VQFT: A Visual Query Approach Based on Full-Text Search for Knowledge Graphs

Zhaozhuo Li
Tianjin University
Tianjin, China
lizhaozhuo@tju.edu.cn

Xin Wang
Tianjin University
Tianjin, China
wangx@tju.edu.cn

Meng Wang
Tongji University
Shanghai, China
mengwangtj@tongji.edu.cn

Yajun Yang
Tianjin University
Tianjin, China
yjyang@tju.edu.cn

Bohan Li
Nanjing University of Aeronautics and Astronautics
Nanjing, China
bhli@nuaa.edu.cn

Dong Han
Tianjin Academy of Fine Arts
Tianjin, China
handong@tjarts.edu.cn

ABSTRACT

Existing knowledge graph query approaches, whether traditional textual query languages or visual query languages, have steep learning curves that are unfriendly for non-expert users. This demonstration presents a Visual Query approach based on Full-Text search for knowledge graphs, called VQFT, which simplifies the process of querying knowledge graphs for users. Inspired by *full-text search* techniques, VQFT aims to combine the user-friendliness of *visual query* with the intuitiveness of *full-text search*, enabling users to query knowledge graphs as straightforward as using a search engine. *Faceted full-text indexes*, *visual query constructor*, and an *interactive user interface* are designed to achieve this goal. User tests and surveys have demonstrated that VQFT is more user-friendly and easier to learn than existing methods, which simplifies the construction of knowledge graph queries for non-expert users.

PVLDB Reference Format:
Zhaozhuo Li, Xin Wang, Meng Wang, Yajun Yang, Bohan Li, and Dong Han. VQFT: A Visual Query Approach Based on Full-Text Search for Knowledge Graphs. PVLDB, 17(12): 4397 - 4400, 2024.
doi:10.14778/3685800.3685884

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Lzz1027/VQFT

1 INTRODUCTION

Complex KG queries: barrier for non-expert users. Knowledge graphs (KGs) represent real-world entities and their relationships in a semi-structured graph form. Large-scale and cross-domain KGs constructed by leading companies and academic institutions have become valuable data resources, attracting many non-expert users who are not familiar with KGs. SPARQL [5] and Cypher [2] are two mainstream textual query methods for retrieving information from KGs. However, mastering these complex query languages requires a significant amount of time and effort. Meanwhile, it is difficult for large language models, such as ChatGPT and Claude, to understand the non-textual topological structures and schema semantics in KGs. These models face difficulties in constructing accurate structured queries even with sufficient example prompts. Consequently, non-expert users face significant challenges when searching KGs.

Visual query has limited effect. Visual querying is a strategy for reducing the complexity of text-based queries. Visual query languages, such as ViziQuer [1], QueryVOWL [3], and KGVQL [4], attempt to increase the user-friendliness of graph queries by mapping syntactic symbols to visual elements. The recent KGNav [6] method proposes a navigational-based approach for visually browsing KGs. However, since the essence of existing methods is to map symbols in textual syntax onto visual elements, the complexity of constructing queries has not been actually reduced, thus the learning curve for non-expert users remains steep. Therefore, while current visual query methods have made progress in terms of usability, it is still difficult for non-expert users to query KGs.

Full-Text search can help. Full-Text search, a well-established method, is the core approach used by search engines to retrieve unstructured data. It builds an inverted index, which allows users to quickly search documents using keywords. As this method matches users' intuitive search behavior, it is highly suitable for non-expert users. Given the semi-structured and weak schema nature of KGs, constructing graph queries is inherently difficult. Full-Text search, however, requires users to enter only keywords to easily query entities, attributes, and their relationships. For users, this approach eliminates the need to understand complex query languages or KG schemas, making it easier for non-experts to learn and use.

Our approach. As it is infeasible to build KG queries with a single search box as in search engines, we propose VQFT, which combines the ease of use of full-text search with the intuitiveness of visual query to satisfy users' KG query needs. We design visual query elements that are as simple as possible and combine multiple full-text searches into faceted queries. Our approach covers the core functionality of graph queries, including basic graph schema matching, entity/relation attribute processing, and complex graph schema querying, where hints and suggestions can be used to respectively reduce difficulty and improve the quality of queries. Our approach has no restrictions on schemas or domains of KGs and

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685884

can significantly reduce the complexity for non-expert users in constructing KG queries.

Demonstration. Conference participants will have the opportunities to experience the user-friendly interface of VQFT and build visual queries over the real-world data from Wikidata[1]. Participants can access our system and interact with the demonstration through our website at http://www.tjudb.cn/vqft/.

2 VQFT OVERVIEW

Figure 1 illustrates the architecture and the four main components of VQFT, i.e., KGFI, VQC, GUI, and VQR, which will be introduced in this section one by one.

2.1 Knowledge Graph Full-Text Indexes (KGFI)

Due to its semi-structured and weak schema nature, a KG can be considered as a collection of unstructured data entities. Based on this idea, we provide a formal conceptual framework of KGFI.

A knowledge graph KG can be represented as $KG = \{E, R, T\}$, where E is the set of entities (or nodes), R is the set of relationships (or edges), and T represents the set of textual descriptions which are related to the elements in E or R. Each entity $e \in E$ or relation $r \in R$ in the KG can be associated with its corresponding sets T_e and T_r, which include names or other attributes. This forms a KG with semi-structure, which can be viewed as a collection of unstructured data, $U = (\cup_{e \in E} T_e) \cup (\cup_{r \in R} T_r)$. Each piece of unstructured data is an aggregation of textual descriptions for entities and relationships in the KG. Consequently, all these unstructured data collectively form the knowledge graph.

An inverted index I for a KG full-text search can be represented as $I : t \rightarrow \{e \mid t \in T_e\} \cup \{r \mid t \in T_r\}$, which maps terms t found within the textual descriptions T to entities e or relationships r. Term t is obtained by a disambiguation process on the textual description T, which accurately corresponds to the content in T.

Faceted full-text search F is defined as $F = \{F_1, F_2, \dots, F_m\}$, where each facet F_i represents a specific dimension (e.g., entity type, relationship type, or entity attributes) within KG. Thus, a faceted full-text search query Q on facet F_i for term t can be expressed as: $Q(F_i, t) = \{x \mid x \in E \cup R,\ t \in T_x,\ x \text{ satisfies } F_i\}$. Where, x denotes an entity $e \in E$ or a relationship $r \in R$, and "x satisfies F_i" means that x conforms to or is part of the specific dimension or category defined by facet F_i.

2.2 Visual Query Constructor (VQC)

For users, the most intuitive form of a KG is a graph structure consisting of nodes and edges. Therefore, we define the fundamental elements of visual queries as nodes and edges, representing entities and relationships in KGs, respectively. A set of logical operators is also provided to construct complex graph pattern queries.

Nodes: Let $N = \{n_1, n_2, \dots, n_k\}$ represent the set of nodes in a KG G., where each node n_i corresponds to an entity in G.

Edges: Let $E = \{e_1, e_2, \dots, e_l\}$ represent the set of edges in G, where each edge e_j corresponds to a relationship between entities in G. An edge can be formally defined as a tuple $e_j = (n_a, n_b)$, indicating a relationship from node n_a to node n_b.

[1]https://www.wikidata.org

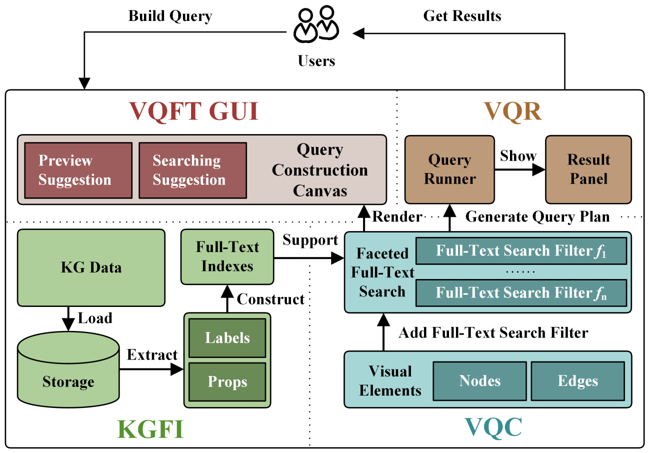

Figure 1: The architecture of VQFT

Operators: The operators for logical operations in visual queries are defined as a set $O = \{AND, OR, NOT\}$, where

- AND indicates a logical conjunction, requiring that all the relationships with the operator must be satisfied.
- OR indicates a logical disjunction, requiring that at least one of the relationships with the operator needs to be satisfied.
- NOT indicates a logical negation, the relationship with the operator is not satisfied.

In summary, all these elements are constructed by VQC, which combines and processes the operators, and finally renders visual elements on the GUI for user interaction.

2.3 VQFT Graphical User Interface (GUI)

Given its direct interaction with users, the VQFT GUI is critical to the effectiveness of the entire method. As shown in Figure 2, the core of the GUI is the *I. Query Construction Canvas*, with secondary focus areas including the *II. Preview Panel* on the left, the *III. Toolbar* at the top, and the *IV. Result Panel* at the bottom.

As we can see, users construct queries on the query construction canvas through the GUI using VQC. To ease the learning process, the GUI provides only two types of nodes: *Result Nodes*, which represent the final query targets, and *Auxiliary Nodes*, which are used to assist in the query construction process. In addition, handles provided on both sides of nodes allow users to draw connections, thereby forming relationships between two entities.

In addition to the visualization elements with graph structure, the core search interaction is accomplished with full-text search filters both on nodes and edges. All these filters combine together and form an intuitive and user-friendly faceted search. As the query is entered, KGFI provides real-time input suggestions. The addition of filters alters the badge number in the top right corner of a node, directly reflecting the effect of filtering. A real-time preview of filtering results for current node is displayed in the right preview panel, allowing users to refine their queries based on preview information. All these interactive behaviors enhance the effectiveness of user queries and reduce the likelihood of constructing ineffective queries.

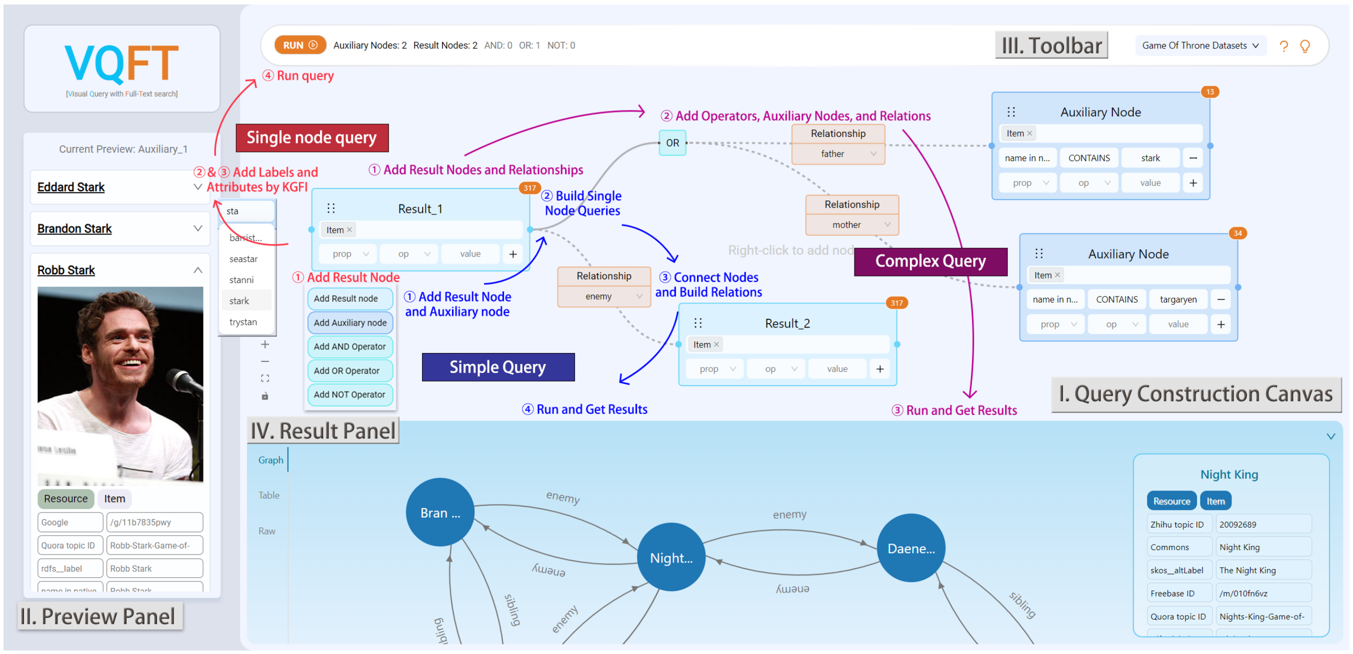

Figure 2: VQFT user interface and demonstration scenarios.

Once the query construction is completed, users can run the query by clicking the "RUN" button on the toolbar. The query will be executed by VQR, and the results can be presented in three formats on the results panel, i.e., graph, table, and raw data, respectively.

As shown above, VQFT has provided an intuitive, reliable, and easy-to-learn interface for non-expert users, which greatly reduces the complexity of querying KGs.

2.4 Visual Query Runner (VQR)

As aforementioned, after the query is constructed, the visual elements on the canvas are integrated into the query language by VQR. Multiple full-text search filters are combined into a faceted full-text search using logical operators. Then, VQR will execute the query in the backend KG database and retrieve the results which will be displayed in the result panel in Figure 2.

Overall, visual query construction and background query processing are performed and integrated by VQR in this step. The visual query constructed by the user is transformed into a query statement, which can be executed in the database. Also, intuitive query results are obtained in this step. It makes the entire query process more comprehensive and perceptive, further reducing the complexity of querying for non-expert users.

3 DEMONSTRATION

We have developed the GUI of VQFT using Vue framework. The data used for the demonstration, which is extracted from Wikidata, includes pre-built full-text indexes constructed by KGFI. The full-text search functionality and database operations are managed by a Node.js backend. We will present some demonstration scenarios that cover most use cases, guiding participants to understand the operation of VQFT, thus experiencing the demo system of VQFT.

Single node query. For users who are not familiar with KGs, building a basic query should be as easy as using a search engine. Keeping this in mind, we demonstrate a scenario where a user constructs a query for a single node.

① To add a "Result Node" to the VQFT canvas, simply right-click on the desired location of the canvas. This will create a new node with a label (i.e., type) filter and an attribute filter constructor.

② The label filter enables users to select and filter according to their type, and it also supports multi-selection for labels.

③ The attribute filter constructor allows the user to select attributes and operators. Keywords can be entered in the search box, which is similar to a traditional search engine. The filter can then be added to the query by clicking the "PLUS" button.

④ To add more attribute filters and build faceted queries, users can repeat Step 3. Once completed, click the "Run" button on the toolbar to execute the query and get the results.

When a type filter is added to a node, the attribute filter constructor changes to return only relevant attributes. Users are provided with input hints and suggestions during both selection and search process, respectively. All operations are accompanied by the mechanism of real-time preview feedback. These techniques can significantly improve the efficiency of users while constructing queries. For non-experts users, these operations are fully intuitive and easy to learn.

Simple query. For users with a basic understanding of KG, the following process can be used to quickly construct a query that matches a basic graph pattern.

① Right-click on the canvas to add nodes. Use "Result Node" for the results intended to be queried, whereas "Auxiliary Node" should be employed for nodes that do not need output.

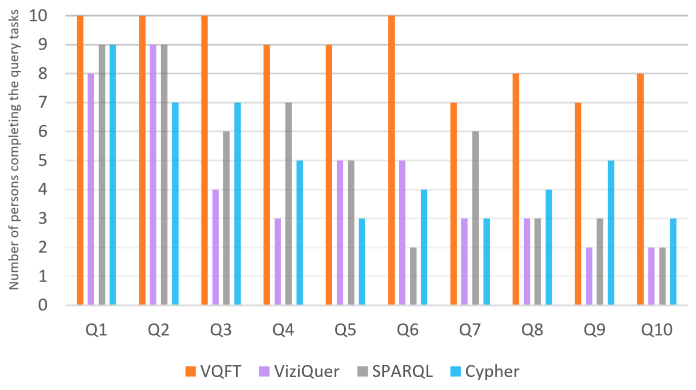

Figure 3: User test results

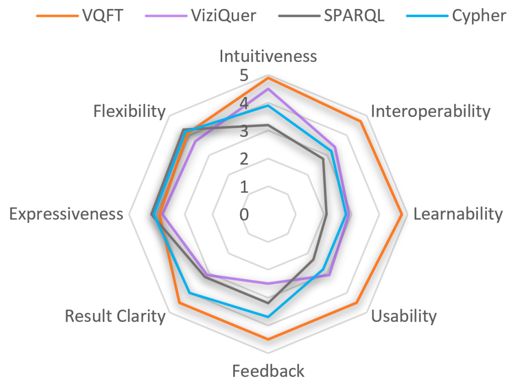

Figure 4: User research results in radar chart

(2) For each node on the canvas, the full-text search construction can be similar to the construction of a "Single Node Query."

(3) Connections between nodes are facilitated by handles, with a flow animation showing the direction of the line. Each line has an optional label search bar. Nodes and edges can be removed with the backspace key or the context menu.

(4) Once a query construction is complete, the query will be executed to retrieve the results.

For users with a simple understanding of the KG schema, the process is both simple and intuitive.

Complex query. While this demonstration primarily targets non-expert users, for those who need to build complex graph queries, the logical operators *AND*, *OR*, and *NOT* are available. As shown in Figure 2, this scenario describes a query to find individuals whose fathers are named "Stark" or whose mothers are named "Targaryen", along with their enemies. One possible equivalent SPARQL query is as follows.

```
SELECT ?person ?enemy WHERE {
  {
    ?person rel:hasFather ?father .
    ?father foaf:name ?fatherName .
    FILTER(CONTAINS(LCASE(?fatherName), "stark"))
  } UNION {
    ?person rel:hasMother ?mother .
    ?mother foaf:name ?motherName .
    FILTER(CONTAINS(LCASE(?motherName), "targaryen"))
  }.
  OPTIONAL { ?person rel:hasEnemy ?enemy .}
}
```

The steps to build this query with VQFT are as follows.

(1) Add Result Nodes for outputting results and their enemies, connect them, and then add the enemy relationship.

(2) Right-click on the canvas to add an OR operator and connect it to the Result Node. After that, construct two Auxiliary Nodes to connect with the OR operator.

(3) Add an attribute filter to the auxiliary node and add father relationship and mother relationship to the edges connected to the OR operator, respectively.

In this way, users can construct queries without having to learn complex syntax or understand the knowledge graph schema.

Demonstration engagement. The existing benchmarks for evaluating KG query approaches focus primarily on the execution speed and the ability to construct complex queries, lacking an assessment of the complexity of constructing the queries. VQFT is evaluated with user testing and questionnaires. Ten testers without relevant knowledge are convened to complete ten query tasks with increasing complexity. We provide brief training about VQFT, ViziQuer, SPARQL, and Cypher for testers. The number of completed tasks within a given time for each query approach, which is shown in Figure 3, shows that the completed tasks of VQFT outperform other approaches, proving its ease of learning. The survey results of testers in Figure 4 also reveal that, compared with textual query languages, traditional visual query methods, such as ViziQuer, do not considerably reduce the learning complexity for users. In contrast, VQFT, which is more aligned with user intuition, presents a lower learning curve and is easier to use. This demonstrates that VQFT is an efficient and useful tool in assisting non-expert users to quickly become proficient in KG queries.

ACKNOWLEDGMENTS

This work is supported by HUAWEI. Xin Wang is the corresponding author of this paper.

REFERENCES

[1] Kārlis Čerāns, Agris Šostaks, Uldis Bojārs, Jūlija Ovčiņņikova, Lelde Lāce, Mikus Grasmanis, Aiga Romāne, Artūrs Sproģis, and Juris Bārzdiņš. 2018. ViziQuer: a web-based tool for visual diagrammatic queries over RDF data. In *The Semantic Web: ESWC 2018*. 158–163.
[2] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*. 1433–1445.
[3] Florian Haag, Steffen Lohmann, Stephan Siek, and Thomas Ertl. 2015. QueryVOWL: A visual query notation for linked data. In *The Semantic Web: ESWC 2015*. Springer, 387–402.
[4] Pengkai Liu, Xin Wang, Qiang Fu, Yajun Yang, Yuan-Fang Li, and Qingpeng Zhang. 2022. KGVQL: A knowledge graph visual query language with bidirectional transformations. *Knowledge-Based Systems* 250 (2022), 108870.
[5] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. 2009. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.* 34, 3 (2009).
[6] Xiang Wang, Xin Wang, Zhaozhuo Li, and Dong Han. 2023. KGNav: A Knowledge Graph Navigational Visual Query System. *Proceedings of the VLDB Endowment* 16, 12 (2023), 3946–3949.