chemical, disease, and taxonomic knowledge from Wikidata and CTD databases,[1] as well as entity extractions from Professor Heng Ji's BLENDER lab at UIUC. We extracted all the items and statements for the 30,000 articles in the CORD-19 corpus [4] that were present in Wikidata at the time of extraction, added all Wikidata articles, authors, and entities mentioned in the BLENDER corpus, homogenized the data to fix inconsistencies (e.g., empty values), created nodes and statements for entities that were absent in Wikidata, incorporated metrics such as PageRank for each KG node, and exported the output in both RDF and Neo4J.

This use case exhibited several of the challenges that KGTK is designed to address. For example, extracting a subgraph from Wikidata articles is not feasible using SPARQL queries as it would have required over 100,000 SPARQL queries; using RDF tools on the Wikidata RDF dump (107 GB compressed) is difficult because its RDF model uses small graphs to represent each Wikidata statement; using the Wikidata JSON dump is possible, but requires writing custom code as the schema is specific to Wikidata (hence not reusable for other KGs). In addition, while graph-tool allowed us to compute graph centrality metrics, its input format is incompatible with RDF, requiring a transformation.

Other efforts employed a similar set of processing steps [4]. These range from mapping the CORD-19 data to RDF, to adding annotations to the articles in the dataset pointing to entities extracted from the text, obtained from various sources [5]. A common thread among these efforts involves leveraging existing KGs such as Wikidata and Microsoft Academic Graph, for example, to build a citation network of the papers, authors, affiliations, etc. Other efforts focused on extraction of relevant entities (genes, proteins, cells, chemicals, diseases), relations (causes, upregulates, treats, binds), and linking them to KGs such as Wikidata and DBpedia. Graph analytics operations followed, such as computing centrality measures in order to support identification of key articles, people or substances, or generation of various embeddings to recommend relevant literature associated with an entity. The resulting graphs were deployed as SPARQL endpoints, or exported as RDF dumps, CSV, or JSON files.

These examples illustrate the need for composing sequences of integrated KG operations that extract, modify, augment and analyze knowledge from existing KGs, combining it with non-KG datasets to produce new KGs. Existing KG tools do not allow users to seamlessly run such

[1] https://ctdbase.org/

sequences of graph manipulation tasks in a pipeline. We proposed that an effective toolkit that supports the construction of modular KG pipelines has to meet the following criteria [3]:
1. **A simple representation format** that all modules in the toolkit operate on (the equivalent of datasets in Scikit-learn and document model in SpaCy), to enable tool integration without additional data transformations.
2. **Ability to incorporate mature existing tools**, wrapping them to support a common API and input/output format. The scientific community has worked for many years on efficient techniques for manipulation of graph and structured data. The toolkit should be able to accommodate them without the need for a new implementation.
3. **A comprehensive set of features** that include import and export modules for a wide variety of KG formats, modules to select, transform, combine, link, and merge KGs, modules to improve the quality of KGs and infer new knowledge, and modules to compute embeddings and graph statistics. Such a rich palette of functionalities would largely support use cases such as the ones presented in this section.
4. **A pipeline mechanism** to allow composing modules in arbitrary ways to process large public KGs such as Wikidata, DBpedia, or ConceptNet.

3.2 KGTK toolkit

The KGTK toolkit is depicted in Figure 1. KGTK is a comprehensive framework for the creation and exploitation of large knowledge graphs, designed for simplicity, scalability, and interoperability. Its key quality attributes are: 1) native support for reading and writing knowledge graphs in many formats; 2) extensibility, by seamless import and export to popular data science tools like Pandas and ElasticSearch; 3) modularity, i.e., pipeline-friendly design to create workflows with multiple components; 4) speed, i.e., to be comparably fast to SQL databases; and 5) scale to billions of statements, e.g., handle all Wikidata on a laptop.

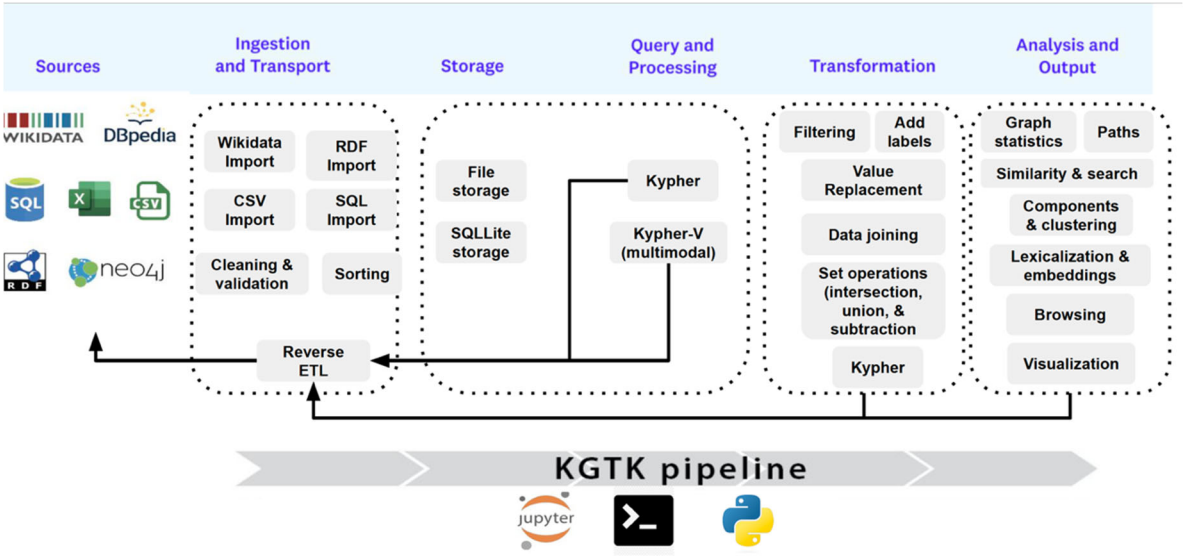

Figure 1. KGTK Toolkit functionality.

3.2.1 File format

The KGTK file format allows representing KGs as hypergraphs. This format unifies the Wikidata data model [2] based on items, claims, qualifiers, and references; property graphs that support arbitrary attributes on nodes and edges; RDF-Schema-based graphs such as DBpedia [1]; and general purpose RDF graphs with various forms of reification. The KGTK format uses tab-separated values (TSV) to represent edge lists, making it easy to process with many off-the-shelf tools. We chose this format instead of an RDF serialization for three reasons. First, tabular formats are easy to generate and parse by standard tools. Second, this format is self-describing and easy to read by humans. Finally, it provides a simple mechanism to define hypergraphs and edge qualifiers, which may be more complicated to describe using Turtle or JSON. KGTK defines KGs as a set of nodes and a set of edges between those nodes. All concepts of meaning are represented via an edge, including edges themselves, allowing KGTK to represent generalized hypergraphs (while supporting the representation of RDF graphs). The snippet below shows a simple example of a KG in KGTK format with three people (Moe, Larry and Curly), the creator of the statements (Hans), and the original source of the statements (Wikipedia):

```
node1           label        node2        creator      source       id
"Moe"           rdf:type     Person       "Hans"       Wikipedia    E1
"Larry"   rdf:type     Person       "Hans"       Wikipedia    E2
"Curly"   rdf:type     Person                    Wikipedia    E3
"Curly"   hasFriend    "Moe"                     Wikipedia    E4
```

The first line of a KGTK file declares the headers for the document. The reserved words *node1, label and node2* are used to denote the subject, property and object being described, while creator and source are optional qualifiers for each statement that provide additional provenance information about the creator of a statement and the original source. Note that the example is not using namespace URIs for any nodes and properties, as they are not needed for local KG manipulation. Still, namespace prefixes (e.g., rdf) may be used for mapping back to RDF after the KG manipulations with KGTK. Nodes and edges have unique IDs (when IDs are not present, KGTK generates them automatically).

3.2.2 KGTK operations

KGTK's operations (Figure 1) can be divided based on their function into 1) ingestion and transport, 2) storage, query, and processing, 3) transformation, and 4) analytics. We describe their contents and list some of the most representative and commonly used operations.

1. Its **ingestion and transport** component transforms data between popular formats and KGTK. The *import* operations transform an external graph format into KGTK TSV format. KGTK supports importing a number of data formats, including N-Triples, ConceptNet, and Wikidata (together with qualifiers). The *export* (inverse ETL) operations transform a KGTK-formatted graph to a wide palette of formats: TSV (by default), N-Triples, Neo4J Property Graphs, graph-tool and the Wikidata JSON format. The *validate* operation ensures that the data meets the KGTK file format specification, detecting errors such as nodes with empty values, values of unexpected length (either too long or too short), potential errors in strings (quotation errors, incorrect use of language tags, etc.), incorrect values in dates, etc. Users may customize the parsing of the file header, each line, and the data values, as well as choose the action taken when a validation rule fails. The *clean* operation fixes a substantial number of errors detected by validate, by correcting some common mistakes in data encoding (such as not escaping 'pipe' characters), replacing invalid

dates, normalizing values for quantities, languages, and coordinates using the KGTK convention for literals. Finally, it removes rows that still do not meet the KGTK specification (e.g., rows with empty values for required columns or rows with an invalid number of columns).

2. KGTK's **storage, query, and processing group** stores its data into an SQLite database and enables scalable queries over this data through the scalable module called Kypher [6]. Kypher stands for KGTK Cypher, and it adopts many aspects of the Neo4J Cypher's [7] query language, with some important differences. To implement Kypher queries, we translate them into SQL and execute them on SQLite, a lightweight file-based SQL database. Kypher queries are designed to look and feel very similar to other file based KGTK commands. They take tabular file data as input and produce tabular data as output. There are no servers or accounts to set up, and users do not need to know that there is in fact a database used underneath to implement the queries. A cache mechanism makes multiple queries over the same KGTK files very efficient. Towards the end of the project, we developed a multimodal variant of Kypher, called Kypher-V [8], which is able to seamlessly and efficiently combine symbolic and subsymbolic patterns, e.g., graph paths and embeddings.
3. **Transformation** functionality includes filtering, joining, sorting, and merging of KGs. Many of these operations are implemented as wrappers of common streaming Unix tools, whereas further custom transformations can be implemented by users based on Kypher. The *sort* operation efficiently reorders any KGTK file according to one or multiple columns. *sort* is useful to organize edge files so that, for example, all edges for node1 are contiguous, enabling efficient processing in streaming operations. The *remove-columns* operation removes a subset of the columns in a KGTK file (node1 (source), node2 (object), and label (property) cannot be removed). This is useful in cases where columns have lengthy values and are not relevant to the use case pursued by a user, e.g., removing edge and graph identifiers when users aim to compute node centrality or calculate embeddings. The *filter* operation selects edges from a KGTK file, by specifying constraints ("patterns") on the values for node1, label, and node2. The pattern language, inspired by graphy.js, has the following form: "subject-pattern ; predicate-pattern ; object-pattern". For each of the three columns, the filtering pattern can consist of a list of symbols separated using commas. Empty patterns indicate that no filter should be performed for a column. For instance, to select all edges that have property P154 or P279, we can use the pattern " ; P154,P279 ; ". Alternatively, a common query of retrieving edges for all humans from Wikidata corresponds to the filter " ; P31 ; Q5". The *join* operation will join two KGTK files. Inner join, left outer join, right outer join, and full outer join are all supported. When a join takes place, the columns from two files are merged into the set of columns for the output file. By default, KGTK will join based on the node1 column, although it can be configured to join by edge id. KGTK also allows the label and node2 columns to be added to the join. Alternatively, the user may supply a list of join columns for each file giving them full control over the semantics of the result. The *cat* operation concatenates any number of files into a single, KGTK-compliant graph file.
4. KGTK includes **graph analytics** operations to compute graph statistics (e.g., centrality), paths between nodes, various text and graph embeddings, and lexicalization of nodes based on their graph environment. Given a set of nodes N and a set of properties P, the

reachable-nodes operation computes the set of reachable nodes R that contains the nodes that can be reached from a node n ∈ N via paths containing any of the properties in P. This operation can be seen as a (joint) closure computation over one or multiple properties for a predefined set of nodes. A common application of this operation is to compute a closure over the subClassOf property, which benefits downstream tasks such as entity linking or table understanding. The *connected-components* operation finds all connected components (communities) in a graph (e.g., return all the communities connected via an owl:sameAs edge in a KGTK file). The *text-embeddings* operation computes embeddings for all nodes in a graph by computing a sentence embedding over a lexicalization of the neighborhood of each node. The lexicalized sentence is created based on a template. The *graph-statistics* operation computes various graph statistics and centrality metrics. The operation generates a graph summary, containing its number of nodes, edges, and most common relations. In addition, it can compute graph degrees, HITS centrality, and PageRank values. Aggregated statistics (minimum, maximum, average, and top nodes) for these connectivity/centrality metrics are included in the summary, whereas the individual values for each node are represented as edges in the resulting graph.

3.2.3 KGTK pipelines

Each KGTK operation expects KGTK (TSV) files as input and produces KGTK files as output. This enables its commands to be combined into pipelines with a flexible number and order of commands, implemented based on Unix pipes. Pipelining increases efficiency by avoiding the need to write files to disk and supporting parallelism allowing downstream commands to process data before upstream commands complete. Note that we have implemented a shortcut pipe operator "/", which allows users to avoid repeating 'kgtk' in each of their operations. We illustrate the chaining operations in KGTK with the following example (Jupyter Notebooks that implement this, and other examples can be found online):[2]

Example: *Alice wants to import the English subset of ConceptNet [9] in KGTK format to extract a filtered subset where two concepts are connected with a more precise semantic relation such as /r/Causes or /r/UsedFor (as opposed to weaker relations such as /r/RelatedTo). For all nodes in this subset, she wants to compute text embeddings and store them in a file called emb.txt.*

To extract the desired subset, the sequence of KGTK commands is as follows:

```
kgtk import-conceptnet --english_only conceptnet.csv /\
filter -p "; /r/Causes,/r/UsedFor,/r/Synonym,/r/DefinedAs,/r/IsA ;" /\
sort -c 1,2,3 > sorted.tsv
```

To compute embeddings for this subset, she would use text-embeddings:

```
kgtk text-embeddings --label-properties "/r/Synonym" \ --isa-properties "/r/IsA"
--description-properties "/r/DefinedAs" \ --property-value "/r/Causes"
"/r/UsedFor" \ --model bert-large-nli-cls-token -i sorted.tsv \ > emb.txt
```

3.3 KGTK services

KGTK includes a wide range of services for knowledge explorers and developers, such as a customizable web browser, similarity endpoint and graphical user interface (GUI) [10], and faceted search.

[2] https://github.com/usc-isi-i2/kgtk/tree/main/examples

3.3.1 Similarity endpoint

Our similarity GUI allows users to search for a primary Wikidata Qnode based on its labels or aliases. The user could then add any number of secondary Qnodes in the same way, based on free text search against the node labels and aliases. We use ElasticSearch to build a text index and enable this search. The interface then displays the similarity between the primary node and each secondary Qnode, according to each of the supported algorithms. Currently, we support a number of algorithms:

1. Class similarity computes the set of common is-a parents for two nodes. Here, the is-a relations are computed as a transitive closure over both the subclass-of (P279) and the instance-of (P31) relations. Each shared parents is weighted by its inverse document frequency (IDF), computed based on the number of instances that transitively belong to that parent class.

2. TransE [11] similarity computes the cosine similarity between the pre-computed TransE embeddings of two Wikidata nodes.

3. ComplEx [12] similarity computes the cosine similarity between the pre-computed ComplEx embeddings of two Wikidata nodes.

4. Text similarity computes the cosine similarity between the pre-computed RoBERTa-large [13] embeddings of two Wikidata nodes. We pre-compute these RoBERTa embeddings over a lexicalized version of each Wikidata Qnode, based on its outgoing edges in the graph. We use outgoing edges that belong to the following seven relation types: P31 (instance of), P279 (subclass of), P106 (occupation), P39 (position held), P1382 (partially coincident with), P373 (Commons Category), and P452 (industry).

Figure 2. Similarity between motorcycle (Q34493) and ten other terms, i.e., three semantically similar nodes: bus (Q5638), Dirt Bike (Q3050907), and yacht (Q170173); four related, but dissimilar nodes: engine (Q44167), helmet (Q173603), road (Q34442), and cyclist (Q2125610); and three unrelated nodes: cheese (Q10943), Norway (Q20), and shelf (Q2637814). The results are ordered based on their class similarity score.

The similarity score between two nodes is computed on the fly, which allows us to facilitate estimation for any pair of Wikidata nodes. We use the operation graph-embeddings of KGTK to compute TransE and ComplEx embeddings. We use the KGTK text-embeddings command to compute the text (RoBERTa-based) embeddings. A snapshot of the similarity interface is shown in Figure 2.

Besides a GUI, we also have a nearest Neighbors REST API, which returns K nearest neighbors for a Qnode based on the ComplEx graph embedding algorithm. We index the ComplEx embeddings in a FAISS [14] index, which facilitates efficient retrieval. The entire code of the similarity GUI and API is provided on GitHub, including instructions for setting up these services locally for custom graphs.[3]

3.3.2 Browser

KGTK includes an adaptive web browser (Figure 3). The browser can be tried online for the Wikidata KG.[4] The browser is designed and implemented to be fast, scalable, and user friendly.

[3] https://github.com/usc-isi-i2/kgtk-similarity

[4] https://kgtk.isi.edu/browser/Q12379

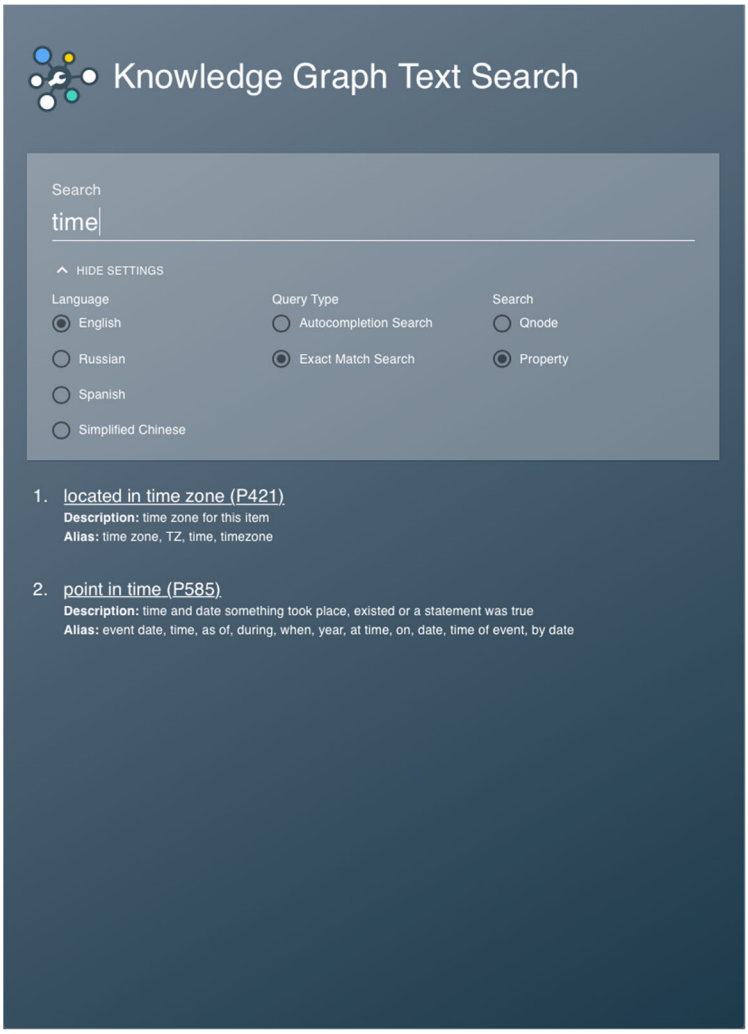

Figure 3. KGTK text search interface example. The user queried the string "time" in English language, looking for exactly matched results of type Property. Then endpoint returns two relevant results.

Its functionality includes incoming links to enable effective finding of information. It includes profiling information on centrality and popularity of nodes. And it includes a visualization of the class hierarchy to help users navigate the complex hierarchy of Wikidata. The code of the browser together with instructions on how to adapt it to novel graphs can be found on GitHub.[5]

3.3.3 Search endpoint

KGTK includes a text search functionality (Figure 4) built on top of ElasticSearch. Like the other services, a version of the search for the Wikidata KG is publicly available.[6] The search endpoint is also available via a programmatic API.[7] The search endpoint enables users to query a KG via natural language, and further constrain the results based on language, semantic types, and meta-

[5] https://github.com/usc-isi-i2/kgtk-browser

[6] https://kgtk.isi.edu/search/

[7] https://kgtk.isi.edu/api?q=apple&extra_info=true&language=en&is_class=false&type=ngram&size=5

type (item, property or class). The user can specify whether additional information, like PageRank and statements, should be returned, and indicate the number of matching items to return. The code of the search API is available on GitHub.[8]

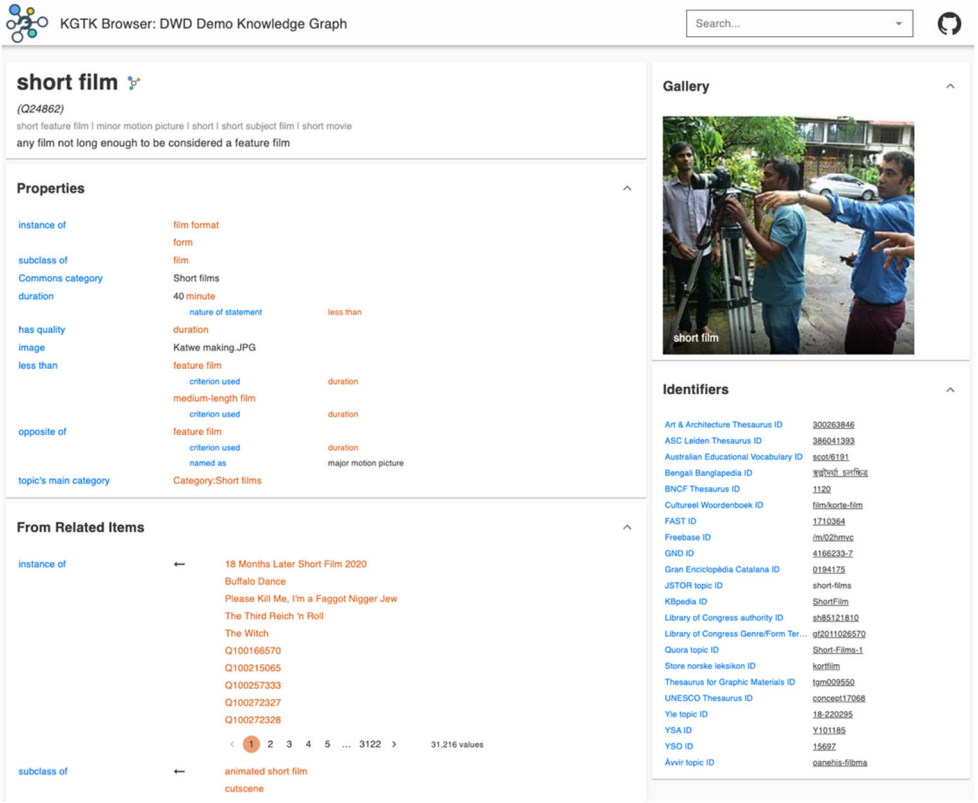

Figure 4. KGTK browser entry for short film (Q24862). Properties and qualifiers are shown in blue, Qnode values in orange, and literals in black. The section 'From Related Items' shows incoming edges. On the right side, there is a representative image and a list of identifiers in other sources.

3.3.4 SPARQL endpoint

Finally, KGTK's service suite includes a SPARQL endpoint (Figure 5) for public structured queries.[9] The SPARQL endpoint is based on Wikibase, analogously to the central Wikidata endpoint. It includes a list of example queries and query building assistant to allow end users to get information out more efficiently.

[8] https://github.com/usc-isi-i2/kgtk-search

[9] https://kgtk.isi.edu/sparql/

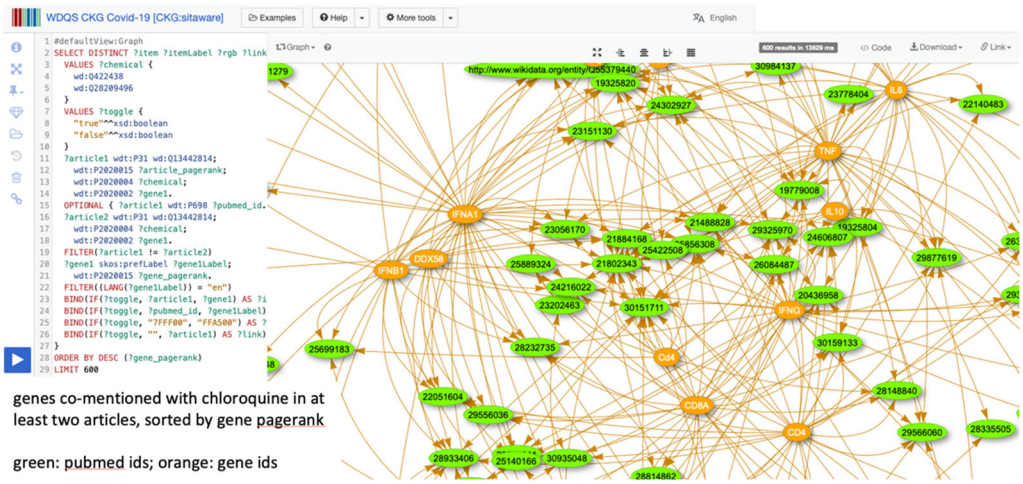

Figure 5. SPARQL query and visualization of the CORD-19 use case, illustrating the use of the Wikidata infrastructure using our KG that includes a subset of Wikidata augmented with new properties such as "mentions gene" and "pagerank".

3.4 KGTK use cases

We have already used KGTK to support a list of diverse use cases. We describe eight of these use cases here, and we provide results for four of them in the next section.

1. Study of the quality of 300+ Wikidata dumps - KGs like Wikidata consist of billions of statements, which may exhibit various challenges of their quality. We leveraged KGTK to study the quality of Wikidata statements based on three indicators: deleted statements, deprecated statements, and violated statements [15]. We performed an in-depth analysis of over 300 dumps of Wikidata that were released since its inception. We provide key insights from this work in the Results and Discussion below.

2. Estimating concept similarity - In one research effort [16], we developed a framework for estimating similarity on top of our similarity GUI. We used graph embedding and text embedding models, as well as ontology-based metrics, as initial similarity estimators. We also concatenated the embeddings in order to combine their scores. We used retrofitting as a knowledge injection mechanism to further tune the individual or the combined embedding models, through distant supervision over millions of weighted pairs extracted automatically from large-scale knowledge graphs. For a given concept pair from Wikidata, the similarity scores generated by the retrofitted embedding models were combined with the scores by the ontology-based models. We provide some of our key results from this research in the next section.

3. Measuring fact surprise - We studied the ability of popular embedding models, like TransE [11], ComplEx [12], and language models like BERT [17], to identify the surprise level of a fact in Wikidata. We formulated Surprising Fact Identification [18], a novel task where a system has to estimate how surprising a Wikidata fact is. We created two benchmarks for this task: Trivia, where systems rank the facts per question according to their surprise, and Survey, where the goal is to provide a global ranking of 70 facts. We proposed two scalable methods based on embedding models with an ability to estimate surprise score for any statement in a graph like Wikidata. We evaluated a list of embedding model variants to perform Surprising Fact Identification on the two benchmarks. Hypothesizing that embedding-based link prediction models regress to the

mean, we also analyzed whether their predictions for a given subject entity tend towards the values that are most commonly correct for similar entities. We provide results that compare these two methods on two benchmarks below.

4. Connecting culture and creative content to understand internet memes - Memes are difficult because they are culture- and context-dependent, creative, and subjective. To understand, for example, what the most common films depicted in memes are, we would need to combine information from meme encyclopedias (like KnowYourMeme), meme generators (e.g., ImgFlip), and sources of background knowledge (like Wikidata). We used KGTK to create a consolidated Internet Meme Knowledge Graph (IMKG) [19], given its ability to connect the dots between internet meme sources like KnowYourMeme and external knowledge graphs, like Wikidata. KGTK also enabled scalable analytics of the resulting IMKG graph through novel entity-centric queries. In the next section, we provide results that show how KGTK helped consolidate this knowledge into a single graph.

5. Analysis of scientific publications and driving scientific innovation - Knowledge workers, like scientists, are often faced with difficult decisions. Who should I invite to speak at my workshop? Who are the rising stars in my field? The recent advent of public large-scale research publications metadata repositories such as OpenAlex [20] enables us to study innovation at scales that have not been possible before. However, dealing with these large-scale repositories is extremely difficult and requires special toolkits. KGTK can be used for data filtering, data transformation, knowledge graph extraction, and knowledge graph embedding training of knowledge graphs with scientific publications.

6. Analytics of financial data and supply chains - Given trading data collected from financial transactions over time, we may want to model in depth the trading patterns, understand the agent motivations, and detect potentially anomalous transactions. Knowledge graph analytics, like centrality metrics and various visualization, could assist with this task. KGTK can be used to analyze financial transaction data. We were able to construct KGTK pipelines with graph transformations, analytics, and visualization steps for the financial sector. The KGTK pipelines enabled us to highlight trading behaviors, find potential colluders, and find inconsistencies through differentiating knowledge graph structures [21].

7. Complex event understanding through moral dimensions - Events happen all the time, and they may be interpreted in different ways by people in accordance with their beliefs and values. Knowledge graphs provide a natural format that can align these perspectives and enable visualization and intuitive access by knowledge explorers and domain experts. We have successfully applied our knowledge graph tool to make sense of complex events. Given a specific domain (or location), we tracked the changes in moral foundations [21] and emotions to understand public perception of these events. The use of KGTK in this project made it easy to scale up, generalize to other domains and locations, and browse and visualize the data.

8. Harmonization and application of commonsense knowledge - KGTK supported the use case of integrating seven heterogeneous sources of common- sense knowledge into a single consolidated graph, the Commonsense Knowledge Graph (CSKG) [22]. Furthermore, KGTK enabled analytics over the resulting graph, making it easier for CSKG to provide relevant information to answer questions that require common sense skills.

4.0 RESULTS AND DISCUSSION

4.1 Importing large files

We designed KGTK to have significant impact within and beyond the Semantic Web community by helping users to easily perform usual data science operations on large KGs. To give an idea about this capability, we downloaded Wikidata (truthy statements distribution, 23.2GB)[10] and performed a test of filtering out all Qnodes (entities) which have the P31 property (instance of) in Wikidata. This filter took over 20 hours in Apache Jena and RDFlib. In graphy, the time went down to 4h 15min. Performing the same operation in KGTK took less than 1h 30min [3].

4.2 Complex queries with Kypher

We performed experiments to compare the execution times of the Kypher and SPARQL implementations of the queries for the following five use cases presented in the original Kypher paper [6]:

1. Retrieval of large amounts of data from Wikidata
2. Analytics on the full Wikidata
3. Extraction of new graphs from Wikidata
4. Queries combining multiple resources
5. Combination of Wikidata and DBpedia

We omitted the Wikibase API from our evaluation, as it does not support the types of queries required by the use cases. We used two configurations for Kypher, MacBook Pro laptops with 16GB memory/256GB SSD disk and 32 GB memory/1TB SSD. We used two configurations for SPARQL, the public Wikidata SPARQL endpoint and a local clone of Wikidata (June 2019) running on a server with 24 Intel Xeon cores and 256 GB of memory and SSD. For the Kypher queries we used the Wikidata February 15, 2021 distribution converted to KGTK format. For the Wikidata clone we used the RDF dump from the June 15, 2019 distribution (we did not load the February 15, 2021 distribution as it takes several days to load, and for the purpose of our experiments the earlier dump is adequate as it is smaller). The Jupyter notebook for the Kypher queries was run twice. The first run of the notebook with an empty SQLite database took 349 minutes in the 32GB laptop, and the second run of the notebook, after the data was loaded and indices were built, took 164 minutes. The difference, 185 minutes includes 98 minutes to load the Wikidata data, 10 minutes to load the DBpedia infobox data, and the rest, 77 minutes is time that Kypher used to build database indices. Table 2 shows the runtimes of the queries presented in the use cases. The times are from the second run of the notebook after the data was loaded in the Kypher SQLite database and indices had been created.

[10] https://dumps.wikimedia.org/wikidatawiki/entities/latest-truthy.nt.bz2

Table 2. Comparison of execution times (minutes) of queries in use cases. (*) submitting over 5, 000 ULAN identifiers produces an error due to the length of the query.

Query	Kypher 16GB	Kypher 32GB	SPARQL local (256GB)	SPARQL public
First names	24.37	8.28	31.05	time out
Class instances	104.97	88.97	>24 hours	time out
Film instances	0.03	0.04	1.91	time out
Author network	61.55	66.39	>24 hours	time out
Cancer network	3.18	2.62	40.19	time out
ULAN identifiers	0.56	0.20	1.08	*
DBpedia spouses	3.92	3.43	n/a	n/a

The main objective of KGTK and Kypher is to democratize the exploitation of Wikidata so that anyone with modest computing resources can take advantage of the vast amounts of knowledge present in Wikidata. Our tools focus on use cases that use large portions of Wikidata to distill new knowledge. The experiments show how expensive queries (e.g., class instances use case) that cannot run in one day on a powerful server, complete in about one hour on a laptop. Analytic queries (first names use case) become possible on a laptop in a few minutes, and distillation of knowledge for analysis (author and cancer network use case) become practical. Kypher enables users to easily combine Wikidata with external sources to extract relevant Wikidata knowledge (ULAN use case), or to enhance Wikidata with knowledge from external sources (DBpedia use case).

Kypher is not meant to address use cases that require the most up-to-date data in Wikidata. KGTK uses the Wikidata JSON dumps published every few days, and the KGTK workflow to process the JSON dump takes one day. The comparison with the Wikidata SPARQL endpoints is preliminary as we have not controlled for caching in the triple store and in the operating system, or performed systematic variations of the complexity of the queries. A more detailed and controlled comparison is reserved for a future paper. Here we speculate on the reasons why Kypher seems to perform significantly better than the Wikidata SPARQL endpoints on the presented use cases:

1. Compact data model: the KGTK data model allows us to translate 1.2B Wikidata statements very directly into 1.2B edges, while the RDF translation requires reification and generates O(10B) triples. KGTK also does not require the use of namespaces which makes data values more compact.

2. Smaller database size: more compact data translates directly into smaller database sizes, for example, 142GB for the Kypher graph cache vs. 718GB for the local Wikidata endpoint. This gives generally better locality for table and index lookups and better caching of data pages.

3. Specialized tables: representing specialized data slices such as P279star in their own graph tables makes their reuse very efficient and their indexes more focused, compact, and cache-friendly.

4. Read-only processing: Kypher does not need to support fine-grained updates of tables and indexes, which need to be supported by the public Wikidata endpoint. This requires additional machinery that slows down performance.

5. Use case selection: triple stores and databases are optimized to support a large number of use cases. Our set of use cases samples a small slice of that space, and performance might be very different for other types of queries.

4.3 Study of Wikidata quality

In [15] we devised a framework for estimating the quality of Wikidata that is based on three indicators: 1) community updates; 2) deprecated statements; and 3) property constraints. We define a community-based indicator of KG quality by considering that the KG statements that have been permanently deleted by the community (i.e., statements deleted at a time point t_i and not restored in time points t_j, $j > i$) are of low quality. Similarly, we consider all statements of Wikidata that are marked as deprecated or that violate one of its 30 semantic constraints to be of low quality.

Table 3. Statistics of the constraints: type (Q21503250), value type (Q21510865), item requires statement (Q21503247), inverse (Q21510855), and symmetric (Q21510862). We show the number of properties with (M)andatory, (N)ormal and (S)uggested constraints, and the corresponding number of statements.

	#properties				#statements	validation time (in sec.)			
constraint type	all	M	N	S	all	min	max	mean	median
type	1,456	165	1,280	11	513,424,170	4.95	5231.15	366.16	174.78
value type	897	106	786	5	182,087,480	11.41	5323.18	352.08	144.15
item requires statement	527	78	418	97	302,642,146	1.89	2199.57	133.51	58.6
inverse	110	6	100	4	9,440,925	8.68	646.22	100.69	54.79
symmetric	38	5	30	3	7,145,197	9.72	527.33	118.44	68.67

The application of these three framework indicators result in: 1) a dataset of 76.5M removed statements, describing 26.2M distinct subjects; 2) a dataset of 10M deprecated statements; and 3) a set of correct statements and constraint violations, according to the constraint types specified in Table 3. This table shows that most of the property constraints have a normal status, and that the median time to validate a property constraint over Wikidata ranges between 55 and 175 seconds for the five constraints. This demonstrates the feasibility of our approach to validate Wikidata constraints at scale. We highlight the main findings of our analysis by shedding light into complex issues related to KG quality, such as node redundancy, naming conventions, taxonomic distinctions, completeness, accuracy of constraints, and type consistency. We also explored whether constraint violations are getting corrected over time, thereby improving the overall quality of Wikidata. Specifically, we studied the following eight research questions:

1. Are entities being deduplicated? Our analysis revealed over 2 million redirected nodes, which affect over 20 million statements (26% of all removed statements). The relatively high number of redirects reflects Wikidata's dynamic nature and the community pursuit for a high-quality, well-integrated graph. It is not known how many duplicate entities currently remain in Wikidata.

2. Can the community distinguish classes from instances? Our analysis of removed statements with object properties revealed nearly half a million cases where one of the taxonomic relations has been changed to the other, which point to the fact that the community struggles to decide whether to use instance-of (P31) or subclass-of (P279) to model inheritance in Wikidata.

3. Are naming conventions needed? Our analysis reveals that the community has already performed millions of updates between semantically (nearly) equivalent forms of literals.

4. Are property types and value types respected? We observe that only a small portion of the mandatory constraints, and a much larger portion of the suggested constraints, violate the set constraints. While the violations are largely concentrated around a small set of properties and could in theory be fixed, it is unclear whether this is desired, as the suggested status implies that they might not need to be strictly enforced. We show statistics of the full violation ratios in Table 3.

5. Can we detect missing triples? As shown in Table 4, the mandatory constraints for these constraint types reveal nearly a thousand violations, which may indicate missing triples. The situation worsens for normal and suggested constraints, whose enforcement would lead to millions of potentially missing triples. While fixing symmetric and inverse constraints is programmatically trivial, it is unclear whether this is always desired, as the constraint violations may be caused by an incorrect original statement rather than a missing one. Table 4 (rows 3-5) illustrates how mandatory IRS and inverse constraints are largely followed (with only 0.02% and 1.9% violations, respectively). As expected, the violation ratios are larger for normal, and largest for suggested constraints, peaking at 8% for the IRS suggested constraints.

Table 4. Correct (constraint-satisfying) and incorrect (constraint-violating) statements for the five constraint types analyzed in this paper: type (Q21503250), value type (Q21510865), item requires statement (Q21503247), inverse (Q21510855), and symmetric (Q21510862). The violation ratio (VR) is the percentage of incorrect statements in the total set of statements in a given category. We separate the statistics among (M)andatory, (N)ormal and (S)uggested constraints.

	mandatory			normal			suggested		
constraint	**correct**	**incorrect**	**VR%**	**correct**	**incorrect**	**VR%**	**correct**	**incorrect**	**VR%**
type	44.99M	37.67k	0.08	464.71M	3.58M	0.76	85.03k	21.65k	20.29
value type	11.44M	5.38k	0.03	169.47M	1.11M	0.65	46.15k	512	1.09
item requires statement	3.98M	767	0.02	272.71M	2.25M	0.82	25.73M	2.24M	8.01
inverse	6.56k	133	1.99	7.13M	0.21M	2.79	2M	95.35k	4.55
symmetric	7.43k	42	0.56	6.23M	78.88k	1.25	0.77M	54.22k	6.55

6. Are constraints correct and complete? As shown in Table 4, the majority of the constraints fit the data, which can be seen as an indicator that the constraints are of good quality. Yet, we note that across all constraint types, a small portion of the constraints yields a large portion of violations. We noted that the violation ratios of properties follow a Zipfian distribution – a few properties have very large violation ratios, and most properties have a low percentage of constraint violations.

7. What statements get deprecated? Among the 10 million statements with deprecated rank in Wikidata, we observe that many belong to the domain of Astronomy. This indicates that the decision between removing and deprecating a statement largely depends on the community and the domain.

8. Are constraint violations getting fixed? Our analysis revealed that Wikidata has millions of deleted statements and constraint violations. Do these two sets overlap? We observe that many of the removed statements violated a constraint, i.e., many of the removals coincide with former violations, thereby improving the quality of Wikidata over time.

4.4 Estimating concept similarity in Wikidata

Table 5. Similarity estimation results. Correlation scores for the raw methods and combinations in [16], for each of the benchmarks: Kendall-Tau (KT), Spearman rank (SR), and Root Mean Square Error (RMSE). Best values per column are marked in bold.

Type	Model	WD-WordSim353				WD-RG65			WD-MC30		
		Cover-age	KT	SR	RMSE	KT	SR	RMSE	KT	SR	RMSE
KGE	TransE	334	0.22	0.305	0.699	0.182	0.301	0.793	0.133	0.244	0.87
	ComplEx	334	0.208	0.294	0.81	0.161	0.274	0.748	0.25	0.426	0.763
	Deepwalk	327	0.281	0.392	0.731	0.238	0.322	0.741	0.291	0.422	0.683
	S-Deepwalk	226	0.042	0.055	0.916	0.03	0.054	1.17	-0.143	-0.201	1.177
LME	Abstract	334	**0.523**	**0.697**	**0.523**	**0.518**	**0.662**	**0.592**	**0.567**	**0.753**	**0.568**
	Lexicalization	334	0.374	0.512	1.031	0.408	0.581	0.734	0.45	0.597	0.799
	Label	334	0.041	0.059	0.732	-0.032	-0.047	0.909	0.167	0.218	0.948
	Label+Desc	334	0.368	0.508	0.646	0.132	0.219	0.897	0.25	0.388	0.937
OT	Jiang Conrath	334	0.28	0.393	0.725	-0.065	-0.095	1.03	0.076	0.1	1.074
	Class	334	0.319	0.441	0.741	0	-0.031	1.054	-0.059	-0.091	1.14
Comb.	Composite-All	334	0.437	0.587	0.707	0.304	0.432	0.882	0.25	0.409	0.97
	Composite-Best	334	0.488	0.654	0.572	0.408	0.516	0.718	0.45	0.585	0.729
	TopSim	334	0.382	0.517	0.703	0.257	0.37	0.660	0.217	0.324	0.764

The results of the similarity framework [16] on three datasets mapped to Wikidata is shown in Table 5. Our evaluation of a variety of knowledge graph embeddings, language model embeddings, ontology metrics, and combinations of models revealed several key insights. First, language models were strong indicators of concept similarity in KGs, however, they were extremely sensitive to the kind of input that they operate on. Therefore, scalable and reliable lexicalization is a key component of language model embedding-based similarity models. Second,

knowledge graph embedding models, which largely transfer knowledge about instances to concepts, were also strong indicators of similarity, but not encoding the literal content was a key limitation for these models. Third, retrofitting was helpful across the board, though its impact is larger for simpler models that originally did not encode structural information. Fourth, we noted that careful selection of knowledge for retrofitting is essential, given the size of the sources of background knowledge and their creation methods. Here, we noted that parent-child relations from Wikidata were most useful for retrofitting, whereas knowledge from ProBase [23] generally hurts performance.

4.5 Measuring surprise of Wikidata facts

Table 6. Surprising Fact Identification results on the Wikidata-MC-Trivia-118 benchmark.

		Qnode facts		Numeric facts		All facts	
Methods		ρ	τ	ρ	τ	ρ	τ
Baselines	**random**	-0.003	-0.002	0.024	0.019	0.003	0.003
	frequency	0.043	0.055	0.134	0.129	0.066	0.074
SOD	**BERT**	0.574	0.502	**0.49**	**0.394**	0.553	**0.475**
	ComplEx	0.556	0.468	0.455	0.382	0.531	0.447
	TransE	**0.638**	**0.526**	0.326	0.228	**0.56**	0.452
	P-ComplEx	0.429	0.381	0.056	0.075	0.335	0.305
	P-Transe	0.421	0.382	0.211	0.137	0.368	0.32
	H-RandomWalk	0.594	0.505	0.401	0.348	0.546	0.466
	A-RandomWalk	0.081	0.088	0.171	0.152	0.103	0.104
	S-RandomWalk	-0.025	-0.067	0.381	0.286	0.076	0.021
LP	**ComplEx**	0.475	0.413	-	-	-	-
	TransE	0.442	0.391	-	-	-	-
	P-ComplEx	0.392	0.346	0.219	0.18	0.348	0.304
	P-Transe	0.246	0.211	0.422	0.376	0.29	0.252
	kNN-P-TransE	0.109	0.099	0.639	0.565	0.242	0.215

The results in Table 6 show that our statistical outlier detection (SOD) methods were able to identify surprising facts better than the baselines and our link prediction methods on the Trivia benchmark [18]. Here, SOD yielded the highest correlation scores with the BERT, TransE, and H embeddings, followed closely by the ComplEx embeddings. The performance of the baselines is low on the Trivia benchmark. On another benchmark, called Survey (which we created in [x]), the frequency baseline performs notably better, giving scores that are competitive with most of our models. We think that this is due to the presence of compound facts in the Survey benchmark, whose combination reveals unlikely facts that align with human judgments of trivia-worthiness or surprise. Yet, several of the SOD methods improve over the frequency baseline on this

benchmark as well, revealing that the subject entity embeddings provide key additional information that is not captured by the predicate-object frequencies.

4.6 Consolidating and analyzing Internet Meme knowledge

An example snippet of an enriched Internet Meme Knowledge Graph (IMKG) for the media frame kym:one- does-not-simply-walk-into-mordor is shown in Figure 6. The figure shows that this enrichment includes key entities such as Lord of the Rings, J.R.R. Tolkien, Mordor, Sauron, and Boromir, extracted from images or the about section in KnowYourMeme. The entities are tightly connected based on numerous Wiki- data links, indicating for instance that Mordor is present in the work Lord of the Rings, created by Tolkien, that Boromir is the enemy of Sauron, and that Sauron is a character in Lord of the Rings. The figure also shows the effect of the data modelling, which allows us to connect similar media frames based on their broader subculture or explicit "rdfs:seeAlso" links in KnowYourMeme.

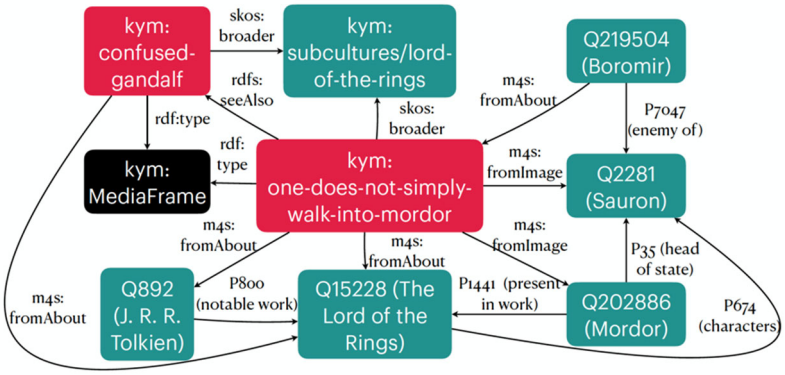

Figure 6. Enrichment example for the meme kym:one-does-not-simply-walk-into-mordor.

Table 7 shows IMKG's general statistics. IMKG has 4.8M nodes described with 16.5M edges. Over a quarter of its nodes are memes, most of which come from ImgFlip. IMKG also has around two thousand templates, i.e., it has on average 2593 IMs per unique template and over twelve thousand frames that are linked to its memes. Most of the edges in IMKG, as can be expected, come from ImgFlip, whereas KnowYourMeme and Wikidata both contribute with hundreds of thousands of edges. The frames in IMKG come from KYM, the memes from practically all sources, and the templates primarily from ImgFlip. As such, IMKG is more than a sum of its parts, as it integrates knowledge in a compatible form.

Table 7. Overall statistics of IMKG and its constituent sources. I2K stands for the collection of links between the ImgFlip templates and the KYM memes.

source	#nodes	#edges	#rels	degree	#frames	#memes	#templates
KYM	167,662	914,941	18	10.91	12,585	12,585	0
ImgFlip	4,698,912	15,129,606	10	6.44	0	1,326,032	1,765
I2K	343	244	1	1.42	96	0	241
WD subset	85,917	504,781	805	11.75	242	242	0
IMKG	**4,850,636**	**16,549,810**	**836**	**6.82**	**12,585**	**1,338,617**	**2,006**

4.7 CONCLUSIONS

Performing common graph operations on large KGs is challenging for data scientists and knowledge engineers. Recognizing this gap, we created the Knowledge Graph ToolKit (KGTK): a data science-centric toolkit to represent, create, transform, enhance, and analyze KGs. KGTK represents graphs in tabular format, and leverages popular libraries developed for data science applications, enabling a wide audience of researchers and developers to easily construct KG pipelines for their applications. KGTK currently supports dozens of common operations, including import/export, filter, join, merge, computation of centrality, and generation of text embeddings. We are using KGTK in our own work for a long list of real-world scenarios which benefit from integration and manipulation of large KGs, such as Wikidata and ConceptNet. Recently, KGTK has been used to support applications by other colleagues in the Semantic Web community. KGTK is open-source software, well documented, actively used and developed, and released using the MIT license. The KGTK tutorial notebooks available and executable online make it easy to get started with minimal investment by trying KGTK in the cloud. KGTK has been fully reproduced by the reviewers of the International Semantic Web conference. Our use cases and tutorial materials demonstrate the diverse set of strengths of KGTK using both popular KGs as well as custom graphs.

We envision that KGTK can serve as a reference infrastructure for knowledge graph engineering. To fulfill this ambitious goal and facilitate adoption, we are working on several complementary directions: improving the ease of importing custom user data into KGTK format, providing informative feedback to end users, better alignment with software engineering best practices, and native interface to the toolkit in Python.

5.0 REFERENCES

[1] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpe- dia: A nucleus for a web of open data. In: The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Lecture Notes in Computer Science, vol. 4825, pp. 722–735. Springer (2007).
[2] Vrandecic, D., Krotzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM 57(10), 78–85 (Sep 2014).
[3] Ilievski, F., Garijo, D., Chalupsky, H., Divvala, N. T., Yao, Y., Rogers, C., ... & Szekely, P. (2020). KGTK: a toolkit for large knowledge graph manipulation and analysis. In The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II 19 (pp. 278-293). Springer International Publishing.
[4] Wang, L.L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R.M., Liu, Z., Merrill, W., Mooney, P., Murdick, D.A., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A.D., Wang, K., Wilhelm, C., Xie, B., Raymond, D.M., Weld, D.S., Etzioni, O., Kohlmeier, S.: CORD-19: The COVID-19 open research dataset. ArXiv abs/2004.10706 (2020).
[5] Gazzotti, R., Michel, F., Gandon, F.: CORD-19 named entities knowledge graph (CORD19-NEKG) (2020), https://github.com/Wimmics/cord19-nekg, Univer- sity Cˆote d'Azur, Inria, CNRS.
[6] Chalupsky, H., Szekely, P., Ilievski, F., Garijo, D., & Shenoy, K. (2021). Creating and querying personalized versions of wikidata on a laptop. Wikidata workshop.
[7] Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., ... & Taylor, A. (2018, May). Cypher: An evolving query language for property graphs. In Proceedings of the 2018 international conference on management of data (pp. 1433-1445).
[8] Chalupsky, H., & Szekely, P. (2022). Hybrid Structured and Similarity Queries over Wikidata plus Embeddings with Kypher-V. Wikidata workshop.
[9] Speer, R., Chin, J., & Havasi, C. (2017, February). Conceptnet 5.5: An open multilingual graph of general knowledge. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).
[10] Ilievski, F., Szekely, P., Satyukov, G., & Singh, A. (2021). User-friendly comparison of similarity algorithms on wikidata. Wikidata workshop.
[11] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. Advances in neural information processing systems, 26.
[12] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016, June). Complex embeddings for simple link prediction. In International conference on machine learning (pp. 2071-2080). PMLR.
[13] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
[14] Johnson, J., Douze, M., J´egou, H.: Billion-scale similarity search with gpus. arXiv preprint arXiv:1702.08734 (2017).
[15] Shenoy, K., Ilievski, F., Garijo, D., Schwabe, D., & Szekely, P. (2022). A study of the quality of Wikidata. Journal of Web Semantics, 72, 100679.
[16] Ilievski, F., Shenoy, K., Chalupsky, H., Klein, N., & Szekely, P. A Study of Concept Similarity in Wikidata.
[17] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

[18] Klein, N., Ilievski, F., Freedman, H., & Szekely, P. (2022). Identifying Surprising Facts in Wikidata. Wikidata workshop.
[19] Tommasini, R., Ilievski, F., Wijesiriwardene, T. (2023). The Internet Meme Knowledge Graph. ESWC 2023.
[20] Priem, J., Piwowar, H., & Orr, R. (2022). OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. arXiv preprint arXiv:2205.01833.
[21] Ilievski, F., Ahrabian, K., Yao, K., Satyukov, G., Pujara, J. (2023). KGTK: User-friendly Toolkit for Manipulation of Large Knowledge Graphs. AAAI Lab.
[22] Ilievski, F., Szekely, P., & Zhang, B. (2021). Cskg: The commonsense knowledge graph. In The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18 (pp. 680-696). Springer International Publishing.
[23] Wu, W., Li, H., Wang, H., & Zhu, K. Q. (2012, May). Probase: A probabilistic taxonomy for text understanding. In Proceedings of the 2012 ACM SIGMOD international conference on management of data (pp. 481-492).

APPENDIX A – PUBLICATIONS AND PRESENTATIONS

6.1 Presentations

- Commonsense Knowledge in Wikidata. **Wikidata-20 workshop, co-located with the In- ternational Semantic Web Conference (ISWC).** Workshop paper presentation. Online. November 2nd, 2021. Speaker: Filip Ilievski.
- KGTK: A Toolkit for High-Volume Knowledge Graph Manipulation. **International Semantic Web Conference (ISWC), Research Track.** Conference paper presentation. Online. November 3rd, 2021. Speaker: Filip Ilievski.
- KGTK: A Knowledge Graph Toolkit for Exploiting Large Knowledge Graphs**. The Web Conference 2021, Developers Track.** Conference paper presentation. Online. April 22nd, 2021. Speaker: Filip Ilievski.
- Knowledge Graph-Based Housing Market Analysis**. Knowledge Graph Construction work- shop, Extended Semantic Web Conference (ESWC) 2021.** Workshop paper presentation. Online. June 6th, 2021. Speaker: Filip Ilievski.
- Open Drug Knowledge Graph. **Knowledge Graph Construction workshop, Extended Semantic Web Conference (ESWC) 2021.** Workshop paper presentation. Online. June 6th, 2021. Speaker: Filip Ilievski.
- CSKG: The CommonSense Knowledge Graph. **Extended Semantic Web Conference (ESWC) 2021.** Conference paper presentation. Online. June 10th, 2021. Speaker: Filip Ilievski.
- User-friendly Comparison of Similarity Algorithms on Wikidata. **Wikidata-21 workshop, International Semantic Web Conference (ISWC).** Workshop paper presentation. Online. October 24th, 2021. Speaker: Filip Ilievski.
- Creating and Querying Personalized Versions of Wikidata on a Laptop. **Wikidata-21 workshop, International Semantic Web Conference (ISWC).** Workshop paper presentation. Online. October 24th, 2021. Speaker: Hans Chalupsky.
- KGTK: Tools for Creating and Exploiting Large Knowledge Graphs. **International Semantic Web Conference (ISWC).** Tutorial. Online. October 24th, 2021. Speakers: Filip Ilievski, Pedro Szekely, Daniel Garijo, Hans Chalupsky.
- Wikidata Birthday: Computed P1963. **WikidataCon.** Talk. Online. October 31st, 2021. Speaker: Pedro Szekely.
- Generating Explainable Abstractions for Wikidata Entities. **K-CAP Conference.** Conference presentation. Online. December 2nd, 2021. Speaker: Nicholas Klein.
- KGTK: Tools for Creating and Exploiting Large Knowledge Graphs. **The ACM Web Conference.** Tutorial. Online. April 25th, 2022. Speakers: Filip Ilievski, Pedro Szekely, Daniel Garijo, Hans Chalupsky, Amandeep Singh, Gleb Satyukov.
- KGTK: Tools for Creating and Exploiting Large Knowledge Graphs. **Knowledge Graph Conference (KGC)**. Tutorial. Online. May 2nd, 2022. Speakers: Filip Ilievski, Pedro Szekely, Daniel Garijo, Hans Chalupsky, Amandeep Singh, Gleb Satyukov.

- CSKG: The CommonSense Knowledge Graph. **Knowledge Graph Conference (KGC).** Conference presentation. Cornell Tech, New York City. May 5th, 2022. Speaker: Filip Ilievski.
- Understanding Narratives through Dimensions of Analogy. **Qualitative reasoning workshop, IJCAI 2022.** Workshop presentation. Vienna, Austria. July 23th, 2022. Speaker: Filip Ilievski.
- Augmenting Knowledge Graphs for Better Link Prediction. **IJCAI 2022**. Conference presentation (oral and poster). Vienna, Austria. July 28th, 2022. Speaker: Filip Ilievski.
- Hybrid Structured and Similarity Queries over Wikidata plus Embeddings with Kypher-V. **Wikidata-22 workshop 2022.** Workshop presentation. Online. October 24th, 2022. Speaker: Hans Chalupsky.
- Tools and User Experience for KG Engineering. Knowledge Graphs and their Role in the Knowledge Engineering of the 21st Century. **Dagstuhl Seminar.** Deep dive presentation. Dagstuhl, Germany. September 13th, 2022. Speaker: Filip Ilievski.
- KGTK: User-friendly Toolkit for Manipulation of Large Knowledge Graphs. **AAAI 2023 Lab.** Washington DC, USA. February 7th, 2023. Speakers: Filip Ilievski, Jay Pujara, Kian Ahrabian, Ke-thia Yao, and Gleb Satyukov.

6.2 Publications

- **KGTK: a toolkit for large knowledge graph manipulation and analysis.** Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Rogers, Rongpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe, Pedro Szekely. International Semantic Web Conference 2020
- **Commonsense knowledge in Wikidata.** Filip Ilievski, Pedro Szekely, Daniel Schwabe. ISWC Wikidata workshop 2020
- **CSKG: The CommonSense Knowledge Graph**. Filip Ilievski, Pedro Szekely, Bin Zhang. Extended Semantic Web Conference (ESWC) 2021
- **Open Drug Knowledge Graph**. Mark Mann, Filip Ilievski, Mohammad Rostami, Aastha Aastha, Basel Shbita. KG Construction workshop 2021
- **Knowledge graph-based housing market analysis**. Ziping Hu, Zepei Zhao, Mohammad Rostami, Filip Ilievski, Basel Shbita. KG Construction workshop 2021
- **A Study of the Quality of Wikidata**. Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, Pedro Szekely. Published in Journal of Web Semantics; 2021
- **User-friendly Comparison of Similarity Algorithms on Wikidata**. Filip Ilievski, Pedro Szekely, Gleb Satyukov, Amandeep Singh. Wikidata-21 workshop 2021
- **Creating and Querying Personalized Versions of Wikidata on a Laptop**. Hans Chalupsky, Pedro Szekely, Filip Ilievski, Daniel Garijo, Kartik Shenoy. Wikidata-21 workshop 2021
- **Generating Explainable Abstractions for Wikidata Entities**. Nicholas Klein, Filip Ilievski, Pedro Szekely. Proceedings of the 11th on Knowledge Capture Conference 2021

- **Analyzing Race and Citizenship Bias in Wikidata**. Zaina Shaik, Filip Ilievski, Fred Morstatter. 2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS) 2021
- **Augmenting Knowledge Graphs for Better Link Prediction**. Jiang Wang, Filip Ilievski, Pedro Szekely, Ke-Thia Yao. Published in IJCAI-ECAI; 2022
- **Enriching Wikidata with Linked Open Data**. Bohui Zhang, Filip Ilievski, Pedro Szekely. Wikidata-22 workshop 2022.
- **Accepted Tutorials at The Web Conference 2022**. Riccardo Tommasini, Senjuti Basu Roy, Xuan Wang, Hongwei Wang, Heng Ji, Jiawei Han, Preslav Nakov, Giovanni Da San Martino, Firoj Alam, Markus Schedl, others. Companion Proceedings of the Web Conference 2022
- **Does Wikidata Support Analogical Reasoning?** Filip Ilievski, Jay Pujara, Kartik Shenoy. Article published in KGSWC; 2022
- **Hybrid Structured and Similarity Queries over Wikidata plus Embeddings with Kypher-V.** Hans Chalupsky, Pedro Szekely. Wikidata-22 workshop 2022.
- **Identifying Surprising Facts in Wikidata**. Nicholas Klein, Filip Ilievski, Hayden Freedman, Pedro Szekely. Wikidata-22 workshop 2022
- **The Internet Meme Knowledge Graph.** Riccardo Tommasini, Filip Ilievski, Thilini Wijesiriwardene. ESWC 2023
- (under review) **A Study of Concept Similarity in Wikidata.** Filip Ilievski, Kartik Shenoy, Hans Chalupsky, Nicholas Klein, Pedro Szekely. Semantic Web Journal.
- (under review) **Comparison of Knowledge Graph Representations for User Consumption Scenarios.** Ana Iglesias-Molina, Kian Ahrabian, Filip Ilievski, Jay Pujara, Oscar Corcho. ISWC.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AAAI	Association for the Advancement of Artificial Intelligence
AI	Artificial Intelligence
API	Application Programming Interface
CSKG	CommonSense Knowledge Graph
CSV	Comma Separated Values
CTD	Comparative Toxicogenomics Database
DOD	Department of Defense
DPR	Dense Passage Retrieval
ESWC	Extended Semantic Web Conference
GUI	Graphic User Interface
HITS	Hyperlink-Induced Topic Search
IDF	Inverse Document Frequency
IJCAI	International Joint Conference on Artificial Intelligence
IMKG	Internet Meme Knowledge Graph
JSON	JavaScript Object Notation
KG	Knowledge Graph
KGTK	Knowledge Graph ToolKit
LME	Language Model Embedding
LP	Link Prediction
RDF	Resource Description Framework
REST	Representational State Transfer
SPARQL	SPARQL Protocol and RDF Query Language
SOD	Statistical Outlier Detection
UIUC	University of Illinois Urbana-Champaign
ULAN	Union List of Artist Names