FusionQuery: On-demand Fusion Queries over Multi-source Heterogeneous Data

Junhao Zhu
Zhejiang University
Hangzhou, China
zhujunhao@zju.edu.cn

Yuren Mao
Zhejiang University
Hangzhou, China
yuren.mao@zju.edu.cn

Lu Chen
Zhejiang University
Hangzhou, China
luchen@zju.edu.cn

Congcong Ge
Huawei Technologies Co., Ltd.
Hangzhou, China
gecongcong1@huawei.com

Ziheng Wei
Huawei Technologies Co., Ltd.
Hangzhou, China
ziheng.wei@huawei.com

Yunjun Gao
Zhejiang University
Hangzhou, China
gaoyj@zju.edu.cn

ABSTRACT

Centralised data management systems (e.g., data lakes) support queries over multi-source heterogeneous data. However, the query results from multiple sources commonly involve between-source conflicts, which makes query results unreliable and confusing and degrades the usability of centralised data management systems. Therefore, resolving the between-sourced conflicts is one of the most important problems for centralised data management systems. To solve it, many batch data fusion-based methods have been proposed, which require traversing all the data in the centralised data management systems and cause scalability and flexibility issues.

To address these issues, this paper explores the problem of on-demand fusion queries, where the between-sourced conflicts are solved with only the query-related data; moreover, we propose an efficient on-demand fusion query framework, FusionQuery, which consists of a query stage and a fusion stage. In the query stage, we frame the heterogeneous data query problem as a knowledge graph matching problem and present a line graph-based method to accelerate it. In the fusion stage, we develop an Expectation Maximization-style algorithm to iteratively updates data veracity and source trustworthiness. Furthermore, we design an incremental estimation method of source trustworthiness to address the lack of sufficient observations. Extensive experiments on two real-world datasets demonstrate that FusionQuery outperforms state-of-the-art data fusion methods in terms of both effectiveness and efficiency.

PVLDB Reference Format:
Junhao Zhu, Yuren Mao, Lu Chen, Congcong Ge, Ziheng Wei, and Yunjun Gao. FusionQuery: On-demand Fusion Queries over Multi-source Heterogeneous Data. PVLDB, 17(X): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/JunHao-Zhu/FusionQuery.

Proceedings of the VLDB Endowment, Vol. 17, No. X ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Localised data management systems cannot meet the needs of managing rapidly growing multi-source heterogeneous data, such as structured, semi-structured, and unstructured data from different organizations. Accordingly, centralised data management systems (e.g., data lakes [39]) have emerged and provide a proper solution for multi-source heterogeneous data management. When submitting queries in centralised data management systems, we can obtain richer information from multiple sources, which can facilitate various downstream applications, such as question answering [1, 45] and knowledge reasoning [4, 13]. However, dirty and erroneous data widely exist in centralised data management systems, which incurs serious between-source conflicts among the query results from different sources. Such conflicts make query results unreliable and confusing, significantly degrading the performance of downstream applications. Therefore, resolving between-source conflicts in the query results is one of the most important problems for centralised data management systems.

To address this problem, many methods have been proposed. However, all these methods rely on batch data fusion before query processing [8, 10, 16–18, 31, 32, 36, 37, 40, 50, 55], which traverses all data within a centralised data management system and resolves all conflicts at once. Batch data fusion-based methods face three following dilemmas. **1) Poor scalability**. Enterprises today store million-scale data (or even larger scale) in their centralised data management systems. Conducting data fusion on such a large scale data at once is time-consuming and even infeasible in practice. **2) Slow response to data updates.** In scenarios where data updates frequently, such as centralised data management systems for stock data, the prolonged batch data fusion can lead to data staleness, resulting in outdated information. **3) Binding with data matching**. Typically, data fusion is regarded as the last step of a data integration pipeline [9, 19, 34], when the schemas of different data sources have been unified [15, 29] and the records across sources referring to the same data item have been detected [21, 35, 47]. The processes before data fusion are collectively termed as (across-source) data matching. Without accurate data matching, one cannot find matched data values describing the same attribute of an entity, let alone perform effective data fusion and find out truths from candidate values.

To avoid the above dilemmas, this paper proposes the on-demand fusion query, which resolves the between-source conflicts with

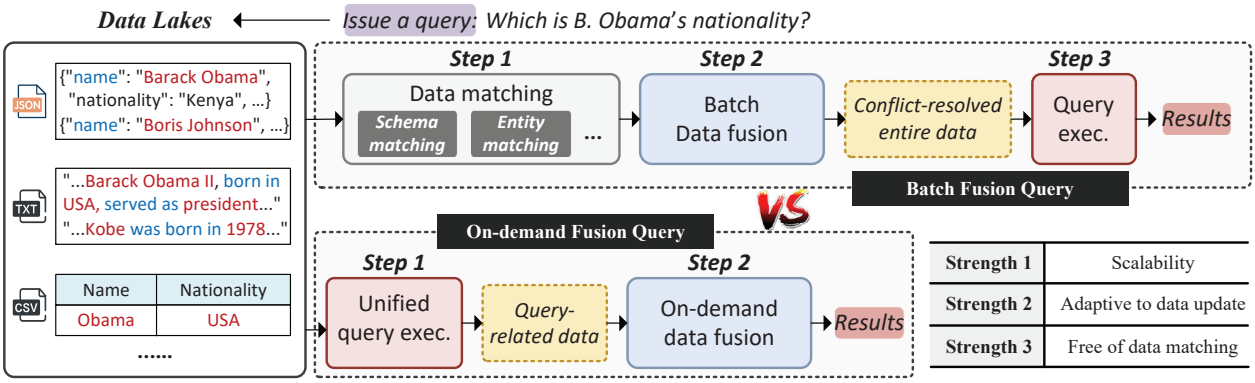


Figure 1: Comparison between batch fusion query (batch paradigm) and on-demand fusion query (On-demand paradigm).

only the query-related data and avoids accessing all the data in a centralised data management system. Figure 1 illustrates the differences in workflows between the batch paradigm and the on-demand paradigm. The advantages of the on-demand fusion query could be concluded in three aspects. **(I) Real-time data fusion.** It only utilizes the query-related data, which commonly makes up only a small proportion of the data in a centralised data management system and can be processed in real-time. **(II) Adaptive to data updates.** Both the query step and data fusion step in the on-demand fusion query can be completed in real-time; thus, it can be adaptive to frequent data updates. **(III) Free of data matching.** By matching data from various sources with users' intents (i.e., queries), on-demand fusion queries effectively sidestep the need for explicit across-source data matching. The advantages include: (1) well-defined query constraints provide clear match criteria; (2) many-to-many comparisons in across-source data matching are reduced to one-to-many comparisons, taking less time complexity. Despite the progress made by a few studies [23, 42, 51], two challenges have still existed in developing on-demand fusion queries over heterogeneous multi-source data.

Challenge I: *How to support unified queries across multi-source heterogeneous data?* Due to the data type heterogeneity (i.e., structured, semi-structured, and unstructured data) and the semantic heterogeneity (i.e., different sources involve different vocabularies) of heterogeneous multi-source data, there is still no proven solution for unified queries across multi-source heterogeneous data.

To solve data type heterogenity, we convert heterogeneous data into knowledge graphs and formulate it as a knowledge graph matching problem. Due to the richness of semantic information on both nodes and edges, knowledge graph matching is much more complex than plain graph matching. Specifically, the search space of the knowledge graph matching is exponential to the scale of the knowledge graph. Given a query graph with $|V^q|$ nodes and $|R^q|$ edges, a data graph with $|V^d|$ nodes and $|R^d|$ edges, taking the simplest solution BFS as an example, the time complexity is $O((|V^q| + |R^q|)(|V^d| + |R^d|))$ in the best case, which is infeasible in practice. To speed up knowledge graph matching, we introduce knowledge line graph transformation to decouple semantic information from graph structure, reducing the time complexity of graph matching to $O(|R^q||R^d|)$. To solve semantic heterogeneity, we focus on approximate matching in semantic information encoded by pre-trained language models, which excel in capturing semantic relations between words. For example, it can capture similar meanings for different terms, such as "spouse", "wife" and "husband"; meanwhile it can also identify different meanings for the similar words like "Apple Inc" and "Big Apple".

Challenge II: *How to perform high-quality data fusion in the on-demand setting?* The performance of data fusion is highly dependent on data-hungry probability estimations. In the on-demand setting, we only have a small amount of query-related data; thus, it is necessary to cope with the data starvation of data fusion and develop a novel on-demand data fusion method.

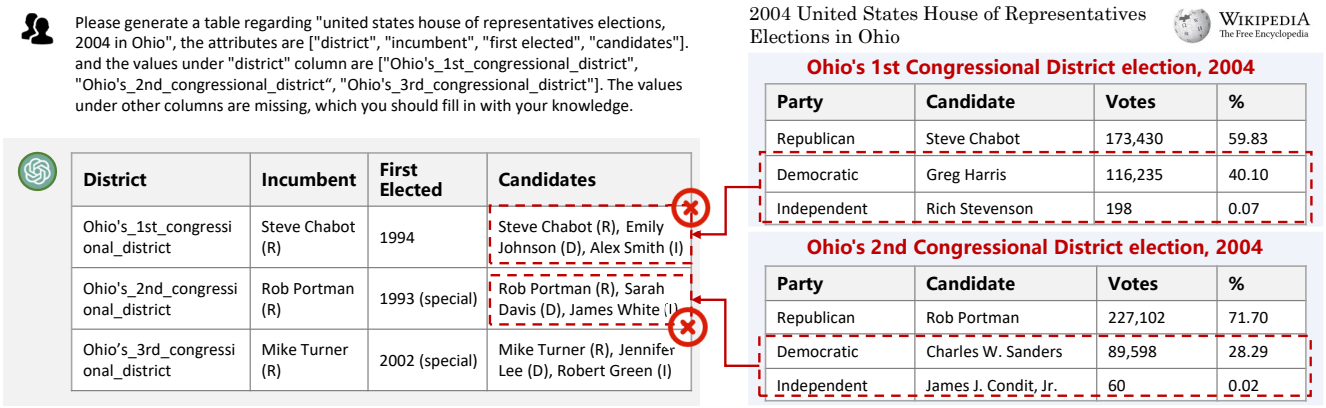
To this end, we develop an *Expectation Maximization* (EM)-style learning strategy that consists of two steps. (i) The data veracity estimation learns the probability that a data item is a correct answer to the query and (ii) the source trustworthiness estimation learns the probability that a data source provides the correct data. The two steps are repeated iteratively until convergence. Besides, considering that observed data is limited, we propose an incremental estimation for source trustworthiness based on the historical estimate and the current query results. Furthermore, to improve the effectiveness and efficiency of data fusion, we design an autonomous semantic matching threshold update mechanism to strike a balance between retrieval precision and recall.

Incorporating optimization strategies solving challenges mentioned above, we propose FusionQuery, an efficient framework for on-demand fusion queries over heterogeneous data.

1.1 Motivating Example

We reinforce the motivations for FusionQuery by illustrating two real-world application scenarios.

Case 1: Data Cleaning. Data cleaning is a pervasive challenge in data management, involving the correction of errors or the imputation of missing values in existing datasets. Retrieval-based data cleaning [2, 11] is one of the important branches of data cleaning approaches, which retrieves reliable values from external knowledge (e.g. ChatGPT, knowledge bases) as correct ones to repair or impute data. However, the data quality of external knowledge (e.g., noise or inconsistency) greatly affects the effectiveness of these approaches. In this scenario, FusionQuery could enhance retrieval-based data cleaning, which treats the data to be cleaned as a query, retrieves relevant values from multiple external sources and identifies the most reliable values among the retrieved ones. The outputs of FusionQuery can be used for data cleaning.



(a) ChatGPT for table generation

(b) Retrieved relevant data for verification

Figure 2: (a) ChatGPT generates values in tuples; (b) the on-demand fusion query retrieves relevant data from third-party sources for verification and aggregates conflicts to provide reliable data.

Case 2: Verified Generative AI. [43] Generative AI has made a significant stride, yet concerns about the accuracy and reliability of its outputs continue to grow. In particular, large language models (LLMs) sometimes make stuff up that either doesn't make sense or doesn't match the information it was given, which is known as LLM hallucinations. Figure 2 showcases a real application where a user asks ChatGPT to generate a table, but the synthesized table is unreliable due to LLM hallucinations and does harm to downstream analytical tasks. In this case, the FusionQuery emerges as an effective tool for regulating generative AI from a data management perspective. It searches relevant data from third-party sources for value verification and discovers truths from conflicts to provide more reliable data. This real-world case demonstrates the potential of FusionQuery in advancing the responsible and trustworthy use of generative AI.

1.2 Contribution

The contributions of this paper are summarized as follows:

- *A framework for on-demand fusion queries.* We propose FusionQuery, an efficient framework for on-demand fusion queries over heterogeneous multi-source data. To the best of our knowledge, this is the first work to present the on-demand fusion query over heterogeneous multi-source data.
- *An efficient knowledge graph matching framework.* We convert heterogeneous data into knowledge graphs and cast heterogeneous data queries as knowledge graph matching. Moreover, we introduce the knowledge line graph transformation to decouple the semantic information from the graph structure to speed up knowledge graph matching.
- *A convergence-guaranteed unsupervised fusion algorithm.* Following the on-demand fusion query setting, we provide new estimations for data veracity and source trustworthiness and then propose an EM-style algorithm, which iteratively updates data veracity and source trustworthiness. Also, we offer a theoretical analysis of the convergence of the iterative procedure.
- *An autonomous semantic matching threshold update mechanism.* To further boost the effectiveness and efficiency of FusionQuery, we propose an autonomous threshold update mechanism based on meta-learning, which controls the quantity and quality of query results automatically.
- *Extensive experiments.* We conduct a comprehensive experimental evaluation on the three real-world datasets. The results demonstrate the superiority of FusionQuery against the state-of-the-art batch data fusion methods in terms of effectiveness and efficiency.

2 PROBLEM STATEMENT

2.1 On-demand Fusion Query

On-demand Fusion Query (OD-FQ) aims to obtain conflict-resolved query results over multi-source heterogeneous data by only accessing to query-related data. Formally, it can be defined as follows.

Definition 1 (On-demand fusion query). *The on-demand fusion query is a kind of conjunctive query which supports on-demand data fusion.*

The on-demand data fusion is defined in Definition 2.

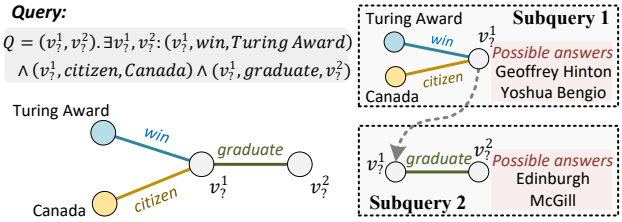
Definition 2 (On-demand data fusion). *Given an query Q issued on a set of heterogeneous sources $\mathcal{D} = \{D_i\}_{i=1}^{n}$, denoting the query-related data as $Data(Q, \mathcal{D})$, on-demand data fusion aims to resolve the between-source conflicts among $Data(Q, \mathcal{D})$ without accessing other data except $Data(Q, \mathcal{D})$.*

The formal definition of term "conflicts" is presented in Definition 4. Different from *batch fusion queries*, which resolve all conflicts within heterogeneous data in its entirety beforehand, *on-demand fusion queries* (OD-FQ) resolve conflicts only within query-related data and emit consistent values. Typically, the size of the query-related data (i.e., $Data(Q, \mathcal{D})$) is smaller than the size of $\mathcal{D}$, particularly for large-scale sources. That is, $|Data(Q, \mathcal{D})| \ll |\mathcal{D}|$.

This paper divides the procedure of an on-demand fusion query into two stages: 1) unified queries over heterogeneous data and 2) on-demand data fusion among the query results. In the following, we proceed to introduce preliminary definitions in these two stages.

2.2 Queries over Heterogeneous Data

There is still no proven solution for unified queries across heterogeneous data. To realize the unified query, this paper converts



(a) Query and its query graph (b) Reduced to two subquery graphs

Figure 3: An example illustrating how a multi-valued query is reduced to single-valued queries.

heterogeneous data into knowledge graphs and cast unified queries across heterogeneous data as a knowledge graph matching problem.

Firstly, we represent an on-demand fusion query Q by a query graph $G^q = (V^q, R^q, T^q)$, where V^q is the set of entities, R^q is the set of relations, and T^q is the set of triples of the form $\langle v^q, r^q, \hat{v}^q \rangle$. Note that there is a special entity $v_? \in V^q$ in the query graph, which represents an output variable of the query. Our framework can be extended to multi-valued queries, since a multi-valued query can be reduced into a set of single-valued queries. Figure 3 illustrates an example to demonstrate how a multi-valued query can be reduced to a set of single-valued subqueries. The example shows a query seeking information about the Canadian Turing award winner and the university where they graduated. This query is decomposed into two subqueries, each intended to be single-valued. Specifically, subquery 1 finds out the Canadian who won the Turing award, and then subquery 2 answers where the Canadian graduate.

Then, we use knowledge graphs $\mathcal{D} = \{G_i^d\}_{i=1}^n$ to represent heterogeneous data (termed data graphs). Each data graph G^d is defined as $G^d = (V^d, R^d, T^d)$, where V^d is the set of entities, R^d is the set of relations and T^d is the set of triples $\langle v^d, r^d, \hat{v}^d \rangle$. By doing so, finding a set of entities to answer the query Q over each source G^d is cast as a knowledge graph matching problem. The knowledge graph matching problem is defined as follows:

Definition 3 (Knowledge graph matching). *Given a query graph G^q and a data graph G^d, a subgraph $M \subseteq G^d$ is a match if and only if the following conditions hold:*

(1) There is one and only one semantically equivalent v^d in M for each v^q in G^q, denoted by $v^d \equiv v^q$;
(2) The relation r^d between v^d and $\hat{v}^d$ is semantically equivalent to r^q between v^q and $\hat{v}^q$ when $v^d \equiv v^q$ and $\hat{v}^d \equiv \hat{v}^q$.

Semantic equivalence is determined by the similarity of semantic information (e.g., entity names, relation names), measured by an evidence function f. Two nodes/edges are considered semantically equivalent if the similarity calculated by f exceeds a predefined τ. Notably, the evidence function can take various forms, such as the similarity of deep learning embeddings. Moreover, note that, for an undetermined entity $v_?$ with no associated semantic information, we consider it semantically equivalent to any entities so as to prevent the omission of any potential answer candidates.

Hereby, the query-related data from the data source D is a set of entities that satisfy query constraints of Q, denoted by $Data(Q, D) = \{v \in D | v \equiv v_?\}$. Naturally, the query-related data from all sources are denoted by $Data(Q, \mathcal{D}) = \{v \mid v \in Data(Q, D), D \in \mathcal{D}\}$.

2.3 On-demand Data Fusion

With the query results $Data(Q, \mathcal{D})$ obtained via knowledge graph matching, we provide a formal definition of "conflicts".

Definition 4 (Conflicts). *$Data(Q, \mathcal{D})$ is considered in conflict if there exist entities $v_{i,j} \in Data(Q, \mathcal{D})$ such that v_i is semantically unequivalent to v_j, denoted as $v_i \not\equiv v_j$.*

On-demand data fusion aims to resolve conflicts among $Data(Q, \mathcal{D})$. It requires estimating the data veracity and source trustworthiness, which are defined in Definition 5 and 6 respectively.

Definition 5 (Data veracity). *The veracity of a entity v, denoted by $Pr(v)$, is the probability that v is correct to the query.*

Definition 6 (Source trustworthiness). *The trustworthiness of a source D, denoted by $Pr(D)$, is the probability that D provides true values for any query.*

Generally, data fusion identifies correct answers via learning data veracity $Pr(v)$; meanwhile, it estimates source trustworthiness $Pr(D)$ that will be used to infer $Pr(v)$. Different from conventional data fusion, it takes as input a full-batch dataset $\mathcal{D}$ and outputs a consistent version of the dataset. Given a on-demand data fusion operation f_{cr}, it takes as input a set of query-related entities $Data(Q, \mathcal{D})$ and returns conflict-resolved results as the answer to the query Q, denoted by $Q(\mathcal{D}) = f_{cr}(Data(Q, \mathcal{D})) = \{v \in Data(Q, \mathcal{D}) \mid Pr(v) \geq \text{threshold}\}$.

3 METHODOLOGY

In this section, we propose an on-demand fusion query framework, FusionQuery, to support efficient conflict-resolving queries over multi-source heterogeneous data. We start with the framework overview, followed by details on each component.

3.1 Overview

FusionQuery consists of two stages, unified queries and on-demand data fusion, as demonstrated in Figure 4. In the first stage, unified queries over heterogeneous sources are performed to obtain raw query results. Then, with the on-demand data fusion, the between-source conflicts that exist in the raw query results are resolved, and the final answers for an on-demand fusion query are provided.

In the first stage, we cast the problem of unified queries over multi-source heterogeneous data as knowledge graph matching; furthermore, to achieve efficient knowledge graph matching, this paper proposes a novel line-graph-based knowledge graph matching method which consists of the following three steps. ① *Knowledge line graph transformation.* It converts knowledge graphs to line graphs, which decouples semantic information from graph structure to facilitate efficient knowledge graph matching. ② *Semantic matching.* It assigns match scores to node pairs based on semantic information and only keeps those with match scores above a certain semantic matching threshold as semantically aligned nodes. ③ *Structure matching.* It eliminates nodes that violate the graph isomorphism principle from semantically aligned nodes. After the above steps, the remaining nodes form the candidate answers, which satisfy both the semantic and structural requirements of the query graph.

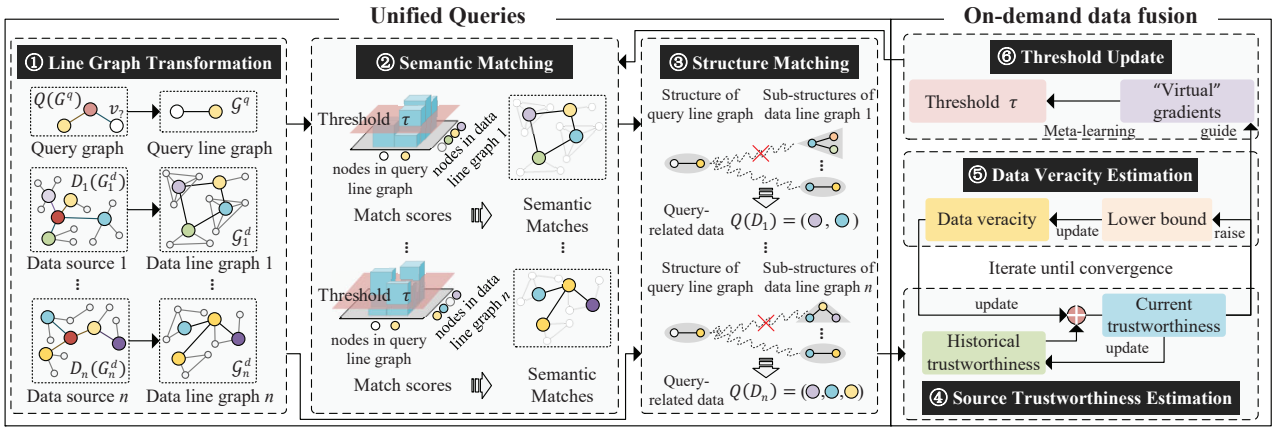


Figure 4: The framework of FusionQuery for on-demand fusion queries.

On-demand data fusion aims to filter out the conflicts by the veracity scores of the raw query results. This stage also comprises three steps. ④ *Source trustworthiness estimation.* It incrementally calculates the source trustworthiness based on historical trustworthiness and current observations. ⑤ *Data veracity estimation.* It computes the data veracity by providing the lower bound of the data veracity. The estimations of source trustworthiness and data veracity update iteratively until convergence. ⑥ *Threshold update.* This step adjusts the semantic matching threshold automatically using meta-learning [3], which strikes a balance between retrieval precision and recall by controlling the quantity and quality of candidate answers.

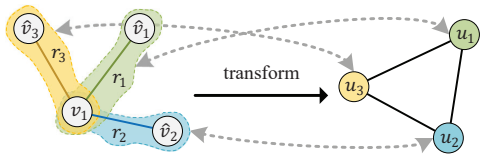
In the following, we present the detailed techniques and steps in these two stages, respectively.

3.2 Unified Queries

To enable unified queries, we convert the query and heterogeneous data to knowledge graphs by adopting MRGC [22] and formulate the problem of unified queries over heterogeneous data as knowledge graph matching. Conceptually, we can view the knowledge graph matching as an implicit join operation between two tables: one recording the *semantic matches* in the data graphs concerning the query graph and another recording *structural matches* with the query graph. In this analogy, the knowledge graph matching essentially performs an inner join between two tables and returns nodes that appear in both tables.

FusionQuery adopts the line graph transformation to divide knowledge graph matching into two independent sub-problems: node-level semantic matching and graph-level structure matching, which facilitates the fast construction of "tables" of semantic matches and structural matches. The details of the knowledge line graph transformation, semantic matching, and structure matching are introduced as follows.

3.2.1 Knowledge Line Graph Transformation. In this step, the query graph and data graphs are converted to the query line graph and data line graphs respectively. The knowledge line graph [20] is defined in Definition 7.



(a) Original knowledge graph (b) Knowledge line graph

Figure 5: Example of knowledge line graph transformation.

Definition 7 (Knowledge line graph). *Given a knowledge graph G and the corresponding knowledge line graph $\mathcal{G}$, the knowledge line graph transformation is a mapping such that*

(1) Each node of $\mathcal{G}$ represents a triple of G;
(2) Two nodes of $\mathcal{G}$ are adjacent if and only if their corresponding edges share a common endpoint in G.

Example 1. *Taking Figure 5 as an example, triples in shades of different colors are mapped to corresponding nodes in the knowledge line graph, such as $\langle v_1, r_1, \hat{v}_1\rangle \rightarrow u_1$ and $\langle v_1, r_2, \hat{v}_2\rangle \rightarrow u_2$. Since $\langle v_1, r_1, \hat{v}_1\rangle$ and $\langle v_1, r_2, \hat{v}_2\rangle$ share the common endpoint v_1, nodes u_1 and u_2 are connected in the line graph. Similarly, the node pairs (u_1, u_3) and (u_2, u_3) are also connected.*

With knowledge line graph transformation, graph structure and semantic information are decoupled, and we can process them independently. For ease of representation, nodes in knowledge line graphs are denoted by $\langle v, r, \hat{v}\rangle$ in the following sections.

3.2.2 Semantic Matching. Semantic matching aims to find semantically aligned nodes between query line graphs and data line graphs. This paper encodes semantic information on triples using pre-trained language models (e.g., BERT [14], SBERT [41]), which has demonstrated strong capabilities in handling semantic heterogeneity. For any node $\langle v^d, r^d, \hat{v}^d\rangle$ in a data line graph, we represent its semantic embedding as $\langle E^d_v, E^d_r, E^d_{\hat{v}}\rangle$. Similarly, for any node $\langle v^q, r^q, \hat{v}^q\rangle$ in a query line graph, we denote the semantic embedding by $\langle E^q_v, E^q_r, E^q_{\hat{v}}\rangle$. Taking the query graph $G^q = \langle v^q, r^q, v_?\rangle$ as an example, $?v$ is an object to be returned by the query, while v^q and r^q are query criteria. For a potentially aligned entity $\hat{v}^d$ in the data graph, $v_?$ matches $\hat{v}^d$ as long as the query criteria v^q and r^q match v^d and r^d, respectively. Hence, the match score of $\hat{v}^d$,

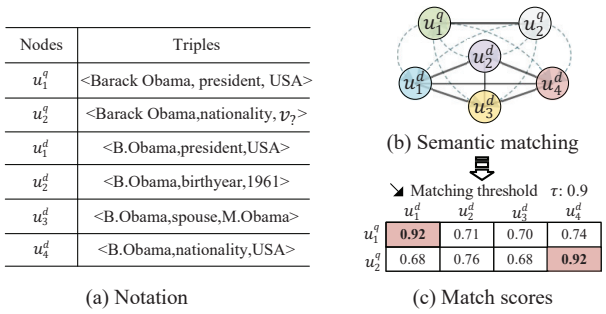


Figure 6: A running example of semantic matching.

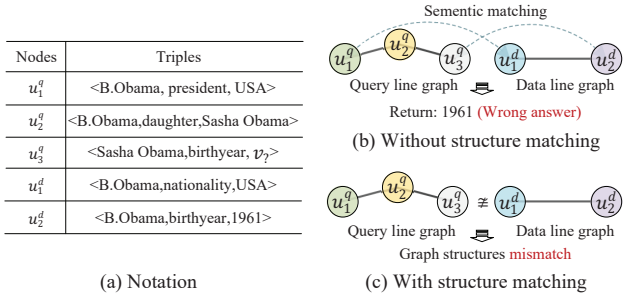


Figure 7: A toy example of why structure matching matters.

denoted by $\delta(\hat{v}^d)$, can be calculated as:

$$\delta(\hat{v}^d) = \frac{sim(E_v^q, E_v^d) + sim(E_r^q, E_r^d)}{2} \tag{1}$$

where $sim(*,*)$ can be any similarity measures, such as cosine similarity. The design of the match score is based on the intuition that if v^d and r^d match the query criteria v^q and r^q, $\hat{v}^d$ is more likely to be an answer to $v_?$. Data in the data source D_i whose match scores are no less than a semantic matching threshold τ are regarded semantically aligned, which forms a set of candidates, i.e., $D_i[G^q] = \{v^d \in D_i | \delta(v^d) \geq \tau\}$. The threshold τ significantly impacts the quality and quantity of query results. The too-low threshold would introduce noisy answers into the query results, while the too-high threshold would reject correct answers. Setting τ manually is quite tricky. To address it, we introduce an automatic semantic matching threshold update mechanism in Section 3.3.4.

Example 2. *The query "which is the nationality of the American president Barack Obama" can be transformed into a query line graph with nodes u_1^q and u_2^q, as shown in Figure 6. Node u_1^q is a triple that represents a query criterion (i.e., Obama is a president of USA), while u_2^q represents the expected query result (i.e., the nationality of Obama). The data being queried is transformed into a data line graph with nodes u_1^d, u_2^d, u_3^d and u_4^d, where each node represents a piece of information about Obama. Semantic matching calculates match scores for each node pairs based on Eq.1. Node pairs whose match scores are no less than the pre-set matching threshold τ are regarded as node alignments in terms of semantics. In this example, if we set τ to 0.9, we find that the scores of the node pairs (u_1^q, u_1^d) and (u_2^q, u_4^d) are higher than τ. Thus, they are considered semantically aligned.*

Since the numbers of nodes in the query line graph and the data line graph are $|R^q|$ and $|R^d|$, respectively, the naive semantic matching takes $O(|R^q||R^d|)$ time complexity in the worst case. To accelerate it, we do the following optimizations. Considering that the kinds of relations are much less than the kinds of entities, in order to reduce the search space, triples of the data graph are assigned to different clusters based on the kinds of relations. Semantic matching takes two steps: it first finds the most similar relation in the data graph to the one in the query graph and then finds all entities that meet the condition within the corresponding cluster. Moreover, an efficient similarity search tool, Faiss [28], is also leveraged to accelerate embedding similarity calculation.

3.2.3 Structure Matching. In addition to finding semantic matches at the node level, knowledge graph matching also must ensure that those matches conform to the graph structure. Structure matching is therefore a critical step in the process of knowledge graph matching. It aims to eliminate semantically aligned nodes that violate graph isomorphism principles.

Example 3. *Figure 7 illustrates an example of how structure matching is essential to ensure the accuracy of query results. Consider a query that seeks information about the birth year of Sasha Obama, daughter of the American President Barack Obama. The corresponding query line graph consists of three nodes: u_1^q (representing the triple ⟨ B.Obama, president, USA⟩), u_2^q (representing the triple ⟨ B.Obama, daughter, Sasha Obama ⟩), and u_3^q (representing the triple ⟨ Sasha Obama, birthyear, $v_?$⟩). The data line graph contains two nodes u_1^d and u_2^d, which match u_1^q and u_3^q, respectively, due to their literal similarity. Without structure matching, the query result will likely be incorrect, returning the birth year of Barack Obama (1961) rather than that of his daughter. However, with structure matching, this error can be avoided by identifying the mismatch between the graph structures. Hence, structure matching is a crucial step in ensuring the accuracy of knowledge graph queries.*

Guaranteed by Whitney isomorphism theorem (see Lemma 1), in our structure matching, line graph isomorphism is equivalent to graph isomorphism because our query graphs are acyclic graphs.

Lemma 1 (Whitney isomorphism theorem). *Two connected graphs are isomorphic if and only if their line graphs are isomorphic, except in the case of the ring graph and the star graph.*

Besides, any non-attributed graph matching methods can be adopted to solve structure matching in our framework. Furthermore, a lot of efficient non-attributed graph matching methods have been proposed, such as graph matching with well-designed data structures [5, 6, 24], graph matching on parallel platforms [30, 52, 53], graph matching on specific hardware [27, 46, 54], etc. Filtering entities in candidate set $D_i[G^q]$ that violate graph isomorphism, the rest of entities forms the query results from source D_i, denoted by $Data(Q, D_i)$.

3.3 On-demand Data Fusion

After obtaining query results $Data(Q, \mathcal{D})$, on-demand data fusion resolves conflicts in $Data(Q, \mathcal{D})$ based on data veracity. In the multi-source setting, the estimation of data veracity relies on the source trustworthiness which requires to estimate as well. Thus, the estimation of data veracity and source trustworthiness are closely intertwined. To accurately estimate data veracity, this paper regards source trustworthiness as a latent variable and designs an

Expectation-Maximization (EM) algorithm to estimate data veracity. The workflow of this EM algorithm includes: (1) initialization of data veracity and source trustworthiness (see Section 3.3.1); (2) iterative estimations of data veracity and source trustworthiness (see Section 3.3.2 and 3.3.3); (3) threshold update (see Section 3.3.4). The overall workflow is presented in Algorithm 1, and the details of each component are introduced as follows.

3.3.1 Initialization. Before iterative data veracity and source trustworthiness estimations, we have to initialize the source trustworthiness and data veracity. Previous approaches choose to assign the same initial veracity scores for all entities and the same trustworthiness scores for all data sources, which prevents data fusion methods from efficient learning. To address this issue, this paper leverages intrinsic features (i.e., *match scores* and *null value proportions*) to learn data veracity and source trustworthiness effectively and efficiently. Specifically, match scores indicate the degree of alignment between an entity and the user's query. Higher match scores suggest a stronger likelihood that the attribute value associated with the entity is the correct answer to the query. Null value proportions indicate the information integrity of different sources. Taking the null value proportion as the initial trustworthiness score of sources makes estimations converge much quicker.

3.3.2 Data Veracity Estimation. We model the data veracity estimation as *maximum likelihood estimation* (MLE) since emitted query results possess high veracity as the observable values. For an entity v, its veracity score is estimated as:

$$\log Pr(v) = \arg\max_{v,D} \log \sum_{D \in \mathcal{D}} Pr(v, D). \tag{2}$$

where $Pr(v, D)$ is a joint distribution of data veracity and source trustworthiness. Maximizing Eq. 2 by searching $Pr(v)$ is infeasible due to the extremely large search space resulting from the relationship between $Pr(v)$ and hidden variables $Pr(D)$. To address this, we derive the lower bound of the data veracity $Pr(v)$ and change it in every iteration. Specifically, we calculate data veracity as:

$$\begin{aligned} \log Pr(v) &= \sum_{D \in \mathcal{D}} Pr(D|v) \log \frac{Pr(v, D)}{Pr(D|v)} \\ &= \sum_{D \in \mathcal{D}} Pr(D|v) \log \frac{Pr(v|D)Pr(D)}{Pr(D|v)} \end{aligned} \tag{3}$$

where $Pr(D|v)$ and $Pr(v|D)$ are two conditional probabilities. The concrete introduction of $Pr(D|v)$ and $Pr(v|D)$, including their meanings and calculations, will be explained later. Besides, the proof of the lower bound and theoretical analysis of its monotonicity will be discussed further in Section 4.

The conditional probability $Pr(v|D)$ represents the veracity score of the entity v when the source D is reliable. Regarding $Pr(v|D)$, consider two circumstances where the data source D provides (i.e., emits) the entity v as an answer or not. $Pr(v|D)$ is designed as follows.

$$Pr(v|D) = \begin{cases} Pr(D) & D \text{ provides } v, \\ 1 - Pr(D) & \text{otherwise.} \end{cases} \tag{4}$$

The intuition is grounded in the premise of the source D being reliable and definitions of $Pr(D)$ and $Pr(v|D)$ firstly proposed in [36]. Specifically, we accept v as a correct answer by the probability of D providing correct answers (i.e. $Pr(D)$) if D provides v; otherwise, we regard D makes a mistake and reject it by probability $1 - Pr(D)$.

To ensure that data veracity falls within the range $[0, 1]$, $Pr(v)$ should be normalized. However, normalization methods such as softmax can smooth out differences in $Pr(v)$ between different entities, which could have a negative impact on the performance of data fusion. Thus, inspired by the previous study [50], $Pr(v)$ is transformed to $-\log(1 - Pr(v))$ for normalization. After transformation, $Pr(v)$ falls between 0 and $+\infty$, where larger values indicate higher veracity. Moreover, we also utilize the Gumbel-Softmax [26] to normalize $Pr(v)$, which can preserve the differences as much as possible. In addition, an answer will be more reliable if the majority of sources provide it for a query. Hence, we introduce a parameter called the vote count, denoted by ω_v, to adjust data veracity. The vote count represents the number of sources that provides the entity v. To sum up, the data veracity is normalized as follows:

$$Pr(v) \leftarrow \frac{\exp\left(\frac{-\omega_v \cdot \log(1-Pr(v))}{z}\right)}{\sum_{\bar{v} \in Data(Q,\mathcal{D})} \exp\left(\frac{-\omega(\bar{v}) \cdot \log(1-Pr(\bar{v}))}{z}\right)} \tag{5}$$

where z is a temperature scaling parameter. As z approaches 0, the softmax computation approaches the argmax; while as z approaches $+\infty$, the distribution of normalized data veracity becomes identical to the uniform distribution.

3.3.3 Source Trustworthiness Estimation. As a latent variable to infer data veracity, in this section, we introduce how to estimate source trustworthiness $Pr(D)$. According to the *Law of Total Probability*, the estimation of $Pr(D)$ could be expanded as:

$$Pr(D) = \sum_{v \in Data(Q,D)} Pr(D|v)Pr(v) \tag{6}$$

where the conditional probability $Pr(D|v)$ represents the trustworthiness score of the source D when the entity v is a true answer for the query.

To estimate $Pr(D|v)$, we re-define source trustworthiness in the context of true positive and false positive. Each source emits entities as true or false predictions for the correct query answers, and emitted entities are taken as the positive for "the prediction" because these entities possess high veracity. Therefore, the source trustworthiness is equivalent to the probability that positive entities emitted from the source are truly correct. It is estimated by $\frac{TP}{TP+FP}$, where TP denotes the number of correct answers provided and FP represents the number of false answers provided.

However, under the on-demand fusion query setting, we do not have access to the entire data during the fusion stage. As a result, we cannot calculate the exact TP and FP of all the sources, let alone the exact estimation of the source trustworthiness. Despite this, we can still obtain the historical source trustworthiness and query-related data for the correct query. Thus, we propose an incremental estimation of $Pr(D|v)$ based on historical trustworthiness and current query-related data.

Assume that the number of entities provided by the data source D for all historical queries is $\mathcal{H}$, and query Q-related data from the source D is denoted by $Data(Q, D)$. With prior knowledge that the entity $v \in Data(Q, D)$ is one of correct answers to the query Q, other entities in $Data(Q, D)$ whose veracity scores are no less than v can also be taken as correct. We denote the set of correct answers

as $D_v[Q] = \{\bar{v} \in Data(Q,D) | Pr(\bar{v}) \geq Pr(v)\}$. Then, $Pr(D|v)$ can be estimated as:

$$Pr(D|v) = \frac{\mathcal{H} \cdot Pr^h(D)}{\mathcal{H} + |Data(Q,D)|} + \frac{\sum_{\bar{v} \in D_v[Q]} Pr(\bar{v})}{\mathcal{H} + |Data(Q,D)|} \tag{7}$$

where $Pr^h(D)$ represents the latest historical estimate of source trustworthiness. Since the source trustworthiness represents the probability that a source provides correct answers, $\mathcal{H} \cdot Pr^h(D)$ can represent the expected number of correct answers provided by the source D for historical queries. Similarly, as $Pr(v)$ represents the probability that an entity is a correct answer, $\sum_{\bar{v} \in D_v[Q]} Pr(\bar{v})$ represents the unbiased estimate for the number of correct answers provided by D in the current query. Therefore, $\mathcal{H} \cdot Pr^h(D) + \sum_{\bar{v} \in D_v[Q]} Pr(\bar{v})$ estimates the number of true positives (TP) in D and the denominator $\mathcal{H} + |Data(Q,D)|$ represents the number of positive entities provided by D, namely $TP + FP$, to some extent.

3.3.4 Threshold Update. This paper finds that semantic matching threshold τ matters the trustworthiness scores of data sources. Furthermore, τ is source-wise, and setting the value of τ depends on the source trustworthiness. Based on source trustworthiness estimates, this paper proposes an autonomous τ update mechanism, which learns to update τ by means of a meta-learning paradigm. Meta-learning [3], which follows the thoughts of *learning-to-learn*, can automatically update hyperparameters based on the gradient of the learning objective. This paper set the learning objective to optimize the source trustworthiness w.r.t τ, and automatically update τ according to the gradient $\nabla_\tau Pr(D)$.

However, computing the gradient $\nabla_\tau Pr(D)$ is challenging for the estimation of $Pr(D)$ is dependent on $Pr(v)$ rather than directly dependent on τ. Consequently, computing the gradient $\nabla_\tau Pr(D)$ is not a straightforward task. To overcome this challenge, we introduce a "virtual" gradient, which links $Pr(D)$ and τ by means of a straightforward transformation applied to $Pr(v)$.

As discussed in Section 3.2, each entity v has to satisfy the condition $Pr(v) \geq \tau$ to be considered as an answer to the query. Applying a simple transformation, we can derive:

$$Pr(v) \geq \tau \Leftrightarrow Pr(v) = \tau + \epsilon_v (\epsilon_v \geq 0) \tag{8}$$

where ϵ_v is a positive constant for each entity v. This equation creates a "virtual" connection between $Pr(D)$ and τ. Substituting Eq. 8 into Eq. 6, we can obtain an equation with respect to $Pr(D)$ and τ. Further, we can derive the "virtual" meta-gradient of trustworthiness with respect to τ, given by:

$$\begin{aligned} \nabla_\tau Pr(D) &= \frac{\partial \sum_{v \in Data(Q,D)} Pr(D|v) Pr(v)}{\partial \tau} \\ &= |Data(Q,D)| + \sum_{v \in Data(Q,D)} \frac{Pr(v) \cdot |D_v[Q]|}{\mathcal{H} + |Data(Q,D)|} \end{aligned} \tag{9}$$

Having derived the "virtual" meta-gradient $\nabla_\tau Pr(D)$, τ can be updated via back-propagation. Here, we leverage a gradient descent algorithm [7], which is widely used in the deep learning domain. The update of τ is presented as:

$$\tau = \tau - \theta sgn(\Delta Pr(D)) \cdot \nabla_\tau Pr(D) \tag{10}$$

where $\Delta Pr(D)$ is the change of $Pr(D)$ during iterations, $sgn(\Delta Pr(D))$ is a sign function that indicates the direction of the change (i.e., increase or decrease), and θ is a hyperparameter that controls the speed of parameter update, also known as the learning rate.

Algorithm 1: FusionQuery

Input: a query Q in form of a graph G^q, a data source set $\mathcal{D} = \{D_i\}$
Output: Answers to the query with veracity $\{(v, Pr(v))\}$

```
1  initialize the query-related data Data(Q, D) ← ∅
2  for each D_i in D do
3      convert D_i to knowledge graphs G_i^d using MRGC
4      transform G_i^d into knowledge line graph G_i^d
5  transform G^q into knowledge line graph G^q
6  for each G_i^d in data graph set do
7      apply semantic matching to G_i^d with the semantic
         matching threshold τ
8      apply structure matching to G_i^d to generate
         query-related data Data(Q, D_i)
9      add Data(Q, D_i) to Data(Q, D)
10 initialize a trustworthiness score for each source
11 initialize data veracity for each entity in Data(Q, D)
12 while divergence do
13     for each D in D do
14         update Pr(D|v) ⟵ Eq. 7
15         update Pr(D) ⟵ Eq. 6
16     for each v in Data(Q, D) do
17         update Pr(v) ⟵ Eq. 3
18         normalize Pr(v) ⟵ Eq. 5
19 update threshold τ based on Eq. 9 and Eq. 10
20 return Query answers (v, Pr(v))
```

Automatic adjustment of τ ensures that when source trustworthiness decreases, the threshold τ increases to improve the quality of candidate answers generated in the unified queries. When the source trustworthiness increases, the threshold τ is lowered to maintain high recall.

In conclusion, the whole workflow of FusionQuery is presented in Algorithm 1. It takes as inputs a query Q in the form of a graph G^q, a set of data source $\mathcal{D} = \{D_i\}$, and outputs the query answers with data veracity $\{(v, Pr(v))\}$. FusionQuery initializes query-related data $Data(Q, \mathcal{D})$ as empty (lines 1). Then, it converts data sources to knowledge graphs and transforms them into knowledge line graphs (lines 2-4). Next, the query graph is also transformed into the knowledge line graph (line 5). In the sequel, for each data line graph, the algorithm applies semantic and structure matching to generate query-related data from each source and add them to $Data(Q, \mathcal{D})$ (lines 6-9). Then, source trustworthiness and data veracity are initialized at the beginning of iterative estimations (lines 10-11). Thereafter, a while-loop is performed to estimate data veracity and source trustworthiness (lines 12-18). For source trustworthiness estimation, it calculates the conditional probability $Pr(D|v)$ and updates source trustworthiness based on Eq. 7 and Eq. 6 respectively (lines 13-15). For data veracity estimation, it calculates $Pr(v)$ based on Eq. 3 and normalizes $Pr(v)$ according to Eq. 5 (lines 16-18). Finally, the algorithm updates the semantic

matching threshold τ based on Eq. 9 and Eq. 10 (line 19), and returns entities with veracity scores as answers to the query (line 20).

4 THEORETICAL ANALYSIS

In this section, we provide the theoretical analysis of the convergence of the fusion algorithm and analyze the time complexity of the entire workflow.

Convergence Analysis. Since data veracity can be regarded as an injective function w.r.t source trustworthiness and vice versa, both converge as long as one of them is proven to be convergent. We will discuss the convergence of data veracity $Pr(v)$ as an example.

Considering the data veracity $Pr(v)$, it is convergent if and only if it is bounded and monotonic according to the *Monotone convergence theorem*. The data veracity is bounded according to its definition, as stated in Section 2. The following lemma establishes the monotonicity of $Pr(v)$.

Lemma 2 (Monotonicity of data veracity). *If the majority of data sources provide the entity v as an answer, $Pr(v)$ is monotonically increasing since the lower bound of $Pr(v)$ is monotonically increasing in iterations.*

To prove Lemma 2, we start with deriving the lower bound of the data veracity $Pr(v)$, as stated in Lemma 3.

Lemma 3 (Lower bound of the data veracity). *Given a random entity v with the veracity score $Pr(v)$, and a set of data sources $\mathcal{D}$, we have the following lower bound for* $\log Pr(v)$*:*

$$\log Pr(v) \geq \sum_{D \in \mathcal{D}} Pr(D|v) \log \frac{Pr(v, D)}{Pr(D|v)}.$$

Proof. Here, we regard $Pr(D|v)$ as a probability distribution concerning the source D. Therefore, $\sum_D Pr(D|v) \log \frac{Pr(v,D)}{Pr(D|v)}$ could be interpreted as an expected value with respect to $\frac{Pr(v,D)}{Pr(D|v)}$. Since the function log is a concave function, the lower bound of the data veracity $\log Pr(v)$ could be derived based on *Jensen's inequality*. Formally, we have

$$\begin{aligned}\log Pr(v) &= \log \sum_{D \in \mathcal{D}} Pr(v, D) = \log \sum_{D \in \mathcal{D}} Pr(D|v) \frac{Pr(v, D)}{Pr(D|v)} \\ &\geq \sum_{D \in \mathcal{D}} Pr(D|v) \log \frac{Pr(v, D)}{Pr(D|v)}.\end{aligned}$$

which completes the proof. □

If the lower bound of data veracity could be proved to increase monotonically in iterations, the monotonicity of the data veracity $Pr(v)$ (i.e., Lemma 2) could also be verified naturally.

Corollary 1 (Monotonicity of the lower bound). *If the majority of data sources provide the entity v as an answer, the lower bound of $Pr(v)$ is monotonically increasing in iterations, otherwise, it is monotonically decreasing.*

Proof. Let $Pr^k(v)$ denote the veracity score at the k-th iteration. Each iteration follows the procedure: $Pr^k(v) \rightarrow Pr(D|v) \rightarrow Pr(D) \rightarrow Pr^{k+1}(v)$. According to Eq. 7 and Eq. 6, it is obvious to derive that $Pr(D|v) \propto Pr(v)$ and $Pr(D) \propto Pr(D|v)$. Substituting Eq.4 into Eq.3, we can derive that $Pr(v) \propto Pr^2(D)$ (the source D provides v) or $Pr(v) \propto Pr(D)(1 - Pr(D))$ (the source D opposes v). That is, $Pr(v)$ increases with the growth of $Pr(D|v)$ if most sources provide the entity v as an answer; otherwise, $Pr(v)$ drops, which completes the proof. □

Table 1: Statistics of the datasets used in experiments.

Datasets	Format	#num.	#ent (avg.)	#rel (avg.)	Query
Movie	JSON (J)	4	19,701	45,790	210
	KG (K)	5	100,229	264,709	
	CSV (C)	4	70,276	184,657	
Book	JSON (J)	3	3,392	2,824	100
	CSV (C)	3	2,547	1,812	
	XML (X)	4	2,054	1,509	
Flight	CSV (C)	10	48,672	100,835	260
	JSON (J)	10	41,939	89,339	
Stock	CSV (C)	10	7,799	11,169	100
	JSON (J)	10	7,759	10,619	

Time Complexity. The time complexity of knowledge graph matching in our framework depends on the specific non-attributed graph matching methods used. Here, we consider the VF2 algorithm [12], which is the most classical graph isomorphism solver. Assume that the number of nodes in a query line graph and a data line graph is $|R^q|$ and $|R^d|$ respectively, while the number of kinds of relations in the data graph is p. Semantic matching costs $O(p|R^q| + \frac{1}{p}|R^q||R^d|)$ time. Assume that semantic matching returns a subgraph with $e(e \ll |R^d|)$ nodes as a semantic match. VF2 needs $O(e|R^q|)$ to perform the structure matching in the best case. To sum up, it needs $O(p|R^q| + \frac{1}{p}|R^q||R^d| + e|R^q|)$ to obtain query-related data in a data source in the best case. Thus, given n sources, it takes $O(np|R^q| + \frac{n}{p}|R^q||R^d| + ne|R^q|)$ to collect candidate answers, denoted by $O(\frac{n}{p}|R^q||R^d|)$, in the best case.

Next, we focus on analyzing the time complexity of our fusion algorithm. Suppose that there are n data sources and m entities related to a query Q, and each source could provide $\frac{m}{n}$ entities on average. In each iteration, for each source D and each entity v, it needs $O(\frac{m}{n}+1)$ to calculate conditional probabilities $Pr(D|v)$. Thus, for all n sources and m entities, it takes $O(m^2 + mn)$ to calculate $Pr(D|v)$. Next, we assign the source trustworthiness $Pr(D)$ to all data sources, which takes $O(n)$ cost. In summary, $O(m^2+mn+n)$ is required to complete the source trustworthiness estimation for each iteration. To estimate data veracity, we need to calculate $Pr(v|D)$ and $Pr(v)$, which takes $O(mn)$ and $O(m)$, respectively. Normalizing $Pr(v)$ requires an additional $O(m)$ time cost. Thus, it needs $O(mn+2m)$ to estimate data veracity for each iteration. Assume that there are I iterations before convergence. The time complexity of our fusion algorithm is $O(Im^2n+2Imn+2Im+In)$, denoted by $O(Im^2n)$.

5 EXPERIMENTS

In this section, we compare our FusionQuery with several batch and on-demand data fusion baselines respectively. Additionally, we conduct extensive experiments to evaluate the effectiveness and efficiency of the techniques used in our method.

Table 2: Comparison with on-demand and batch data fusion baselines w.r.t. effectiveness and efficiency.

Datasets	Types	On-demand data fusion baselines										Batch data fusion baselines										Ours	
		OL-MV		OL-TF		OL-LTM		OL-DART		OL-CASE		QS-MV		QS-TF		QS-LTM		QS-DART		QS-CASE		FusionQuery	
		F1	Time	F1	Time	F1	Time	F1	Time	F1	Time	F1	Time	F1	Time	F1	Time	F1	Time	F1	Time	F1	Time
Movie	J/K	0.21	0.07	31.7	36.5	13.2	55.1	8.65	2.85	22.6	4.92	1.77	1399	37.1	9717	41.4	1995	43.2	3809	40.4	4900	**51.3**	**2.64**
	J/C	0.11	0.13	24.1	38.5	8.01	91.7	4.85	4.32	14.2	5.06	1.72	41.9	41.9	7214	42.9	1884	45.9	3246	42.3	3981	**54.0**	**2.36**
	K/C	0.09	0.18	24.2	51.3	13.4	118.0	4.30	6.49	14.9	5.99	3.68	1397	37.8	2199	41.2	1576	37.6	2027	39.4	1699	**48.3**	**4.40**
	J/K/C	0.13	0.19	44.7	67.5	7.71	201.1	5.76	9.57	21.7	**8.80**	1.79	1400	36.6	11225	40.8	2346	41.5	5151	42.1	5480	**54.3**	10.8
Book	J/C	1.13	0.01	38.3	1.98	18.5	4.06	22.5	**0.30**	24.7	1.84	7.20	34.8	40.2	1017	42.4	195.3	35.2	165.0	41.3	376.6	**62.4**	0.47
	J/X	0.17	0.01	35.5	2.07	11.1	6.32	26.2	**0.35**	24.7	1.84	8.89	34.9	35.5	1070	35.6	277.7	36.1	200.1	35.5	377.8	**60.0**	0.56
	C/X	0.83	0.01	40.2	0.93	14.0	3.53	32.9	**0.25**	21.2	1.66	10.0	34.2	43.0	1033	44.1	232.6	42.6	201.4	40.3	811.0	**59.6**	0.38
	J/C/X	0.13	0.01	42.9	2.51	8.76	8.75	27.2	**0.51**	40.8	1.96	7.36	35.4	37.3	2304	41.0	413.2	40.4	394.1	40.3	811.0	**60.3**	1.07
Flight	C/J	0.06	0.32	27.3	6049	21.3	1846	72.3	**20.2**	12.0	54.5	67.1	1445	-	-	79.1	14786	**80.1**	73380	-	-	72.9	109.9
Stock	C/J	55.3	0.01	68.4	2.30	28.0	9.25	64.8	**0.33**	64.8	2.27	21.1	65.4	20.6	5034	16.7	431.0	19.2	1337	17.4	1366	**71.6**	0.36

[1] The symbol "-" denotes that the method fails to finish within 1 day.

5.1 Experimental Settings

Datasets. The experiments are conducted on three real-world benchmark datasets [17, 33, 50]. Table 1 presents the detailed statistics of the datasets. (i) The *Movie* dataset contains movie data collected from 13 sources. In experiments, 210 queries are executed on the *Movie* dataset. (ii) The *Book* dataset contains book data from 10 sources. In experiments, we execute 100 queries on the *Book* dataset. (iii) The *Flight* dataset collects information on over 1200 flights from 20 sources. In experiments, 260 queries are issued on the *Flight* dataset. (iv) The *Stock* dataset collects trading data of 1000 stock symbols from 20 sources. In experiments, 100 queries are issued over it.

Baselines. In order to demonstrate the superiority of our proposed FusionQuery, we compare it with the state-of-the-art data fusion method as well as other baseline methods. We summarize these baselines below.

- **MajorityVoter** takes the data as reliable if most data sources provide it, which is abbreviated as MV.
- **TruthFinder** [50] is an iterative method that utilizes the interdependency between sources and provided data to infer the trustworthiness of sources and the correctness of data. TruthFinder is abbreviated as TF.
- **LTM** [55] is a probabilistic data fusion method, which constructs a graphical model and uses Gibbs sampling to determine the source trustworthiness and the data veracity scores.
- **DART** [36] incorporates the domain expertise score and the confidence score definition for truth determination. However, we do not consider domain features in the experiments, which are not available under our problem definition.
- **CASE** [37] constructs a heterogeneous network to model relationships between sources and data, and then learns the representation of sources and data automatically from the interactions between sources and data. In our implementation, we use the unsupervised variant, as there is no labeled data available.

Implementation details. For all baselines, we carefully tune the parameters according to the original papers in our settings. All the methods are implemented in Python 3.8. In our method, we use SBERT [41] as the backbone structure of the semantic matching step due to its superiority, and the dimension of the embedding vectors is set to 768. We use the VF2 algorithm to perform structure matching for its simple implementation. The temperature parameter z is set to 0.5, and the learning rate θ is set to 3e-5. The number of entities for historical queries $\mathcal{H}$ is initialized with 50. All the experiments are conducted on a server with an Intel Xeon(R) E5-2640, 2.40GHz CPU, an NVIDIA 1080ti 12G GPU, and 128GB memory.

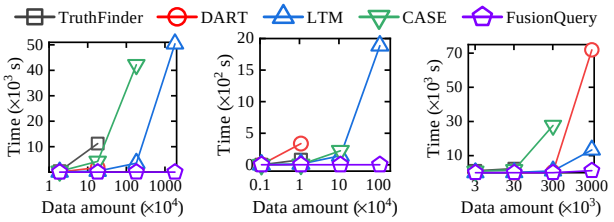
Evaluation metrics. To evaluate effectiveness, following previous studies [36, 55], we use the F1 score (F1) as the evaluation metric on data fusion results, which is the harmonic mean of precision (P) and recall (R) computed as $F1 = \frac{2\times P\times R}{P+R}$. In addition, we use runtime (in seconds) as an evaluation metric to verify efficiency.

5.2 Comparison with On-demand Baselines

Setting. We compare FusionQuery with on-demand data fusion baselines. For a fair comparison, we replace our data fusion method with five data fusion baselines in the FusionQuery framework. Thereby, we obtain five corresponding variants to evaluate their on-demand fusion query performance, namely, online MajorityVoter (OL-MV), online TruthFinder (OL-TF), online LTM (OL-LTM), online DART (OL-DART), and online CASE (OL-CASE). Table 2 summarizes the data fusion performance (i.e., F1 score and total fusion time) of FusionQuery and baselines on three datasets.

Effectiveness. Table 2 indicates that FusionQuery outperforms all competitors on all datasets. Statistically, it achieves almost 1.5× and 2× higher F1-score than TruthFinder and CASE, and achieves much better performance than other baselines.

MV achieves poor performance on all datasets. The reason is that MV can only return a single answer for the query, which cannot fit in the situation that there usually are multiple values for a query. For instance, a movie or a book generally has multiple directors or authors. LTM, DART, and CASE do not perform well on all datasets. This is because they learn data veracity from large data, which is unavailable in the online setting. TF achieves higher F1-scores compared to other baselines because the data veracity learning process of TF does not heavily rely on the size of observed data.



(a) *Movie* (b) *Book* (c) *Flight*

Figure 8: Efficiency and scalability vs. data amount.

Table 3: Ablation study on line graph and incremental estimates.

Datasets	Types	FusionQuery			-incremental			-line graph		
		F1	QT	FT	F1	QT	FT	F1	QT	FT
Movie	J/K	**51.3**	25.7s	2.64s	41.4	24.8s	0.73s	12.2	2783s	0.28s
	J/C	**54.0**	12.7s	2.36s	48.4	11.7s	0.40s	49.1	1882s	0.29s
	K/C	**48.3**	31.6s	4.40s	44.7	29.7s	0.70s	45.5	4233s	0.29s
	J/K/C	**54.3**	39.2s	10.8s	32.2	40.7s	0.28s	50.4	4437s	0.32s
Book	J/C	**62.4**	0.19s	0.47s	48.5	0.18s	0.10s	57.1	11.9s	0.17s
	J/X	**60.0**	0.22s	0.56s	46.1	0.20s	0.10s	59.3	11.7s	0.17s
	C/X	**59.6**	0.16s	0.38s	49.4	0.16s	0.10s	55.3	8.39s	0.16s
	J/C/X	**60.3**	0.31s	1.07s	47.2	0.30s	0.12s	57.2	15.8s	0.18s
Flight	C/J	72.9	29.8s	109.9s	63.0	28.8s	31.1s	**75.2**	13.2h	0.50s
Stock	C/J	**71.6**	0.72s	0.36s	36.1	0.55s	0.12s	69.6	450.8s	0.19s

Efficiency. FusionQuery achieves the best or comparable performance in most cases. Note that, we do not count MajorityVoter due to its naive idea and bad performance. For each query, FusionQuery only takes 0.24s, 0.01s, 0.54s, and 0.04s to execute the entire workflow on *Movie*, *Book*, *Flight*, and *Stock* respectively, which are shorter than an update interval of most real-world data lakes. Moreover, even in scenarios where the data update interval is shorter than the execution time of FusionQuery, our framework still has a high probability of not being impacted by data updates. Because the query-related data that FusionQuery performs data fusion on remains unchanged in data updates and consequently the outputs can be reused for queries even if underlying data are modified, given the high probability that modifications primarily affect irrelevant data. Hence, these two features ensure that FusionQuery adapts to support data updates as claimed in Section 1.

5.3 Comparison with Batch Baselines

Setting. Prior to conducting batch data fusion, it is required to detect the records referring to the same entities. Here, we use SIF [44] with pre-trained SBERT to identify relevant entities, whose benefits are free from the model training, fast execution, and high recall. According to the statistics of our datasets, there are on average three latent true answers for each query. Therefore, we select top-3 values with the highest veracity scores as the fusion results to generate consistent data. In addition, we measure the accuracy of query answering over the generated data via the F1-score. We distinguish these query-supported variants of batch data fusion baselines by prefixing them with "QS". Table 2 lists the performance of our method and baselines on three datasets.

Effectiveness. FusionQuery achieves the best effectiveness and outperforms batch data fusion baselines by around 10 points in F1-score. The main reason is that the performance of batch data fusion extremely relies on high-quality entity matching, which is still unresolved, especially in the case of the multi-sourced heterogeneous circumstance. Low-quality entity matching will introduce a lot of noise into data fusion, which leads to poor query quality. In contrast, FusionQuery finds out entities referring to the same real-world objects in the meantime of query processing. Thus, FusionQuery is able to support fast and accurate detection of relevant entities without complex data matching.

Efficiency. FusionQuery dominates other batch data fusion methods. FusionQuery is completed in seconds while the batch data fusion methods are completed in minutes, or even in hours. The reasons can be concluded in two aspects. On the one hand, entity matching has inherently quadratic complexity, which is time consuming as a preprocessing step. On the other hand, batch data fusion updates source trustworthiness using entire data in all sources once data veracity updates, which takes much more time, especially on large-scale data. In contrast, FusionQuery performs data fusion on the query-related data on demand during query processing, which eliminates irrelevant data so as to speed up the iterative process of data fusion.

Scalability. To evaluate the scalability of data fusion methods on large-scale data, we repeat sampling data from the dataset to augment it. Figure 8 depicts the scalability of FusionQuery compared to batch data fusion baselines when varying the size of datasets. When the data amount scales up to 10 million, only LTM and FusionQuery complete to return answers to queries. TF, DART, and CASE fail because of time limits (1 day) or storage limits. In addition, FusionQuery takes shorter time to finish query processing than LTM, especially on the larger dataset. This efficiency in handling data fusion operations further supports the framework's ability to operate effectively in scenarios with frequent data updates.

5.4 Ablation Study

Threshold update. We conduct an ablation study to evaluate the effect of threshold update on the performance of FusionQuery. Figure 9 illustrates the results of the ablation study. We can find several observations: (1) FusionQuery is sensitive to the semantic matching threshold τ, especially on the *Book* dataset, since the semantic matching threshold τ controls the quality and quantity of the candidate answer set. (2) FusionQuery with threshold update outperforms FusionQuery without threshold update on two datasets. The reason is that the automatic adjustment of τ helps FusionQuery strike a balance between precision and recall.

Line graph. We conduct an ablation study where the line graph-based search proposed in our work is replaced with Node-First-Framework (NFF) search [25], which is the most efficient method for natural language answering over KGs to our best knowledge. The study measures query efficiency in terms of query time (QT) in seconds. Table 3 showcases the superior query efficiency achieved by FusionQuery with the line graph transformation compared to the NFF-based search method. Notably, the line graph-based method achieves approximately 100× acceleration in query processing on all datasets.

Incremental estimation. We compare FusionQuery with incremental estimations with a non-optimized version that lacks these incremental components. The findings reveal that FusionQuery

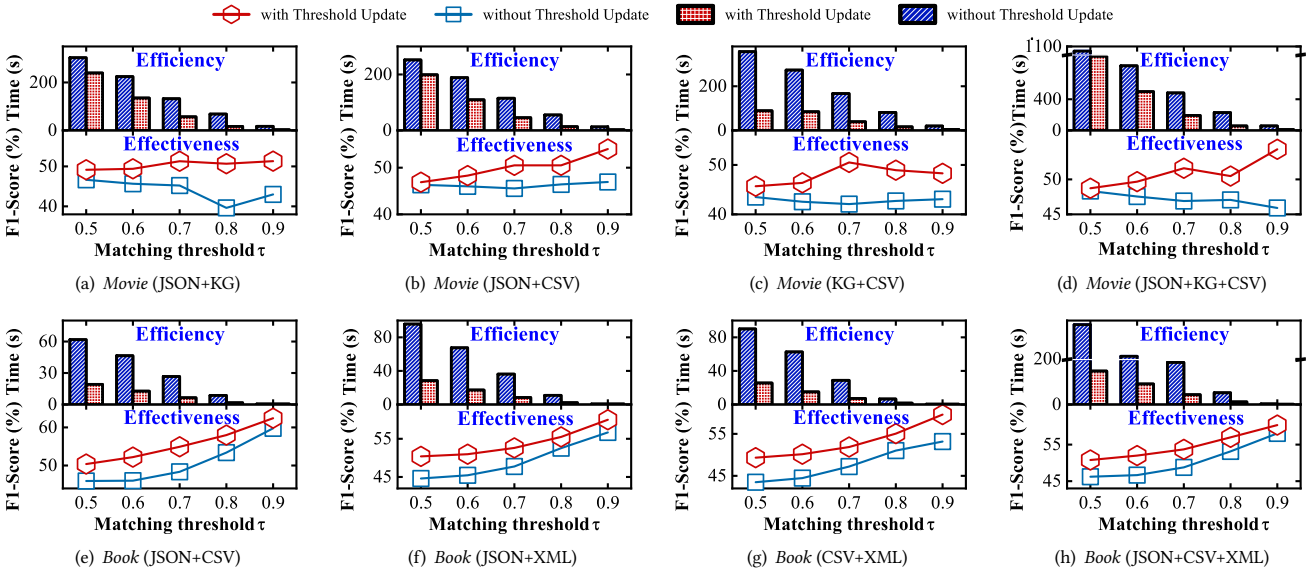


Figure 9: Performance of FusionQuery with Threshold update and without Threshold update vs. matching threshold τ.

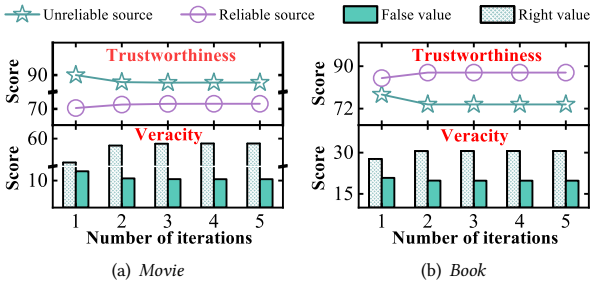


Figure 10: Estimates of FusionQuery under different number of iterations.

with incremental source trustworthiness estimations outperforms the non-optimized version by approximately 10 points in F1 score. While there is a trade-off in data fusion efficiency (as indicated by FT), the overall improvement in data fusion quality reinforces the effectiveness of FusionQuery.

5.5 Parameter Sensitivity

Initial Trustworthiness. We then explore the impact of initial trustworthiness on the performance of FusionQuery, by varying the initial trustworthiness from 0.95 to 0.75. As shown in Figure 11, the setting of the initial trustworthiness would not seriously affect the F1-score, which shows the robustness of FusionQuery to the initialization. Moreover, we find that FusionQuery shows stronger robustness on the smaller dataset (i.e. *Book*). The reason can be that FusionQuery needs fewer queries to obtain the stable estimates of source trustworthiness on a smaller dataset.

Number of iterations. To verify the theoretical analysis on the convergence of FusionQuery, we conduct experiments on the convergence process of FusionQuery. Taking two on-demand fusion queries on *Movie* and *Book* datasets as case studies, as shown in Figure 10, we examine the variance of data veracity and source trustworthiness by increasing the number of iterations. We can observe that both data veracity and source trustworthiness converge in 2 iterations regardless of the initialized values and datasets.

6 CASE STUDY

We present a case of applying FusionQuery to real-world heterogeneous data. Firstly we apply MRGC [22] to transform (semi-)structured data to KGs and an LLM (Vicuna-13B)-based information extraction (IE) method [38] to convert textual data (Wikidata) into KGs. Then, apply FusionQuery to process queries over KGs.

Information loss analysis. On the one hand, MRGC realizes lossless transformation from (semi-)structured data to KGs. On the other hand, we assess the information loss caused in converting textual data into KGs. As we represent texts with semantic embeddings in the FusionQuery framework, we employ embedding similarity to quantify information loss. To visualize information loss, we compare semantic embeddings of the extracted triples with the original sentences and the ground truth (human-annotated triples derived from sentences). Figure 12 illustrates the embedding similarity between the extracted triples, original sentences, and the ground truth. The majority of extracted triples show high embedding similarities with both the original sentences (around 0.7) and the ground truth (around 0.9), indicating that most key information is retained during the IE process.

Applicability. In the case study, we apply FusionQuery to a set of real-world heterogeneous data, including textual data (e.g., wikidata) and structural data (e.g., CSVs from IMDb). Figure 13 presents a case of evaluating FusionQuery over multiple sources to find directors of the movie "Crazy Eights". In this scenario, query-related data collected by FusionQuery exhibits conflicts, with only IMDb providing "Jimi Jones" as the answer while other movie data sources all provide "James K. Jones". Leveraging the evaluation of data veracity, FusionQuery emits "Jimi Jones" as the answer with a veracity score of 0.53. The decision is based on recognizing IMDb as a high-quality movie data source with higher trustworthiness.

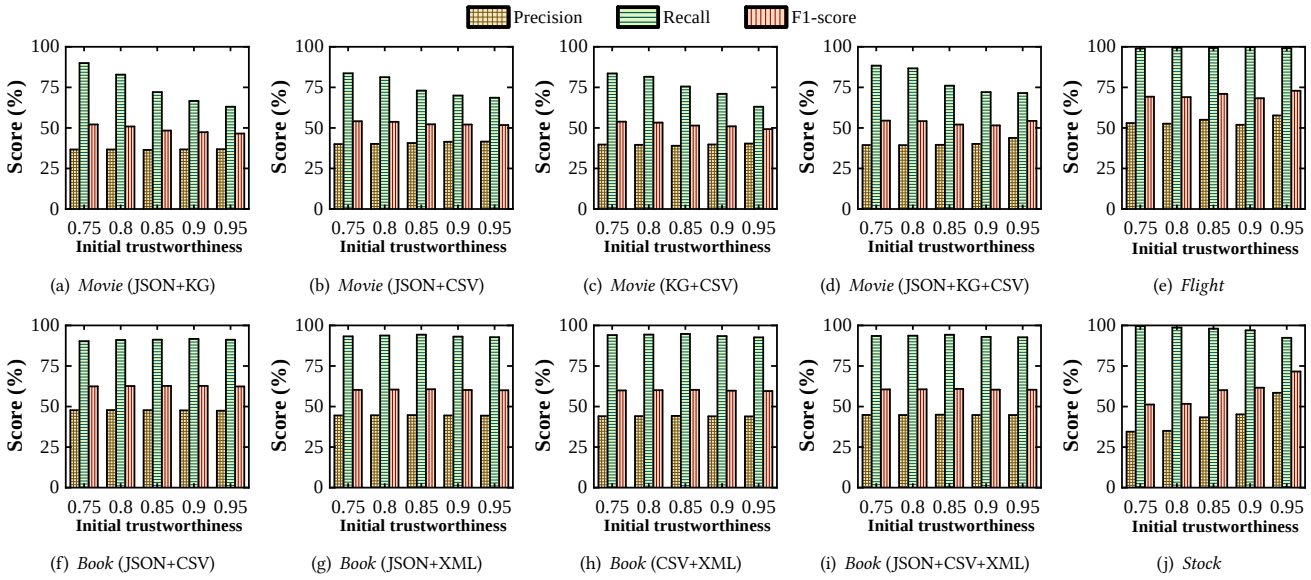


Figure 11: Performance evaluation vs. initial Trustworthiness.

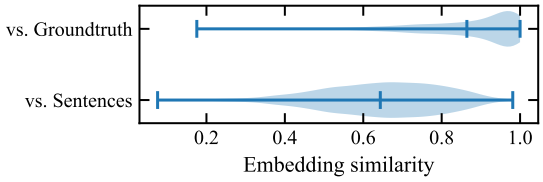


Figure 12: Information loss measured by embedding similarity: Extracted triples vs. Sentences and Extracted triples vs. Ground truth.

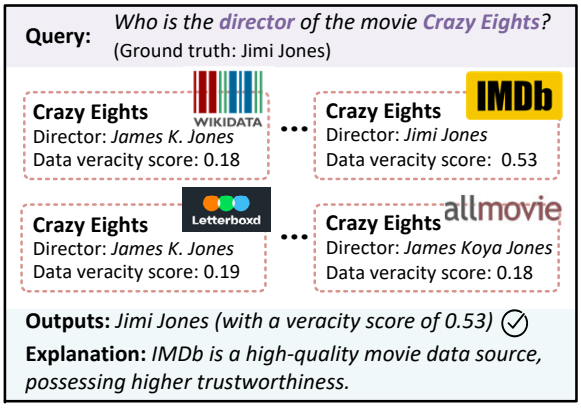


Figure 13: A real-world case of FusionQuery.

Limitations. Considering an extreme case where there is a total absence of suitable KGs, FusionQuery faces challenges in identifying relevant data to form query-related data, resulting in its inability to generate reliable answers for a given query. Figure 14 illustrates the simulated scenario, where FusionQuery is applied to retrieve information about movies. In this example where the movie "Audition" is not present in the data sources, FusionQuery attempts to form query-related data by collecting movies with names prefixed by "audition". This introduces irrelevant data and consequently FusionQuery fails to return correct answers.

7 RELATED WORK

Extensive studies have been proposed for the data fusion task, aiming to resolve conflicts in multiple sources. Most existing works apply offline data fusion in a full batch manner and create consistent data for downstream tasks, which could be generally divided into three categories: 1) iterative methods; 2) optimization-based methods; and 3) probabilistic methods. **Iterative methods** [17, 18, 36, 50] calculate the trustworthiness of data sources and veracity of data items iteratively. Specifically, TruthFinder [50] is the first work to formulate the data fusion problem, and builds a heuristic algorithm that iteratively infers data veracity and source trustworthiness. DART [36] introduces domain features, and develops an iterative Bayesian based algorithm to realize accurate data fusion. However, these studies come with few theoretical guarantees of their convergence. To this end, EMRGMM [49] proposes an EM-like data fusion method with a theoretical guarantee. However, it only solves the data fusion problem on numeric data, which is greatly different from our problem setting. **Optimization-based methods** [8, 10, 32, 37] minimize the distance between data items and their corresponding groundtruth. Specifically, CrowdFusion [10] models the data fusion problem as an optimization problem, and makes use of the crowd to realize high-quality data fusion. CASE [37] leverages a heterogeneous network to model relationships between data sources and data items, and learns the presentation of data sources and data items with an optimization goal. However, these studies heavily rely on human annotation, which limits their applications in real-world scenarios. **Probabilistic methods** [16, 31, 40, 55] model a joint distribution of data items and data sources based on groundtruth, and maximize the likelihood. Specifically, LTM [55] applies a probabilistic graphical model and a Gibbs sampling to infer the joint probability of data items and data sources. MBM [48] presents an integrated Bayesian approach, and takes auxiliary features (e.g.,

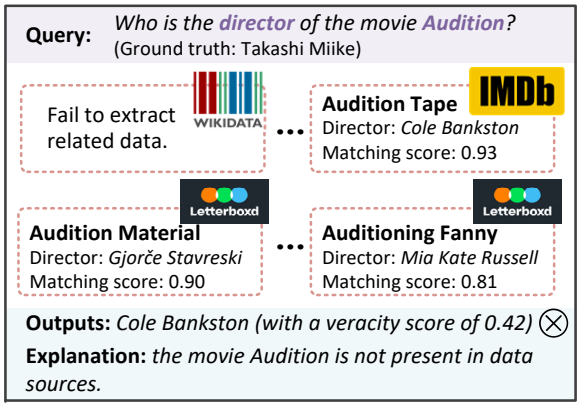


Figure 14: An extreme case of FusionQuery.

mutual exclusive relation of values) into account. However, their performance highly depends on hyperparameters. In addition, as stated in Section 1, batch data fusion suffers from poor scalability, slow response to data updates, and complex data matching. To address these, we propose the on-demand fusion query to perform data fusion on a small amount of query-related data to eliminate the conflicts in the query results.

Several efforts have been devoted to the on-demand fusion query. FuseM [51] is the first pipeline designed for query-centric data fusion on web markup data, which first adopts the BM25 model to retrieve entities and then uses machine learning classifiers in a supervised manner to realize data fusion. PolyFuse [23] classifies the incoming data according to its data type and then calls the corresponding processor to realize data fusion. BrewER [42] supports SQL SP queries on dirty relational tables, which achieves entity resolution and conflict resolution on-demand. Nonetheless, All of them are only designed for a specific data type, which cannot solve conflicts across heterogeneous data. In view of this, we explore the problem of on-demand fusion queries over heterogeneous multi-source data.

8 CONCLUSIONS

In this paper, we propose FusionQuery, an efficient framework for on-demand fusion queries over heterogeneous data. It includes a query stage and a fusion stage. At the query stage, we cast heterogeneous data query as a knowledge graph matching problem. To accelerate graph matching, we introduce knowledge graph transformation to decouple the semantic information from graph structure, which facilitates the application of efficient semantic matching methods and non-attributed graph matching methods. At the fusion stage, we devise an EM-style algorithm with a theoretical guarantee, which iteratively updates data veracity and source trustworthiness. Under the on-demand fusion query setting, we provide a new estimation of source trustworthiness incrementally. To further improve effectiveness and efficiency, we present an autonomous threshold update mechanism to control the quantity and quality of query results automatically. Comprehensive experiments demonstrate the superiority of FusionQuery. In the future, we plan to further enhance the effectiveness of the on-demand fusion query.

REFERENCES

[1] Guy Aglionby and Simone Teufel. 2022. Faithful Knowledge Graph Explanations in Commonsense Question Answering. In *EMNLP*. 10811–10817.
[2] Mohammad Shahmeer Ahmad, Zan Ahmad Naeem, Mohamed Y. Eltabakh, Mourad Ouzzani, and Nan Tang. 2023. RetClean: Retrieval-Based Data Cleaning Using Foundation Models and Data Lakes. *CoRR* abs/2303.16909 (2023).
[3] Yoshua Bengio. 2000. Gradient-Based Optimization of Hyperparameters. *Neural Comput.* 12, 8 (2000), 1889–1900.
[4] Prajjwal Bhargava and Vincent Ng. 2022. Commonsense Knowledge Reasoning and Generation with Pre-trained Language Models: A Survey. In *AAAI*, Vol. 36. 12317–12325.
[5] Bibek Bhattarai, Hang Liu, and H. Howie Huang. 2019. CECI: Compact Embedding Cluster Index for Scalable Subgraph Matching. In *SIGMOD*. 1447–1462.
[6] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. 2016. Efficient Subgraph Matching by Postponing Cartesian Products. In *SIGMOD*. 1199–1214.
[7] Léon Bottou et al. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes* 91, 8 (1991), 12.
[8] Klaus Broelemann, Thomas Gottron, and Gjergji Kasneci. 2017. LTD-RBM: Robust and Fast Latent Truth Discovery Using Restricted Boltzmann Machines. In *ICDE*. 143–146.
[9] Gabrielle Karine Canalle, Ana Carolina Salgado, and Bernadette Farias Lóscio. 2021. A survey on data fusion: what for? in what form? what is next? *J. Intell. Inf. Syst.* 57, 1 (2021), 25–50.
[10] Yunfan Chen, Lei Chen, and Chen Jason Zhang. 2017. CrowdFusion: A Crowdsourced Approach on Data Fusion Refinement. In *ICDE*. 127–130.
[11] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. ACM, 1247–1261.
[12] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 10 (2004), 1367–1372.
[13] Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based Reasoning for Natural Language Queries over Knowledge Bases. In *EMNLP*. 9594–9611.
[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018).
[15] Hong-Hai Do and Erhard Rahm. 2002. COMA—a system for flexible combination of schema matching approaches. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 610–621.
[16] 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 *SIGKDD*. 601–610.
[17] Xin Luna Dong, Laure Berti-Équille, and Divesh Srivastava. 2009. Integrating Conflicting Data: The Role of Source Dependence. *Proc. VLDB Endow.* 2, 1 (2009), 550–561.
[18] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-Based Trust: Estimating the Trustworthiness of Web Sources. *Proc. VLDB Endow.* 8, 9 (2015), 938–949.
[19] Xin Luna Dong and Divesh Srivastava. 2015. *Big Data Integration*. Morgan & Claypool Publishers. https://doi.org/10.2200/S00578ED1V01Y201404DTM040
[20] Valeria Fionda and Giuseppe Pirrò. 2020. Learning Triple Embeddings from Knowledge Graphs. In *AAAI*, Vol. 34. 3874–3881.
[21] Yunjun Gao, Xiaoze Liu, Junyang Wu, Tianyi Li, Pengfei Wang, and Lu Chen. 2022. ClusterEA: Scalable Entity Alignment with Stochastic Training and Normalized Mini-batch Similarities. In *SIGKDD*. 421–431.
[22] Congcong Ge, Pengfei Wang, Lu Chen, Xiaoze Liu, Baihua Zheng, and Yunjun Gao. 2021. CollaborER: A Self-supervised Entity Resolution Framework Using Multi-features Collaboration. *CoRR* abs/2108.08090 (2021).
[23] Michael N. Gubanov. 2017. PolyFuse: A Large-Scale Hybrid Data Fusion System. In *ICDE*. 1575–1578.
[24] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *SIGMOD*. 1429–1446.
[25] Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *IEEE Trans. Knowl. Data Eng.* 30, 5 (2018), 824–837.
[26] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*. https://openreview.net/forum?id=rkE3y85ee
[27] Xin Jin, Zhengyi Yang, Xuemin Lin, Shiyu Yang, Lu Qin, and You Peng. 2021. FAST: FPGA-based Subgraph Matching on Massive Graphs. In *ICDE*. 1452–1463.

[28] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547.

[29] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *ICDE*. 468–479.

[30] Longbin Lai, Zhu Qing, Zhengyi Yang, Xin Jin, Zhengmin Lai, Ran Wang, Kongzhang Hao, Xuemin Lin, Lu Qin, Wenjie Zhang, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2019. Distributed Subgraph Matching on Timely Dataflow. *Proc. VLDB Endow.* 12, 10 (2019), 1099–1112.

[31] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. 2014. A Confidence-Aware Approach for Truth Discovery on Long-Tail Data. *Proc. VLDB Endow.* 8, 4 (2014), 425–436.

[32] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*. 1187–1198.

[33] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. 2012. Truth Finding on the Deep Web: Is the Problem Solved? *Proc. VLDB Endow.* 6, 2 (2012), 97–108.

[34] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. A Survey on Truth Discovery. *SIGKDD Explor.* 17, 2 (2015), 1–16.

[35] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (2020), 50–60.

[36] Xueling Lin and Lei Chen. 2018. Domain-Aware Multi-Truth Discovery from Conflicting Sources. *Proc. VLDB Endow.* 11, 5 (2018), 635–647.

[37] Shanshan Lyu, Wentao Ouyang, Yongqing Wang, Huawei Shen, and Xueqi Cheng. 2021. Truth Discovery by Claim and Source Embedding. *IEEE Trans. Knowl. Data Eng.* 33, 3, 1264–1275.

[38] Nandana Mihindukulasooriya, Sanju Tiwari, Carlos F. Enguix, and Kusum Lata. 2023. Text2KGBench: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text. *CoRR* abs/2308.02357 (2023).

[39] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proc. VLDB Endow.* 12, 12 (2019), 1986–1989.

[40] Jeff Pasternack and Dan Roth. 2013. Latent credibility analysis. In *WWW*. 1009–1020.

[41] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR* abs/1908.10084 (2019).

[42] Giovanni Simonini, Luca Zecchini, Sonia Bergamaschi, and Felix Naumann. 2022. Entity Resolution On-Demand. *Proc. VLDB Endow.* 15, 7 (2022), 1506–1518.

[43] Nan Tang, Chenyu Yang, Ju Fan, and Lei Cao. 2023. VerifAI: Verified Generative AI. *CoRR* abs/2307.02796 (2023).

[44] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep Learning for Blocking in Entity Matching: A Design Space Exploration. *Proc. VLDB Endow.* 14, 11 (2021), 2459–2472.

[45] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Y. Halevy. 2021. Database reasoning over text. In *ACL/IJCNLP*. 3091–3104.

[46] Ha Nguyen Tran, Jung-Jae Kim, and Bingsheng He. 2015. Fast Subgraph Matching on Large Graphs using Graphics Processors. In *DASFAA (Lecture Notes in Computer Science)*, Vol. 9049. 299–315.

[47] Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. PromptEM: Prompt-tuning for Low-resource Generalized Entity Matching. *Proc. VLDB Endow.* 16, 2 (2022), 369–378.

[48] Xianzhi Wang, Quan Z. Sheng, Xiu Susie Fang, Lina Yao, Xiaofei Xu, and Xue Li. 2015. An Integrated Bayesian Approach for Effective Multi-Truth Discovery. In *CIKM*. 493–502.

[49] Houping Xiao, Jing Gao, Zhaoran Wang, Shiyu Wang, Lu Su, and Han Liu. 2016. A Truth Discovery Approach with Theoretical Guarantee. In *SIGKDD*. 1925–1934.

[50] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. 2008. Truth Discovery with Multiple Conflicting Information Providers on the Web. *IEEE Trans. Knowl. Data Eng.* 20, 6 (2008), 796–808.

[51] Ran Yu, Ujwal Gadiraju, Besnik Fetahu, and Stefan Dietze. 2017. FuseM: Query-Centric Data Fusion on Structured Web Markup. In *ICDE*. 179–182.

[52] Ye Yuan, Delong Ma, Zhenyu Wen, Zhiwei Zhang, and Guoren Wang. 2021. Subgraph Matching over Graph Federation. *Proc. VLDB Endow.* 15, 3 (2021), 437–450.

[53] Ye Yuan, Delong Ma, Aoqian Zhang, and Guoren Wang. 2022. Consistent Subgraph Matching over Large Graphs. In *ICDE*. 2536–2548.

[54] Li Zeng, Lei Zou, M. Tamer Özsu, Lin Hu, and Fan Zhang. 2020. GSI: GPU-friendly Subgraph Isomorphism. In *ICDE*. 1249–1260.

[55] Bo Zhao, Benjamin I. P. Rubinstein, Jim Gemmell, and Jiawei Han. 2012. A Bayesian Approach to Discovering Truth from Conflicting Sources for Data Integration. *Proc. VLDB Endow.* 5, 6 (2012), 550–561.