Quantum-Inspired Keyword Search on Multi-Model Databases

Gongsheng Yuan[1,2], Jiaheng Lu[1], and Peifeng Su[3]

[1] Department of Computer Science, University of Helsinki, FI-00014, Helsinki, Finland
{gongsheng.yuan,jiaheng.lu}@helsinki.fi
[2] School of Information, Renmin University of China, Beijing 100872, China
[3] Department of Geosciences and Geography, University of Helsinki, FI-00014, Helsinki, Finland
peifeng.su@helsinki.fi

Abstract. With the rising applications implemented in different domains, it is inevitable to require databases to adopt corresponding appropriate data models to store and exchange data derived from various sources. To handle these data models in a single platform, the community of databases introduces a multi-model database. And many vendors are improving their products from supporting a single data model to being multi-model databases. Although this brings benefits, spending lots of enthusiasm to master one of the multi-model query languages for exploring a database is unfriendly to most users. Therefore, we study using keyword searches as an alternative way to explore and query multi-model databases. In this paper, we attempt to utilize quantum physics's probabilistic formalism to bring the problem into vector spaces and represent events (e.g., words) as subspaces. Then we employ a density matrix to encapsulate all the information over these subspaces and use density matrices to measure the divergence between query and candidate answers for finding top-k the most relevant results. In this process, we propose using pattern mining to identify compounds for improving accuracy and using dimensionality reduction for reducing complexity. Finally, empirical experiments demonstrate the performance superiority of our approaches over the state-of-the-art approaches.

1 Introduction

In the past decades, due to an explosion of applications with the goal of helping users address various transactions in different domains, there are increasing needs to store and query data produced by these applications efficiently. As a result, researchers have proposed diverse data models for handling these data, including structured, semi-structured, and graph models. Recently, to manage these data models better, the community of databases introduces an emerging concept, multi-model databases [16], which not only embraces a single and unified platform to manage both well-structured and NoSQL data, but also satisfies the system demands for performance, scalability, and fault tolerance.

Although multi-model database systems provide a way to handle various data models in a unified platform, users have to learn corresponding specific multi-model query languages to access different databases (e.g., using AQL to access ArangoDB [2], SQL++ for AsterixDB [1], and OrientDB SQL for OrientDB [19]). Moreover, users also need to understand complex and possibly evolving multi-model data schemas as background

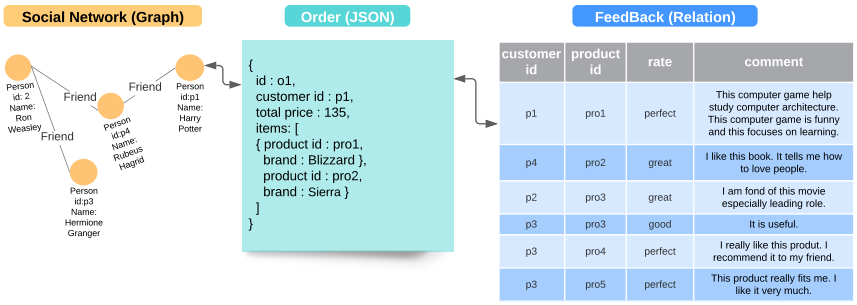

Fig. 1. An Example of Multi-model Data.

knowledge for using these query language. This is unfriendly to most users because it usually with a steeper learning curve. For example, Fig. 1 depicts multi-model data in social commerce. *Suppose we want to find Rubeus Hagrid's friends who have bought Blizzard and given a perfect rating.* It won't be an easy job to write a multi-model query involving social network (Graph), order (JSON), and feedback (Relation) for novices to achieve this goal. Therefore, in this paper, we study using keyword searches as an alternative way to explore and query multi-model databases, which does not require users to have strong background knowledge.

After reviewing the literature, we find that most existing works [27] only restrict keyword searches over a specific database supporting a single data model (e.g., relational, XML, and graph databases). Unfortunately, there is a lack of relevant research literature for the issue of performing keyword searches on multi-model databases. However, we think it is a promising research topic that remains a big challenge. This is because a trivial solution, which firstly performs a keyword search on individual data models by conventional methods and then combines results by assembling the previous results, cannot work well. The reason is that it may miss results that consist of multiple models simultaneously.

Facing this challenge, previous researchers used graph methods [14] to solve it. However, there are several matters needing attention in this method. Firstly, when we use a graph to represent these heterogeneous data, this graph may be vast and complex. So we need to divide this graph into many subgraphs, which is a graph partition problem. And if we perform keyword searches on these graphs to find answers, this means we need to find some subgraphs relating to partial keywords or all as the answer. Therefore, this involves subgraph matching and subgraph relevance ranking problems. And lastly, we need to consider how to take the dependencies among keywords and schema information into consideration when doing keyword searches. We could see all these problems have a significant influence on the final returned results. This means we should be careful to choose the corresponding solutions for these graph problems.

To avoid these graph issues, we start by introducing the "*quantum-inspired*" framework into the database community [28], in which we utilize the probabilistic formalism of quantum physics to do keyword searches. In quantum probability, the probabilistic space is naturally encapsulated in a vector space. Based on the notion of information

need vector space, we could regard the data in multi-model databases as the information (*statements*) collection, define events (e.g., words) as subspaces, use a density matrix (probability distribution) to encapsulate all the information over these subspaces for measuring the relevance of the candidate answers to the user's information need.

For this framework, the idea behind the quantum-inspired approach to disciplines other than physics is that, although macroscopic objects cannot exhibit the quantum properties manifested by particles such as photons, some phenomena can be described by the language or have some features of the phenomena (e.g., superposition) represented by the quantum mechanical framework in physics [24]. Therefore, except for the above theoretical method introduction, there are two other reasons underlying this attempt. One is that the similarity between the quantum mechanical framework to predict the values which can only be observed in conditions of uncertainty [24] and the decision about the relevance of the content of a text to an information need is subject to uncertainty [18]. Another one is that increasing works support the notion that quantum-like phenomena exist in human natural language and text, cognition and decision making [22], all related to the critical features of keyword searches.

Besides, the pioneering work [24] formalized quantum theory as a formal language to describe the objects and processes in information retrieval. Based on this idea, we use this mathematical language to describe relational, JSON, and graph data in the database community as information collection. Next, we transform keyword searches from a querying-matching work into a calculating-ranking task over this information collection and return the most relevant top-k results. And we take the possible relevance among input query keywords and database schema information into consideration, which helps the framework understand the user's goal better. Now, we summarize our contributions as follows:

1. Based on quantum theory, we attempt to use a quantum-inspired framework to do keyword searches on multi-model databases, utilizing quantum physics' probabilistic formalism to bring the problem into information need vector space. We want to take advantage of the ability of quantum-inspired framework in capturing potential semantics of candidate answers and query keywords for improving query accuracy.
2. In this process, we introduce the co-location concept to identify compounds for enhancing the topic of statements. We want to use it to improve query performance (precision, recall, and F-measure).
3. By analyzing the existing quantum-inspired method, we propose constructing a query density vector instead of a density matrix to reduce the framework's complexity. And we present an algorithm to perform keyword searches on multi-model databases.
4. Finally, we perform extensive empirical experiments that demonstrate our approaches' overall performance is better than the state-of-the-art approaches. The F-measure is at least nearly doubled.

2 Preliminaries

For the sake of simplicity, we assume the vector space is $\mathbb{R}^n$ in this paper. A unit vector $\vec{u} \in \mathbb{R}^n$ is defined as $|u\rangle$ and termed as *ket* on the basis of Dirac notation. Its

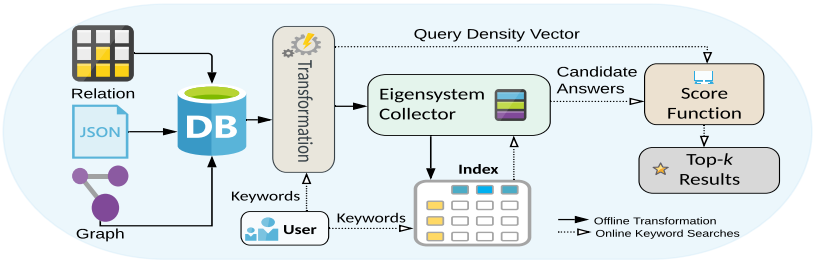

Fig. 2. The Framework of Quantum-Inspired Keyword Search

conjugate transpose $\vec{u}^H = \vec{u}^T$ is written as $\langle u|$ and called *bra*. The inner product between two vectors is denoted as $\langle u|v\rangle = \sum_{i=1}^{n} u_i v_i$. The outer product between two vectors is denoted as $|u\rangle\langle v|$ and called *dyad*. When the vector has a unitary length (i.e., $\|\vec{u}\|_2 = 1$), a special operator can be defined as the outer product between two vectors. We call this operator *projector*. To explain this, suppose $|u\rangle$ is a vector; the projector corresponding to this vector is a dyad written as $|u\rangle\langle u|$. For example, if there is $|u_1\rangle = (1,0)^T$, the projector corresponding to this ket is transforming it into $|u_1\rangle\langle u_1| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. Due to the projector, $|u\rangle$ could be mapped to the generalized probability space. In this space, each rank-one dyad $|u\rangle\langle u|$ can represent a quantum *elementary event* and each dyad $|\kappa\rangle\langle\kappa|$ represent a *superposition event*, where $|\kappa\rangle = \sum_{i=1}^{p} \sigma_i |u_i\rangle$, the coefficients $\sigma_i \in \mathbb{R}$ satisfy $\sum_i \sigma_i^2 = 1$. And density matrices ρ are interpreted as generalized probability distributions over the set of dyads when its dimension greater than 2 according to Gleason's Theorem [7]. A real density matrix ρ is a positive semidefinite ($\rho \geq 0$) Hermitian matrix ($\rho = \rho^H = \rho^T$) and has trace 1 ($Tr(\rho) = 1$). It assigns a generalized probability to each dyad $|u\rangle\langle u|$, whose formula is:

$$\mu_\rho(|u\rangle\langle u|) = Tr(\rho\,|u\rangle\langle u|). \tag{1}$$

For example, $\rho_1 = \begin{pmatrix} 0.75 & 0 \\ 0 & 0.25 \end{pmatrix}$, $\rho_2 = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$, density matrix ρ_2 assigns a probability value $Tr(\rho_2\,|u_1\rangle\langle u_1|) = 0.5$ to the event $|u_1\rangle\langle u_1|$. If ρ is unknown, we could utilize the *Maximum Likelihood* (MaxLik) estimation to get it. Finally, through the value of negative Von-Neumann Divergence (VND) between ρ_q (query) and ρ_c (candidate answer), we could get their difference. Its formalization is:

$$-\Delta_{VN}(\rho_q||\rho_c) \overset{rank}{=} \sum_i \lambda_{q_i} \sum_j (\log \lambda_{c_j} \langle q_i|c_j\rangle^2). \tag{2}$$

3 The framework of the Quantum-inspired Keyword search

An overview of this entire framework is shown in Fig. 2, which has two functional modules. One is offline transformation. We will use quantum language to represent

heterogeneous data in a uniform format. Another one is online keyword searches with generalized probabilities for finding the most relevant results to answer keyword searches.

3.1 Transformation

Here we consider three kinds of data. They are relational, JSON, and graph data. For the **relational** data model, it is natural to think of a tuple with schema in tables as a *statement* (a piece of information). To avoid information explosion (caused by information fragmentation), we observe the **JSON** file in a coarse granularity and treat each object in JSON as an integral *statement*. For each node in the **graph** data, we put information about all of its neighbor nodes in the same *statement*, including itself information. In this way, it can keep complete neighbor relationships and node information.

Only considering these discrete data is not enough. We need to take join into consideration to get complete information in our framework. For this problem, there are many methods. For example, we could mine the relationship among different data models with [5], then join them. Or we could choose meaningfully related domains to do equi-join operations based on expert knowledge, which is a quite straightforward and easy way for finding meaningful joins and cutting down the search space by avoiding generating a lot of useless intermediate results. Since this part is not the focus of this paper, we will use the later one to continue our work.

Now we have gotten a statement collection. Next, we use the mathematical language of quantum theory to represent these statements. To achieve this goal, we first use an elementary event to represent a single word and use a superposition event to represent a compound. In this process, to construct proper superposition events to enhance the topic of statement for improving query accuracy (i.e., choose appropriate compounds κ from a given statement and determine the value of each coefficient σ_i in κ), we introduce a special spatial pattern concept called *co-location pattern* [12]. After this process, we could get an event set $\mathscr{P}_{st}$ for each statement. Then, we learn a density matrix from each event set $\mathscr{P}_{st}$ with MaxLik estimation and use this density matrix to represent the corresponding statement. Here, we use a $R\rho R$ algorithm, which is an iterative schema of MaxLik outlined in the [17]. And this iterative schema has been used in the [23] for getting density matrices. Finally, we could use VND to measure the difference between different statements by their density matrices and use VND' values to rank them.

To get an event set $\mathscr{P}_{st}$ for each statement, we introduce a special spatial pattern concept called *co-location pattern* [12] to help construct compounds κ. Next, we will start by redefining some concepts in the field of co-location to adapt our problem for identifying compounds. When a word set $c = \{w_1, ..., w_p\}, \|c\| = p$, appears in any order in a fixed-window of given length L, we say there is a relationship R among these words. For any word set c, if c satisfies R in a statement, we call c co-location compound.

Definition 1. ***Participation ratio (PR)*** *$PR(c, w_i)$ represents the participation ratio of word w_i in a co-location compound $c = \{w_1, ..., w_p\}$, i.e.*

$$PR(c, w_i) = \frac{T(c)}{T(w_i)} \quad w_i \in c, \tag{3}$$

where $T(c)$ is how many times the c is observed together in the statement , $T(w_i)$ is the number of word w_i in the whole statement.

Table 1. Example about identifying compound κ

$\mathscr{W}_{st}$ = {*This computer game help study computer architecture this computer game is funny and this focuses on learning.* }				
c_1 = { *computer, game* }	PR(c_1, computer) PR(c_1, game)	$\frac{2}{3}$ $\frac{2}{2}$	PI(c_1)	$\frac{2}{3}$
c_2 = { *game, architecture* }	PR(c_2, game) PR(c_2, architecture)	$\frac{1}{2}$ $\frac{1}{1}$	PI(c_2)	$\frac{1}{2}$
c_3 = { *computer, architecture* }	PR(c_3, computer) PR(c_3, architecture)	$\frac{1}{3}$ $\frac{1}{1}$	PI(c_3)	$\frac{1}{3}$
c_4 = { *computer, game, architecture* }	PR(c_4, computer) PR(c_4, game) PR(c_4, architecture)	$\frac{1}{3}$ $\frac{1}{2}$ $\frac{1}{1}$	PI(c_4)	$\frac{1}{3}$
PI(c_1) $\geq$ *min_threshold* = 0.6, c_1 is a compound κ				

Definition 2. ***Participation index (PI)*** *$PI(c)$ of a co-location compound c is defined as:*

$$PI(c) = \min_{i=1}^{p}\{PR(c, w_i)\}. \tag{4}$$

Given a minimum threshold *min_threshold*, a co-location compound c is a compound κ if and only if $PI(c) \geq$ *min_threshold*. It is obvious that when *min_threshold* = 0, all the co-location compounds are compounds. Based on the method of estimating whether c is a compound κ, with each appearance of word $w_i \in c$ in the statement, there has been a great effect on calculating value of *PR* which determines whether $PI(c)$ is greater than or equal to *min_threshold*. This is to say each appearance of word w_i plays an important role in the statement for expressing information. Therefore, when we set a value for the coefficient σ_i of $|w_i\rangle$ in κ , we need to take the above situation into consideration. A natural way is to set $\sigma_i^2 = T(w_i)/\sum_{j=1}^{p} T(w_j)$ for each w_i in a compound $\kappa = \{w_1, ..., w_p\}, 1 \leq \mathrm{i} \leq p$, where $T(w_i)$ is the number of times that word w_i occurs in this statement. We call it *co-location weight*.

For example, we assume "This computer game help study ..." is a statement $\mathscr{W}_{st}$, which is in the table FeedBack of Fig. 1. Given c_1, c_2, c_3, and c_4 (see Table 1), if *min_threshold* = 0.6, thus c_1 is a compound. This helps our framework to have a better understanding of this statement whose theme is about the computer game, not computer architecture. Meanwhile, with co-location weight we could set $\sigma^2_{computer} = 3/5$, $\sigma^2_{game} = 2/5$ for the κ = {*computer, game*}. According to the related state-of-the-art work [4], the fixed-window of Length L could be set to $l||c||$. And it provides a robust pool for l. Here, we select l = 1 from the robust pool. Because if one decides to increase l, more inaccurate relations will be detected, and performance will deteriorate.

Next, we could use an iterative schema of $MaxLik_{matrix}$ [23] to learn a density matrix from each event set $\mathscr{P}_{st}$. However, unfortunately, both dyads and density matrices are $n \times n$ matrices, i.e., their dimensions depend on n, the size of word space $\mathscr{V}$. When n is

bigger, the cost of calculating is higher, especially in the field of databases. Therefore, we need to find a way to reduce complexity.

According to the Equation (1), the probability of quantum event Π is $Tr(\rho\Pi)$. Through $Tr(\rho\Pi)$, we could get the dot product of two vectors, i.e.,

$$Tr(\rho\Pi) = \vec{\rho}.\vec{\Pi}^2, \tag{5}$$

where $\vec{\rho} = (\beta_1, \beta_2, \ldots, \beta_n)$, we call it *density vector*. If we assume $\vec{\Pi} = ((\sum_{i=1}^{n} u_{i1}v_{i1}), (\sum_{i=1}^{n} u_{i1}v_{i2}), \ldots, (\sum_{i=1}^{n} u_{i1}v_{in}))$, then $\vec{\Pi}^2 = ((\sum_{i=1}^{n} u_{i1}v_{i1})^2, (\sum_{i=1}^{n} u_{i1}v_{i2})^2, \ldots, (\sum_{i=1}^{n} u_{i1}v_{in})^2)$.

Definition 3. ***Density vector*** *A density vector is a real vector* $\vec{\rho} = (\beta_1, \beta_2, \ldots, \beta_n)$*, noted by* $\langle\rho|$*, where* $\beta_i \geq 0$*, and* $\beta_1 + \beta_2 + \ldots + \beta_n = 1$.

For example, density vector $\langle\rho_3|$ = (0.2, 0.1, 0.7). Next, we will use a density vector instead of a density matrix in the rest of this paper. This is because it has three advantages.

- Firstly, density vectors have a more concise way of calculating quantum probabilities than density matrices' (comparing Equation 1 with Equation 6);
- Secondly, when one calculates VND between two density vectors, it is faster to get the value of VND than density matrices do (comparing Equation 2 with Equation 7);
- Thirdly, comparing with density matrices, learning a query density vector from the input keywords is easier.

Based on the generated density matrices, we could get *val* (the list of all eigenvalues) and *vec* (the list of eigenvectors, each eigenvector corresponding to its own eigenvalue) through eigendecomposition of density matrices. To simplify our framework further, we use the Principal Component Analysis (PCA) method to take the first h largest eigenvalues and require that the sum of these values is greater than or equal to a $threshold$. Based on the research concerning PCA [10], we could set $threshold$ = 85%, which is enough for our work. This is because if one increases $threshold$, a higher $threshold$ may have a higher chance to cause the increase of the number of val_i instead of precision of framework and deteriorate the performance of the framework. Then, we store h eigenvalues and corresponding eigenvectors vec_i (termed as $\{val, vec\}$) into the set of density systems, S_{denSys} (the component, "Eigensystem Collector" in Fig.2).

We have now transformed all kinds of data into eigenvalues (density vectors) and eigenvectors (mapping directions) and converted querying-matching problem into calculating-ranking. In this way, we eliminate heterogeneous data and convert the querying-matching problem into calculating-ranking. Next, we will consider how to do keyword searches on these density vectors online.

3.2 Online Keyword Search Query

In this module, when users submit their queries, there are two data flows about queries.

- The first one is from users to "Transformation", which will construct query density vectors from input keywords. This step will consider the relationship between input keywords instead of treating keywords as mutually independent entities.

– Another one is from users to index structure for getting candidate answers, which will reduce the query scope through our index structure (inverted list).

In the above data flows, it involves constructing query density vectors. Before doing it, we propose the new representation for single words and compounds in query keywords. Through analyzing Equation (5), we could represent each word $w_i \in \mathscr{V}$ by $|e_i\rangle$, where $|e_i\rangle$, the standard basis vector, is an one-hot vector. Based on this new representation, we present compound by $|\kappa\rangle = \sum_{i=1}^{p} \sigma_i |e_{w_i}\rangle$, where $\kappa = \{w_1, ..., w_p\}$, the coefficients $\sigma_i \in \mathbb{R}$ satisfy $\sum_{i=1}^{p} \sigma_i^2 = 1$ to guarantee the proper normalization of $|\kappa\rangle$.

For example, Considering $n = 3$ and $\mathscr{V} = \{computer, science, department\}$, if $\kappa_{cs} = \{computer, science\}$ and $|\kappa_{cs}\rangle = \sqrt{2/3}|e_c\rangle + \sqrt{1/3}|e_s\rangle$, then we could get:

$$\left|e_{computer}\right\rangle = \begin{pmatrix}1\\0\\0\end{pmatrix}, \quad |e_{science}\rangle = \begin{pmatrix}0\\1\\0\end{pmatrix}, \quad |\kappa_{cs}\rangle = \sqrt{\frac{2}{3}}\begin{pmatrix}1\\0\\0\end{pmatrix} + \sqrt{\frac{1}{3}}\begin{pmatrix}0\\1\\0\end{pmatrix} = \begin{pmatrix}\sqrt{\frac{2}{3}}\\\sqrt{\frac{1}{3}}\\0\end{pmatrix}.$$

Next, we still use the iterative schema of *Maximum Likelihood* (MaxLik) estimation for getting the query density vector. We would also use the following formula to get the quantum probability of event $|\Pi\rangle$ at *MaxLik* estimation.

$$Probability_{\Pi} = \left\langle \rho \middle| \Pi^2 \right\rangle. \tag{6}$$

Finally, we propose Algorithm 1 to do online keyword searches, in which we use line 1 - 9 to get candidate density system set C_{denSys} through the index. Then we use them to help get corresponding query density vectors and regard them as the score function (VND) inputs to answer queries (line 10 - 26).

Firstly, we could classify the input keywords into two groups according to whether or not w_i is relevant to the database schema. For the group K_{schema}, we use the union to get a candidate density system set C_{schema} in line 4, where C_{w_i} can be gotten by the inverted list, which is relevant to the input word w_i. For the K_{non_schema}, we use intersection to get C_{non_schema}. Then we could get the candidate density system set C_{denSys} through difference (line 9). We use these operations to take advantage of the schema for helping users explore more potential possibilities about answers.

Line 10 - 26 are going to get top-k results. Firstly, we use every mapping direction vec_{st} of $\{val, vec\}_{st}$ in the candidate density systems C_{denSys} to transform original events into new representations so that the algorithm could learn a query density vector from these new events. The above transforming could guarantee that each constructed query density vector and corresponding density vector val_{st} in one coordinate system. Then we could calculate the divergence between $\langle\rho|_q$ and $\langle\rho|_{st}$ (val_{st}) by Equation 7 in line 23.

$$-\Delta_{VN}(\langle\rho|_q \,||\, \langle\rho|_{st}) \overset{rank}{=} \sum_i \beta_{q_i} \log \beta_{st_i}, \ where\ \beta_{q_i} \in \langle\rho|_q, \beta_{st_i} \in \langle\rho|_{st}. \tag{7}$$

Considering the previous problem "*Suppose we want to find Rubeus Hagrid's friends who have bought Blizzard and given a perfect rating.*", we take {Rubeus Hagrid friends Blizzard perfect} as Algorithm 1 inputs and perform keyword searches on the Fig. 1.

Algorithm 1 Answer Keyword Searches with Density Vectors

Input: Input keywords $K = \{w_1, \ldots, w_t\}$
Output: the top-k most relevant statements about queries

1: $\{K_{schema}, K_{non\_schema}\} = Classification(K)$
2: $C_{schema} \leftarrow \Phi$
3: **for** each $w_i \in K_{schema}$ **do**
4: $\quad C_{schema} \leftarrow C_{schema} \cup (C_{w_i} \subset S_{denSys})$
5: **end for**
6: **for** all $w_j \in K_{non\_schema}$ **do**
7: $\quad C_{non\_schema} \leftarrow C_{w_1} \cap, \ldots, \cap C_{w_j}$
8: **end for**
9: $C_{denSys} \leftarrow C_{non\_schema}$ - C_{schema}
10: $S_{Result} \leftarrow \Phi$
11: **for** each $\{val, vec\}_{st} \in C_{denSys}$ **do**
12: $\quad \mathscr{P}_q \leftarrow \Phi$
13: $\quad$ **for** each $w_i \in K$ **do**
14: $\quad\quad$ Get event $\vec{\Pi}_{w_i}$ through rotating the $|e_i\rangle$ (w_i) into a new coordinate by vec_{st}
15: $\quad\quad \mathscr{P}_q \leftarrow \mathscr{P}_q \cup \vec{\Pi}_{w_i}$
16: $\quad$ **end for**
17: $\quad$ **for** each c *in* K **do**
18: $\quad\quad$ **if** $PI(c) \geq min\_threshold$ **then**
19: $\quad\quad\quad \mathscr{P}_q \leftarrow \mathscr{P}_q \cup |\kappa_c\rangle$
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: $\quad$ Learn a query density vector $\langle\rho|_q$ from $\mathscr{P}_q$ by $MaxLik$
23: $\quad score_{st} = ScoreFunction(\langle\rho|_q, val_{st})$
24: $\quad S_{Result} \leftarrow S_{Result} \cup score_{st}$
25: **end for**
26: Sort S_{Result} and return top-k results

And we could get the result in which a person named Harry Potter bought Blizzard and gave a perfect rating, and he is Rubeus Hagrid's friend. The original result is "*social network person id p1 name harry potter friend person id p4 name rubeus hagrid order id o1 custom id p1 total price 135 item product id pro1 brand blizzard feedback rate perfect comment this computer game help study computer architecture this computer game is funny and this focuses on learning*". And its score is -2.665.

4 Experiment

4.1 Data Sets

We use synthetic data (UniBench) and real data (IMDB and DBLP) to evaluate our approaches. The statistics about them are listed in Table 2.

UniBench [29] is a multi-model benchmark, including data of relational, JSON, and graph models. It simulates a social commerce scenario that combines the social network with the E-commerce context. The relational model includes the structured feedback

Table 2. The number of records/objects in different data models

	Relational	JSON	Graph-entity	Graph-relation
UniBench	150 000	142 257	9 949	375 620
IMDB	494 295	84 309	113 858	833 178
DBLP	1 182 391	435 469	512 768	3 492 502

Table 3. Queries employed in the experiments

ID	Queries
(a) Queries on DBLP	
Q_1	Soni Darmawan friends
Q_2	Gerd Hoff friends' rank 1 paper
Q_3	Slawomir Zadrozny rank 1 paper
Q_4	Phebe Vayanos phdthesis paper
Q_5	neural sequence model
Q_6	Brian Peach 2019 papers
Q_7	Performance of D2D underlay and overlay for multi-class elastic traffic. authors
Q_8	Carmen Heine rank Modell zur Produktion von Online-Hilfen.
Q_9	Exploring DSCP modification pathologies in the Internet.
Q_{10}	The papers of Frank Niessink
(b) Queries on UniBench	
Q_{11}	Abdul Rahman Budjana friends BURRDA feedback perfect
Q_{12}	Shmaryahu Alhouthi order Li-Ning Powertec Fitness Roman Chair
Q_{13}	Kamel Abderrahmane Topeak Dual Touch Bike Storage Stand
Q_{14}	Alexandru Bittman whether has friend Ivan Popov
Q_{15}	Mohammad Ali Forouhar Oakley Radar Path Sunglasses
Q_{16}	Soft Air Thompson 1928 AEG Airsoft Gun and Genuine Italian Officer's Wool Blanket
Q_{17}	Roberto Castillo Total Gym XLS Trainer and Reebok
Q_{18}	Who Kettler, Volkl and Zero Tolerance Combat Folding Knife
Q_{19}	Francois Nath Nemo Cosmo Air with Pillowtop Sleeping Pad
Q_{20}	Hernaldo Zuniga Advanced Elements AdvancedFrame Expedition Kayak and TRYMAX
(c) Queries on IMDB	
Q_{21}	Lock, Stock and Two Smoking Barrels actors
Q_{22}	Forrest Gump
Q_{23}	The title and imdbVote of films of Bruce Willis
Q_{24}	The films of director Robert Zemeckis
Q_{25}	films in 1997 Genre Action, Adventure, Family
Q_{26}	The Legend of 1900 awards
Q_{27}	Scent of a Woman imdbRating
Q_{28}	The film of Dustin Hoffman with Tom Cruise
Q_{29}	Morgan Freeman friends
Q_{30}	Aamir Khan films

information; The JSON model contains the semi-structured orders; The social network is modeled as a graph, which contains one entity and one relation, i.e., customer, and person knows person. These also have correlations across the data models. For instance, the customer makes transactions (Graph correlates with JSON).

The IMDB dataset is crawled from website[4] by OMDB API. Through extracting inner relationships and potential information, we generate several data models to represent the original data. The relational data includes performing information and rating information, which are stored in different tables; The JSON model is made up of film information (e.g., imdbID, title, and year); The graph is about cooperative information, where two actors would be linked together if they have ever worked for the same movie.

The DBLP[5] data consists of bibliography records in computer science. Each record in DBLP is associated with several attributes such as authors, year, and title. The raw data is in XML format. Here we describe it in three data models. The publication records are presented in relational data, including author id, paper id, and the author's rank in the author list. A subset of the papers' attributes (e.g., paper id, key, and title) are represented in JSON. We also construct a co-authorship (friend) graph where two authors are connected if they publish at least one paper together.

Table 4. Presion, Recall, F-measure on DBLP

	AKSDV							EASE
min_threshold	0	0.2	0.4	0.6	0.8	1.0	non	
Precision	0.830	0.847	0.847	0.847	0.847	0.847	0.797	0.090
Recall	0.867	0.917	0.917	0.917	0.917	0.917	0.817	0.500
F-measure	0.834	0.861	0.861	0.861	0.861	0.861	0.794	0.141

4.2 Queries and Answers

In the experiments, three groups of keyword queries, as shown in Table 3, are proposed by a few people randomly to evaluate our methods on the DBLP, UniBench, and IMDB datasets, respectively. Each keyword query involves one or more than one data model to test the ability of our methods in capturing potential semantics of keywords and search accuracy. Besides, the corresponding AQL query for each Q_i is issued in ArangoDB, and the output answers are used to evaluate the results produced by the algorithm, Answer Keyword Searches with Density Vectors (AKSDV), and EASE [14]. EASE models heterogeneous data as graphs and aims at finding r-radius Steiner graphs as query results. In this method, each returned Steiner graph includes at least two keywords.

The experiments are implemented in Java except for eigendecomposition by Matlab (offline work). The experiments are run on a desktop PC with an Intel(R) Core(TM) i5-6500 CPU of 3.19 GHz and 16GB RAM. Note that all operations are done in memory,

[4] https://www.omdbapi.com/

[5] https://dblp.uni-trier.de/xml/

Table 5. Presion, Recall, F-measure on UniBench

	AKSDV							EASE
min_threshold	0	0.2	0.4	0.6	0.8	1.0	non	
Precision	0.902	0.902	0.917	0.922	0.922	0.922	0.897	0.220
Recall	0.883	0.883	0.885	0.886	0.886	0.886	0.882	0.136
F-measure	0.844	0.844	0.848	0.850	0.850	0.850	0.842	0.061

Table 6. Presion, Recall, F-measure on IMDB

	AKSDV							EASE
min_threshold	0	0.2	0.4	0.6	0.8	1.0	non	
Precision	0.753	0.753	0.753	0.758	0.758	0.758	0.758	0.548
Recall	0.782	0.782	0.782	0.784	0.784	0.784	0.784	0.466
F-measure	0.657	0.657	0.657	0.661	0.661	0.661	0.661	0.269

and the standard NLP pre-processing such as dropping the stop words and stemming are conducted in advance. In the experiments, we measure the precision, recall, and F-measure for the top-20 returned results.

4.3 Results Analysis

Table 4 shows the comparison of the average precision, recall, and F-measure of AKSDV with EASE's. This comparison result demonstrates that our proposed methods outperform EASE on the DBLP dataset. Table 5 and Table 6 show that the performance of AKSDV also outperforms EASE's on the UniBench and IDMB dataset. And the F-measure of AKSDV is at least nearly twice EASE's on these datasets. These high accuracy values show that our framework could understand the potential semantics underlying the statements and get the most relevant statements about queries.

For example, Q_9 wants to find all information about the paper "Exploring DSCP modification pathologies in the Internet". EASE returns a Steiner graph consisting of a single node that includes the paper name itself. AKSDV could find all of the information about this paper; Q_{11} wants to look for Abdul Rahman's friends who have bought BURRDA and given a perfect rating. For EASE, it returns answers mainly about "Abdul Rahman". But AKSDV could return the relevant information which users want to find.

In these three tables, the "*min_threshold*" decides which co-location compounds will be regarded as compounds $\kappa = \{w_1, ..., w_p\}$. Each column corresponds to the performance of keyword searches when assigned a value to *min_threshold*. For example, in Table 4, the second column illustrates that when we set *min_threshold* = 0, the values of average precision, recall, and F-measure of AKSDV on the DBLP data set are 0.830, 0.867, and 0.834, respectively. In general, our method of identifying compounds works well on these datasets for improving query performance.

Column "non" means that we assign an average weight ($\sigma_i^2 = 1/p$, $||\kappa|| = p$) to each w_i in the compound κ instead of considering which word will have more contributions to the compound (without using co-location weight). Column "0" and "non" regard all the

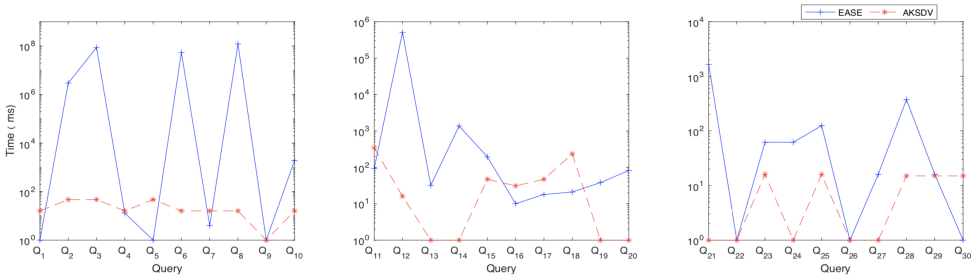

Fig. 3. Execution time of Q_i

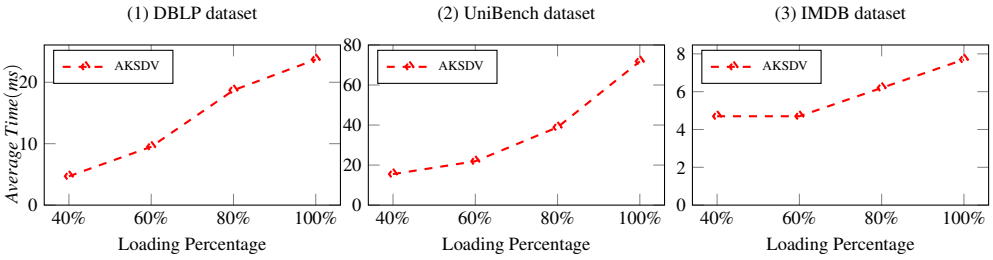

Fig. 4. Scalability

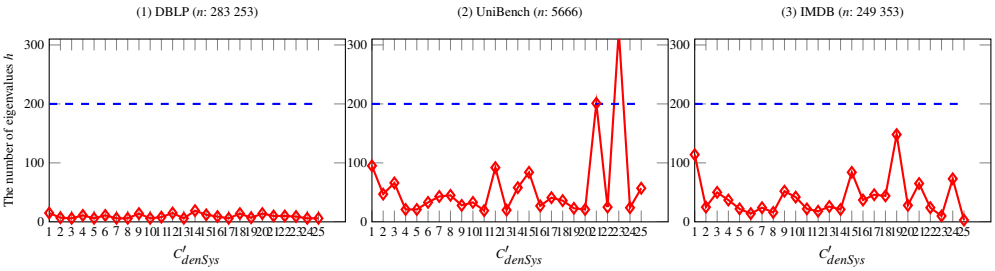

Fig. 5. The change in value of h on different datasets

co-location compounds as compounds. The difference between them is whether using co-location weight when constructing compounds. Table 4 and Table 5 demonstrate our co-location weight is better than the average weight method on the DBLP and UniBench dataset. In Table 6, the performance of column "0" is little less than the column "non"s.

4.4 Time and Scalability Analysis

In this part of the experiments, firstly, we analyze the time cost of AKSDV and EASE. The reported time cost values are collected by executing each query several times to take the median value. Fig. 3 gives executing time of AKSDV ($min\_threshold$ = 0.6) and EASE. In Fig. 3, there is a different fluctuation in the scales of time about different

queries, which is caused by the queries of complexity involving different data models. But in general, comparing with EASE, AKSDV has a good time performance.

Now, we focus on the scalability of AKSDV. To test the scalability, we load different proportions of metadata into our framework, respectively. And we let the correct results increase as loading more data for each query. Then, we perform all queries in table 3 on these datasets having different sizes to get a median time after executing several times. Finally, we calculate the average time on these datasets, respectively. Fig. 4 (1)-(3) show the results of experiments on the DBLP, Unibench, and IMDB dataset. In general, AKSDV has a good scalability performance on these datasets. The UniBench dataset, due to existing mass joins as the loading percentage increases, shows the query time increases faster than others. But generally, it is within an acceptable range.

4.5 The Dimension of Density Vectors Analysis

Unlike the word embedding in the machine learning community, which needs a toolkit using lots of time and samples to train vector space models, we only need to extend the events' dimension. And elementary events still keep disjoint feature. Although the dimensions of events have increased, the query's cost still depends on the h (the number of eigenvalues of density matrices). The Fig. 5 shows the change in values of h on the candidate statement set C'_{eigen}, where $C'_{eigen} \subset C_{eigen}$ and C_{eigen} is made of candidate statements about all the queries on different datasets in Table 3, and C'_{eigen} is made of selected 25 candidate answers from the corresponding C_{eigen} randomly. In Fig. 5, we can see the values of h are much less than the word space n, even less than 200 in most situations. It guarantees the complexity of our framework at the appropriate level.

5 Related works

Keyword search has been well known as a user-friendly way of satisfying users' information needs with few keywords in diverse fields such as Information Retrieval (IR) and database. Unfortunately, finding a fundamental axiomatic formulation of the IR field has proven tricky, maybe because of the intrinsic role humans play in the process [3]. However, due to information having proven considerably easier to capture mathematically than "meaning", researchers are investigating how to apply quantum theory to attack the various IR challenges, which can be traced back to [21]. Piwowarski et al. [20] used the probabilistic formalism of quantum theory to build a principled interactive IR framework. Frommholz et al. [6] utilized poly representation and quantum measurement [25] to do keyword searches in a quantum-inspired interactive framework.

Considering the power of quantum-inspired framework and the similarity of keyword searches between databases and IR, we attempt to use the quantum-inspired method to support keyword searches on multi-model databases. Hence our research work also conforms to the current trend of seamlessly integrating database and information retrieval [26]. The keyword search on the database is particularly appealing. From relation, XML to graph databases, there are already many exciting proposals in the scientific literature [13,11,8,9]. However, most of the existing keyword search works are designed for specific data models. Through review literature, the only complete relevant current work

is EASE [14]. They returned r-radius Steiner graphs as results to users for answering keyword searches on heterogeneous data. In addition, there is another work [15], which presented an architecture to support keyword searches over diverse data sources. Due to lacking complete performance analysis, we temporarily do not compare it in this paper.

6 Conclusion

This paper has proposed using the quantum probability mechanism to solve the keyword search problem on multi-model databases. To reduce complexity and improve performance, we introduced new representations of events and the density vector concept. We also used the spatial pattern to help improve query accuracy and used PCA to support the framework to work well further. Finally, extensive experiments have demonstrated the superiority of our methods over state-of-the-art works.

Acknowledgements. The work is partially supported by the China Scholarship Council and the Academy of Finland project (No. 310321). We would also like to thank all the reviewers for their valuable comments and helpful suggestions.

References

1. Altwaijry, S., Behm, A., Carey, V.B.Y.B.M., Cheelangi, I.C.M., Faraaz, K., Heilbron, E.G.R.G.Z., Vernica, P.P.V.T.R., Wen, J., Westmann, T.: Asterixdb: A scalable, open source bdms. Proceedings of the VLDB Endowment **7**(14) (2014)
2. ArangoDB: Three major nosql data models in one open-source database. (2016), `http://www.arangodb.com/`
3. Ashoori, E., Rudolph, T.: Commentary on Quantum-Inspired Information Retrieval. arXiv e-prints arXiv:1809.05685 (Sep 2018)
4. Bendersky, M., Croft, W.B.: Modeling higher-order term dependencies in information retrieval using query hypergraphs. In: SIGIR (2012)
5. Bergamaschi, S., Domnori, E., Guerra, F., Lado, R.T., Velegrakis, Y.: Keyword search over relational databases: a metadata approach. In: SIGMOD '11 (2011)
6. Frommholz, I., Larsen, B., Piwowarski, B., Lalmas, M., Ingwersen, P., Van Rijsbergen, K.: Supporting polyrepresentation in a quantum-inspired geometrical retrieval framework. In: Proceedings of the third symposium on Information interaction in context. pp. 115–124 (2010)
7. Gleason, A.M.: Measures on the closed subspaces of a hilbert space. Journal of mathematics and mechanics pp. 885–893 (1957)
8. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over xml documents. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data. pp. 16–27 (2003)
9. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: Proceedings of the 2007 ACM SIGMOD international conference on Management of data. pp. 305–316 (2007)
10. Holland, S.M.: Principal components analysis (pca). Department of Geology, University of Georgia, Athens, GA pp. 30602–2501 (2008)
11. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on xml graphs. In: Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405). pp. 367–378. IEEE (2003)

12. Huang, Y., Shekhar, S., Xiong, H.: Discovering colocation patterns from spatial data sets: a general approach. IEEE Transactions on Knowledge and data engineering **16**(12), 1472–1485 (2004)
13. Kargar, M., An, A., Cercone, N., Godfrey, P., Szlichta, J., Yu, X.: Meaningful keyword search in relational databases with large and complex schema. In: 2015 IEEE 31st International Conference on Data Engineering. pp. 411–422. IEEE (2015)
14. Li, G., Feng, J., Ooi, B.C., Wang, J., Zhou, L.: An effective 3-in-1 keyword search method over heterogeneous data sources. Information Systems **36**(2), 248–266 (2011)
15. Lin, C., Wang, J., Rong, C.: Towards heterogeneous keyword search. In: Proceedings of the ACM Turing 50th Celebration Conference-China. pp. 1–6 (2017)
16. Lu, J., Holubová, I.: Multi-model databases: a new journey to handle the variety of data. ACM Computing Surveys (CSUR) **52**(3), 1–38 (2019)
17. Lvovsky, A.: Iterative maximum-likelihood reconstruction in quantum homodyne tomography. Journal of Optics B: Quantum and Semiclassical Optics **6**(6), S556 (2004)
18. Melucci, M.: Deriving a quantum information retrieval basis. The Computer Journal **56**(11), 1279–1291 (2013)
19. OrientDB, D.: Orientdb. hybrid document-store and graph nosql database (2017)
20. Piwowarski, B., Frommholz, I., Lalmas, M., Van Rijsbergen, K.: What can quantum theory bring to information retrieval. In: Proceedings of the 19th ACM international conference on Information and knowledge management. pp. 59–68 (2010)
21. van Rijsbergen, C.J.: Towards an information logic. In: Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 77–86 (1989)
22. Song, D., Lalmas, M., Van Rijsbergen, K., Frommholz, I., Piwowarski, B., Wang, J., Zhang, P., Zuccon, G., Bruza, P., Arafat, S., et al.: How quantum theory is developing the field of information retrieval. In: 2010 AAAI Fall Symposium Series (2010)
23. Sordoni, A., Nie, J.Y., Bengio, Y.: Modeling term dependencies with quantum language models for ir. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 653–662 (2013)
24. Van Rijsbergen, C.J.: The geometry of information retrieval. Cambridge University Press (2004)
25. Wang, J., Song, D., Kaliciak, L.: Tensor product of correlated text and visual features. QI'10 (2010)
26. Weikum, G.: Db & ir: both sides now (keynote). In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (2007)
27. Yu, J.X., Qin, L., Chang, L.: Keyword search in databases. Synthesis Lectures on Data Management **1**(1), 1–155 (2009)
28. Yuan, G.: How the quantum-inspired framework supports keyword searches on multi-model databases. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 3257–3260 (2020)
29. Zhang, C., Lu, J., Xu, P., Chen, Y.: Unibench: A benchmark for multi-model database management systems. In: Technology Conference on Performance Evaluation and Benchmarking. pp. 7–23. Springer (2018)