Demonstration Paper

BOLD: Knowledge Graph Exploration and Analysis Platform

Egor Dmitriev
Utrecht University
Utrecht, Netherlands
egordmitriev2@gmail.com

Melisachew Wudage Chekol
Utrecht University
Utrecht, Netherlands
m.w.chekol@uu.nl

Mirko Tobias Schaefer
Utrecht University
Utrecht, Netherlands
m.t.schaefer@uu.nl

ABSTRACT

The linked open data (LOD) cloud maintains several interlinked knowledge graphs. These graphs span various domains such as government, media, life sciences, etc. The graphs are often manually curated or automatically extracted (e.g. YAGO—Yet Another Great Ontology) using information extraction techniques. They are used in various applications such as data governance, fraud detection, fact checking, etc. Although the graphs in LOD are widely used, they do not contain metadata about their representativeness (distribution of key features). Since most of the graphs are automatically curated, bias can manifest due to sensitive features and their causal influences, or through under (over)-representation of certain entities (e.g. people) and relations (e.g. president-of, works-for). The aim of this work is to develop a system to automatically generate bias profiles (metadata about the representativeness of data) for knowledge graphs. As a result, the metadata can be used as a guide for users to choose bias free (balanced) datasets for their studies. Moreover, it enables researchers to quickly gauge the relevance of a graph for a problem at hand (e.g. classification task).

1 INTRODUCTION

In this work, we develop a tool to automatically detect the different kinds of bias that may exist in knowledge graphs. To motivate our work, we performed an explorative and qualitative data analysis on the Wikidata [11] knowledge graph in order to understand where, and in which ways bias is expressed. In other words, we want to investigate the types of bias and their causes. The types of bias are uncovered through the descriptive statistics about the data present in the knowledge graph, the causes are mostly due to practises of knowledge curation in Wikidata.

In a preliminary investigation we focus on the class human, we investigated all entities within Wikidata which are instances of human. We also investigate which relationships are present and which objects are associated with these entities. Our results exemplify the skewedness of Wikidata: there are three times as many men as there are women, the most common occupation is researcher, the second-most common place of death is the concentration camp Theresienstadt, etc. This skewedness result comes from the different forms of bias. The over-representation of researchers, western people and men is grounded in the cultural context of the data authors (e.g. [2, 6]). The place of death refers to another source of bias, i.e., the availability of information. The descriptive statistics gives insight into which forms of bias are present, and their degree. Our preliminary results, therefore, underline the importance of further investigation and necessitate a tool that allows (lay) users to do this.

In addition to representing skewed data, knowledge graphs (KG) may contain incorrect information either due to an error in the KG construction or intentionally supplied by content curators. Besides, KGs can also be incomplete. As an example, in Freebase [1], over 70% of person entities have no known place of birth and over 75% have no known nationality [4]. In Wikidata [11], we observe a similar behavior, for instance, over 97% of humans have no known religion and over 83% of humans have no known spoken, written or signed languages. Subsets of both Wikidata and Freebase have been widely used for testing knowledge graph completion models. However, these subsets do not take into account the incompleteness of the KGs and are prepared in a way to test solely the accuracy of models. However, if the subsets are incomplete (or unbalanced), the models can be biased [10]. For instance, the Wikidata12K [7] dataset contains 80% male and 20% female politicians. Clearly, this dataset is unbalanced and a model trained on it will likely over-represent men in its predictions. The observations described above motivated our work to create BOLD (Bias in Open Linked Data), a system that allows (lay) users to profile knowledge graphs. Specifically, BOLD enables users to accomplish the following tasks:

- Seamless import of knowledge graphs from the linked open data cloud (LODC)[1] and Triply DB[2],
- Interact with external SPARQL endpoints,
- Create persistent reports and share them with others,
- Run SPARQL or pre-built analysis queries,
- Explore KGs with interactive visualizations, and
- Select entities with fuzzy search.

During the conference, participants or users have the opportunity to engage with the BOLD system in a multitude of ways, ensuring a rich and interactive experience. Users can choose to either deploy BOLD on their individual machines, providing a personalized and hands-on exploration, or utilize our centralized setup for comprehensive testing. Users can seamlessly, upload or import existing knowledge graphs, create SPARQL queries, harness BOLD's widgets to generate different distributions of entities or visualize a subset of a given dataset. Alternatively, users can also interact and explore with our ready-made reports. Instructions, along with handy animations (in the form of GIF files), on how to use BOLD are provide at the address: https://egordmitriev.net/BOLD/user_manual/. An instance of BOLD which runs in a web browser can be found at https://egordmitriev.net/BOLDER/.

2 THE BOLD SYSTEM

The BOLD platform depends on StarDog[3] and PostgreSQL databases for knowledge graph and state storage.

2.1 Architecture

The BOLD architecture is designed to be scalable and to handle datasets of all sizes. Therefore we keep the data and the code

[1] https://lod-cloud.net/
[2] https://triply.cc/triplydb/
[3] https://www.stardog.com/

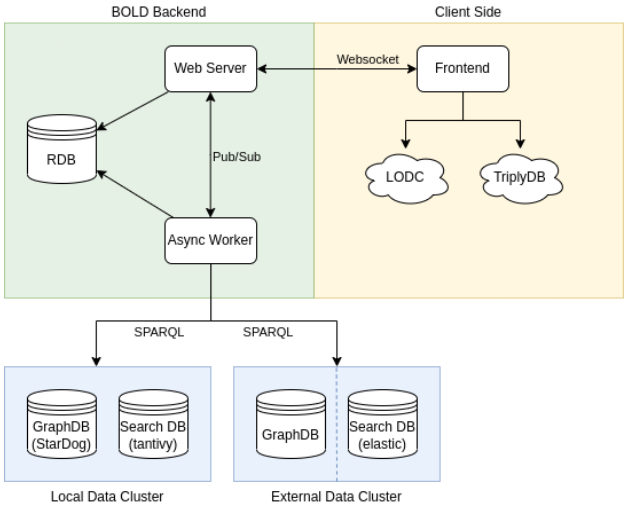

Figure 1: Architecture of the BOLD system.

separate. Additionally, we define asynchronous execution architecture to allow for parallel execution of queries, dataset imports and fault tolerance. A high level overview of the architecture is given in Figure 1, where we can see the described separation. It is also important to make the analysis and visualization extensible so that new features can be added as required. In order to achieve this, we separate the analysis and visualization from the execution logic by strictly assigning these roles to the frontend and backend.

The backend is responsible for the execution of queries and the data management. It receives an execution plan in the form of a report, with report cells containing SPARQL queries, and executes them in parallel for a given dataset. The client side allows to import/upload datasets (locally, from LODC or TriplyDB). The import component does not work with third party services, but with RDF data imports/uploads or SPARQL endpoints directly. The data is either stored in a self-hosted *local data cluster* (using StarDog triple store) or is accessed in a read-only fashion from an *external data cluster* using SPARQL and Elasticsearch endpoints. Moreover, the state of the workers is decoupled from the backend server and vice-versa for fault tolerance.

Indexing. When a dataset is uploaded, English resource labels and descriptions (if available) are used to create the full-text search index in Tantivy, which abstracts most of the index creation process. In addition to labels, we also tag their usage (whether it is a subject, predicate, or object) and count their occurrences to enhance search result quality.

2.2 User Interface

When the user logins into (or runs) the system, they see the page shown in Figure 2. This is the main page where the user has the option to create a dataset, interact with reports, see running tasks or select existing or import datasets.

2.2.1 Datasets. A BOLD dataset represents an (imported) RDF dataset. The dataset consists of a SPARQL endpoint and a search index. Both can point to either a local or a remote resource. The datasets can be shared between users and reports. There are two main ways to import datasets into BOLD: direct import/upload and via a SPARQL endpoint.

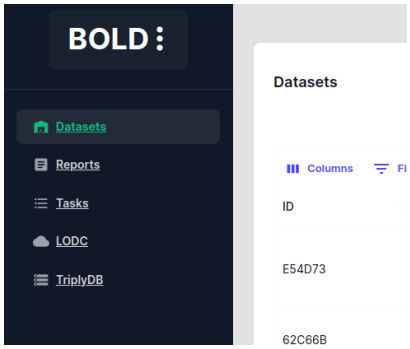

Figure 2: BOLD's main page where 5 menu items are shown.

- Direct import - it is possible to import, an RDF dataset into BOLD, and create a search index. This requires the machine, where BOLD is running, to have sufficient memory and storage to store the dataset.
- SPARQL endpoint - it is possible to create a dataset from a SPARQL endpoint and use an external search index. This does not require the machine (where BOLD is running) to have enough storage to store the dataset as it is accessed directly from the endpoint.

2.2.2 Reports. A report cell contains either SPARQL queries or a configurable widget which generates queries for the database. The widgets can be configured to generate different plots. There are 8 different widgets that allow to build histograms, pie charts, retrieve subgraphs, browse taxonomies, and so on. As an example, Figure 3(A) shows a widget for building a histogram about the sex or gender distribution of American scientists, on the Wikidata dataset. When the widget is executed, it produces the histogram shown in sub-figure (C). The plot shows that the majority of entities in Wikidata that are scientists have the sex/gender male compared to females. For the other three categories, namely, non-binary, trans woman and trans man, there is only a single entity. In BOLD, it is also possible to input SPARQL queries in a report cell as shown in Figure 3(B).

In Figure 4, we demonstrate the process of creating a BOLD report. When the user navigates to the reports page, they encounter image (1). Here, users can choose from eight different widgets using the "cell type" dropdown menu (image 2). If the user selects "code," a cell with a pre-filled SPARQL query that can be edited appears. Choosing a plot builder widget reveals image (3). Using the plot builder, users can specify the entities and relations for plotting. For example, in image (4), the user selects Pokemon character types based on length. Image (5) displays the user specifying the variables for plotting (type and length). Upon executing the widget (image 6), a pie chart plot is generated (image 7). Users have the option to transform the pie chart into a bar plot (images 8 and 9).

2.2.3 Tasks. A BOLD task represents a collection of works that can be scheduled and assigned to a worker. Tasks are meant to be used for long-running tasks and run in parallel to avoid blocking the main server.

2.3 Deployment

It is possible to set up a BOLD instance using Docker. In order to do so, one needs to create a Docker configuration file specifying

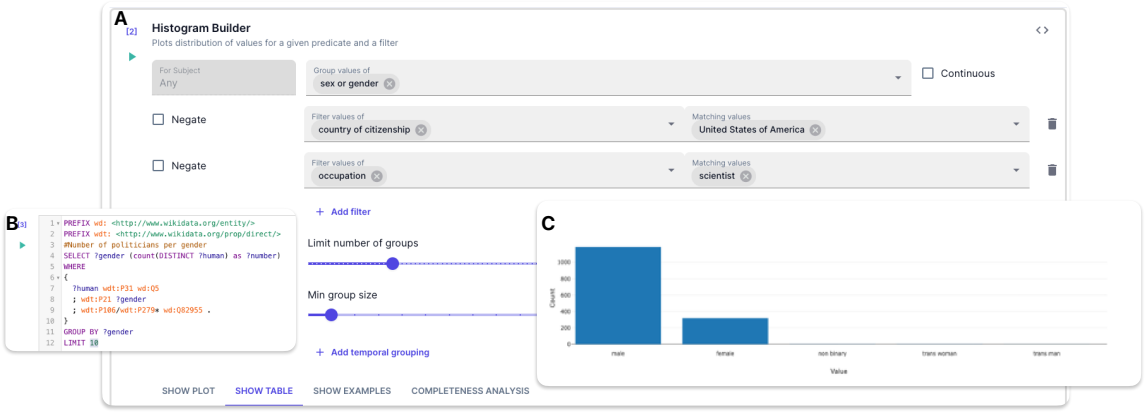

Figure 3: A) Widget that is used to generate the barplot in (C) when run on Wikidata. B) A query to accomplish the same.

Table 1: The table illustrates the system memory requirement for BOLD per dataset size.

Number of triples	Total system memory (in GB)
100 million	8
1 billion	32
10 billion	128
25 billion	256
50 billion	512

the triple store (StarDog, BlazeGraph[4], etc) and search (Tantivy, Elasticsearch, etc) database to use. Docker uses the configuration file to execute the BOLD instance, starting the container that runs on localhost with the default port set to 8000. For additional information on setting up BOLD locally, we refer the reader to the user manual which can be found at https://github.com/UtrechtUniversity/BOLD-KB-Profiler. Alternatively, users can also interact with the BOLD system through the streamlined web interface[5], which runs on top of the Wikidata knowledge graph.

In order to install BOLD, it is recommended to have twice the amount of storage the datasets require, for instance, for the YAGO [8] dataset (which is ~60GB), one needs 120GB. In addition, at least 2 CPU cores, on the machine where BOLD runs, must be allocated. These system requirements are mostly bound by the StarDog database. While using StarDog is not mandatory, if chosen, it is not possible to import full datasets; only external SPARQL endpoints are allowed. Alternatively, (expert) users can configure BOLD to work with other triple stores such as BlazeGraph or Virtuoso [5]. The memory requirements of BOLD are tied to the size of the datasets. As shown in Table 1, a dataset that has 50 billion triples requires at least 512GB of memory.

3 RELATED WORK

Demartini [3] proposes methods to trace the provenance of crowd sourced fact checking to enable bias transparency rather than aiming at eliminating bias from a KG. Furthermore, they investigate how paid crowd sourcing can be used to understand contributors' implicit bias. Specifically, they recruit click workers to verify controversial facts and study the process as they do so. They track what search engines are used and which position the URL used to validate was ranked in the result page. An example verification task is the question of whether Catalonia is a part of Spain or an independent country. The paper proposes adding both facts to the knowledge graph, with a statement testifying how much support there is for each fact.

Wisesa et al. [12] introduces ProWD, a framework and tool for profiling the completeness of Wikidata. Completeness measure is based on Class-Facet-Attribute (CFA) profiles. For example one could compare how often the attribute "educated at" or"date of birth" compare between male, German computer scientists, and female, Indonesian computer scientists.

Another tool used for analyzing the gender gap in Wikidata is Denelezh[6]. It provides an overview of gender distribution among all human entities that have different language labels. Across all editions, only in the occupational category of actors, females are present with more than 40% proportion. In addtion, the Wikidata Concepts Monitor (WDCM[7]) tool provides an overview of the different entities in Wikidata and which applications are using them. Both tools have been used to evaluate how each of them represents gender bias in Wikipedia. Denelezh can combine various dimensions, such as occupation and country of citizenship, to calculate gender bias in different categories of profession.

Although RDF-ANALYTICS [9] was not originally developed for bias analysis. The authors claim that, it allows to explore and create analytical queries on knowledge graphs. The tool allows lay users to create queries without having to know dataset vocabulary or the SPARQL query language, showcasing a flexible and accessible analytical framework.

In contrast to BOLD, the aforementioned tools are tailored specifically for Wikidata. They have limitations in terms of the attributes that they can analyze; for instance, Humaniki[8] and Denelezh are primarily focused on addressing gender bias. Furthermore, their analysis is confined in scope; for instance, ProWD analyzes representativeness based on completeness, it does not delve into analyzing the underlying values.

[4]https://blazegraph.com/
[5]https://egordmitriev.net/BOLDER/

[6]https://denelezh.wmcloud.org/documentation/
[7]https://www.wikidata.org/wiki/Wikidata:Wikidata_Concepts_Monitor
[8]https://humaniki.wmcloud.org/search

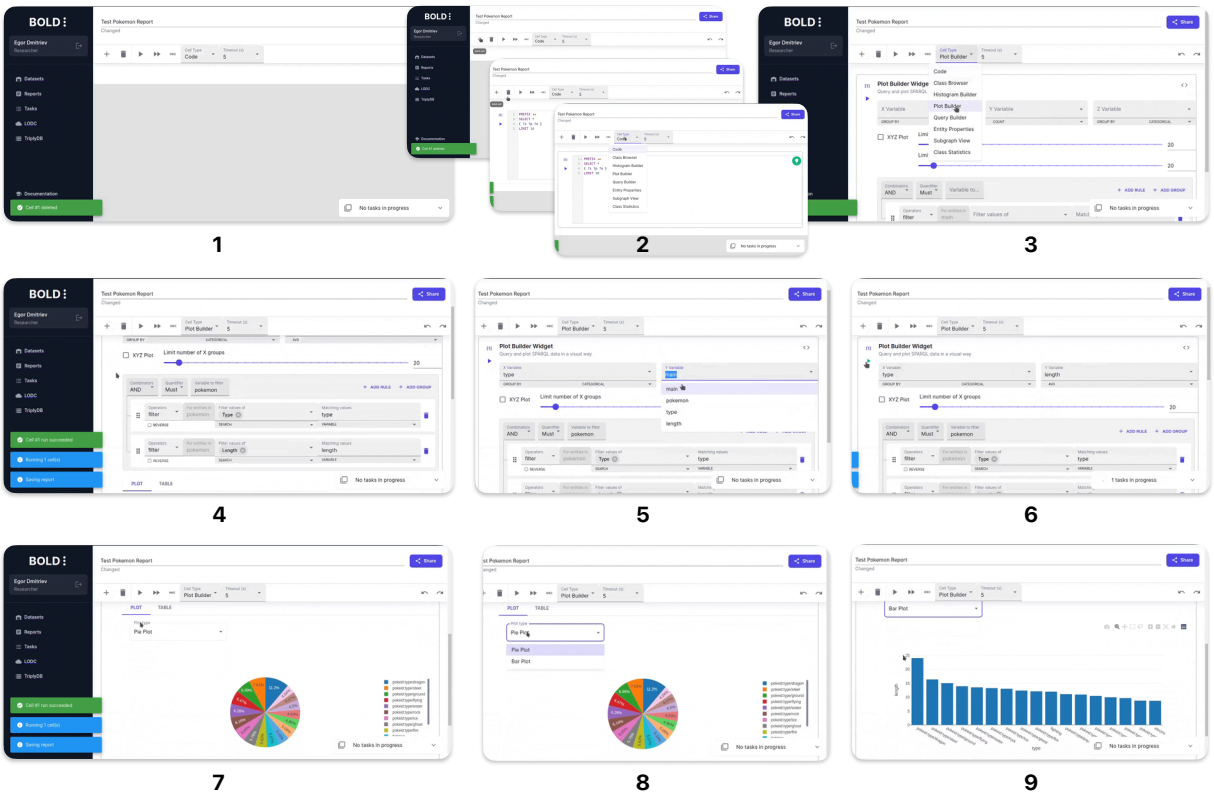

Figure 4: Steps to create a pie chart or bar plot showing the distribution of Pokemon character types based on their length.

These initiatives, among others, indicate that there is a growing interest to identify bias in knowledge graphs. Given that knowledge graphs are often collaborative repositories, it is important to provide users with accessible tools to identify potential bias. While the mentioned examples are beneficial, they have limitations in that they are either tailored to a specific knowledge graph or focus on a restricted set of attributes. A general framework/system (like BOLD) might provide more possibilities to map bias in knowledge graphs and enable users to become aware of the distribution of entities and attributes in a given knowledge graph. It also allows for a wider range of users; while expert users might utilize it for complex queries, lay users could benefit from reports generated by expert users. With their own subject specific expertise, these users can then decide which bias is problematic, and how to address it.

ACKNOWLEDGMENTS

This work was supported by SIDN Fonds (www.sidnfonds.nl/).

REFERENCES

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

[2] Ewa S Callahan and Susan C Herring. 2011. Cultural bias in Wikipedia content on famous persons. *Journal of the American society for information science and technology* 62, 10 (2011), 1899–1915.

[3] Gianluca Demartini. 2019. Implicit Bias in Crowdsourced Knowledge Graphs. In *Companion Proceedings of The 2019 World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 624–630. https://doi.org/10.1145/3308560.3317307

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

[5] Orri Erling and Ivan Mikhailov. 2009. Virtuoso: RDF support in a native RDBMS. In *Semantic web information management: a model-based perspective*. Springer, 501–519.

[6] Eduardo Graells-Garrido, Mounia Lalmas, and Filippo Menczer. 2015. First women, second sex: Gender bias in Wikipedia. In *Proceedings of the 26th ACM conference on hypertext & social media*. 165–174.

[7] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*. 1771–1776.

[8] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org.

[9] Maria-Evangelia Papadaki and Yannis Tzitzikas. 2023. RDF-Analytics: Interactive Analytics over RDF Knowledge Graphs. In *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*. OpenProceedings.org, 807–810.

[10] Wessel Radstok, Melisachew Wudage Chekol, and Mirko T. Schäfer. 2021. Are Knowledge Graph Embedding Models Biased, or Is it the Data That They Are Trained on?. In *Proceedings of the 2nd Wikidata Workshop (Wikidata 2021) co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24, 2021 (CEUR Workshop Proceedings)*, Vol. 2982. CEUR-WS.org.

[11] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[12] Avicenna Wisesa, Fariz Darari, Adila Krisnadhi, Werner Nutt, and Simon Razniewski. 2019. Wikidata Completeness Profiling Using ProWD. In *Proceedings of the 10th International Conference on Knowledge Capture (K-CAP '19)*. Association for Computing Machinery, New York, NY, USA, 123–130.