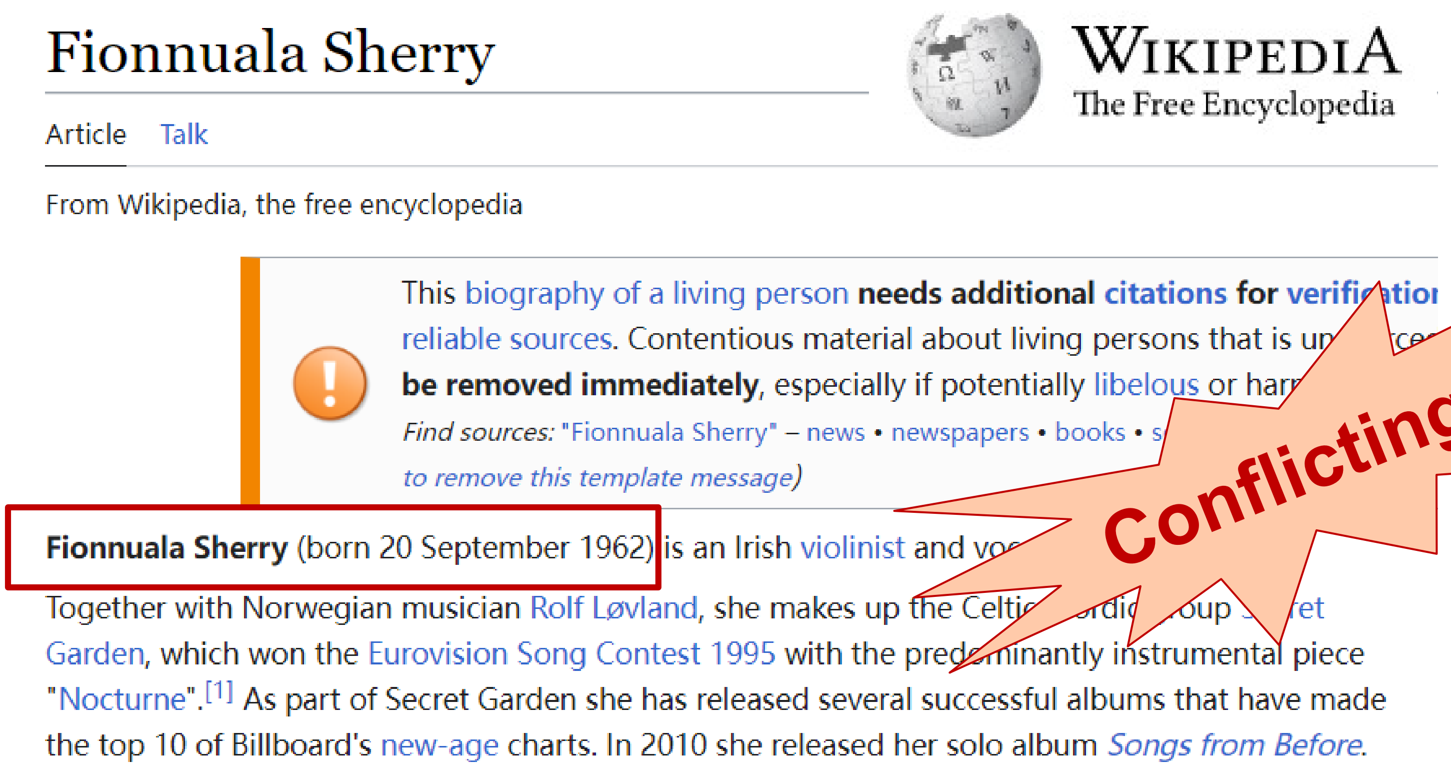
FusionQuery: On-demand Fusion Queries over Multi-source Heterogeneous Data

Junhao Zhu[1], Yuren Mao[1], Lu Chen[1], Congcong Ge[1],
Ziheng Wei[2], Yunjun Gao[1]

[1]Zhejiang University, [2]Wuhan University

Conflicting Data is Everywhere



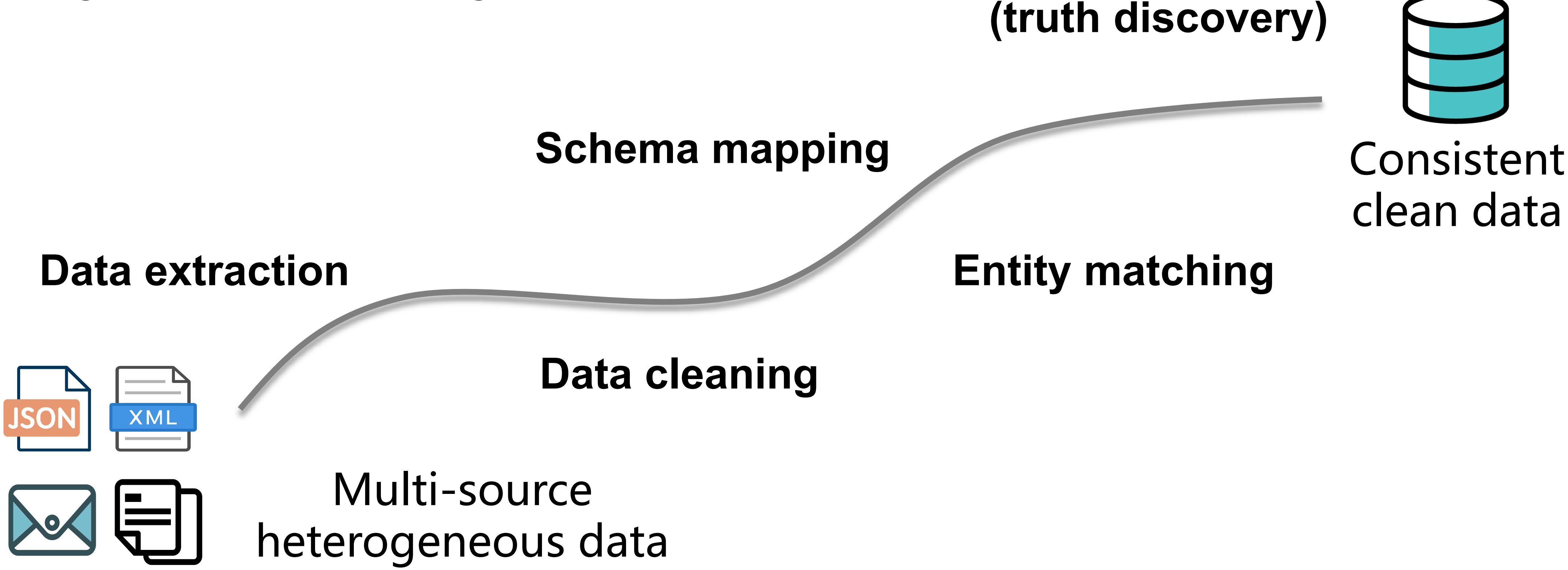
Conflicting!

Conflicting Data is Everywhere

Even well-known data providers make mistakes!

Possible Solution: Data Integration

Data integration suffers from low efficiency and flexibility



Interactive Application Example

Many applications (e.g., RAGs) need interactive speed.

Please generate a table regarding "united states house of representatives elections, 2004 in Ohio", the attributes are ["district", "incumbent", "first elected", "candidates"]. and the values under "district" column are ["Ohio's_1st_congressional_district", "Ohio's_2nd_congressional_district“, "Ohio's_3rd_congressional_district"]. The values under other columns are missing, which you should fill in with your knowledge.

District	Incumbent	First Elected	Candidates
Ohio's_1st_congressional_district	Steve Chabot (R)	1994	Steve Chabot (R), Emily Johnson (D), Alex Smith (I)
Ohio's_2nd_congressional_district	Rob Portman (R)	1993 (special)	Rob Portman (R), Sarah Davis (D), James White (I)
Ohio’s_3rd_congressional_district	Mike Turner (R)	2002 (special)	Mike Turner (R), Jennifer Lee (D), Robert Green (I)

Table synthesis by ChatGPT

2004 United States House of Representatives Elections in Ohio

WIKIPEDIA
The Free Encyclopedia

Ohio's 1st Congressional District Election, 2004

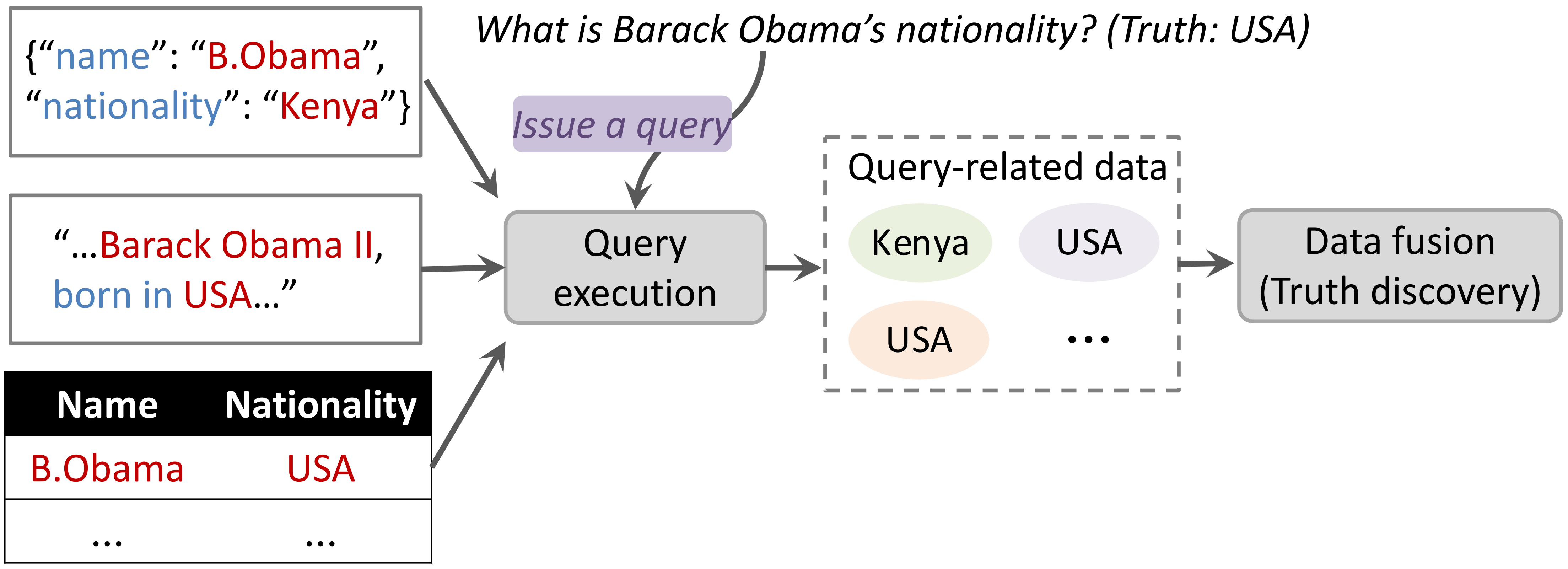
Party	Candidate	Votes	%
Republican	Steve Chabot	173,430	59.83
Democratic	Greg Harris	116,235	40.10
Independent	Rich Stevenson	198	0.07

Ohio's 2nd Congressional District Election, 2004

Party	Candidate	Votes	%
Republican	Rob Portman	227,102	71.70
Democratic	Charles W. Sanders	89,598	28.29
Independent	James J. Condit, Jr.	60	0.02

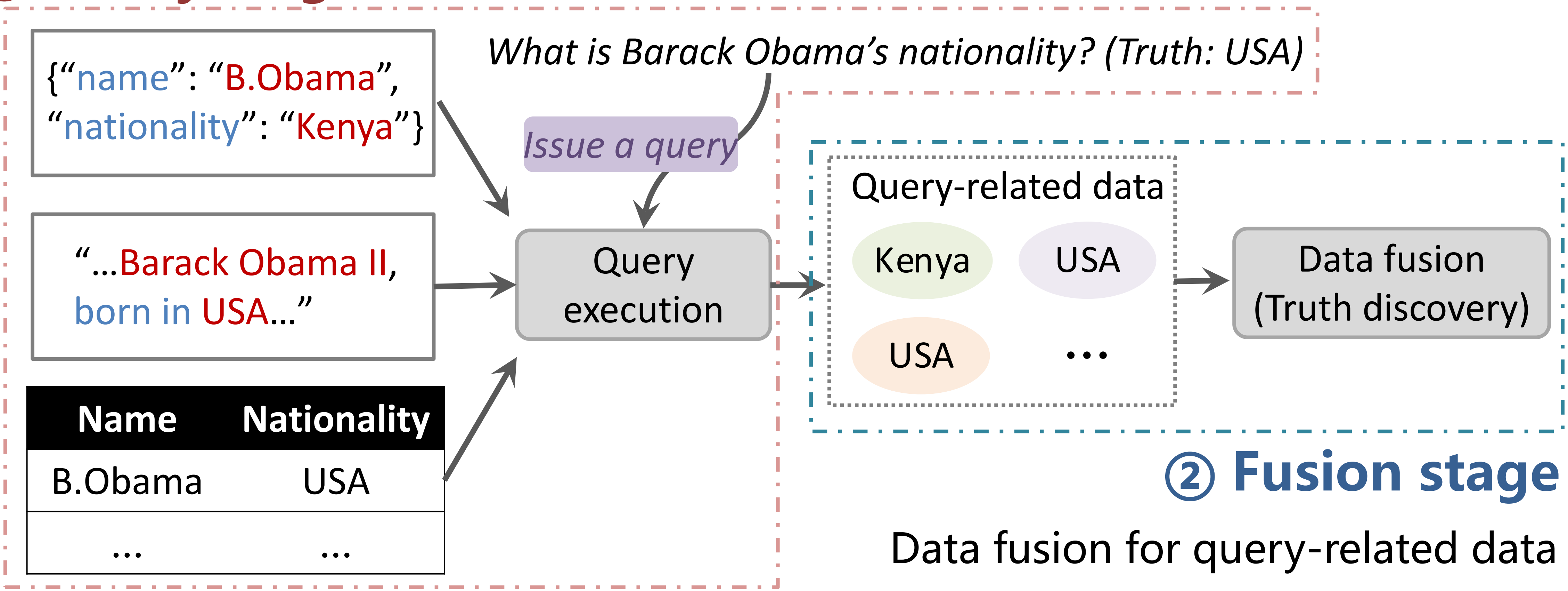
Conflict with data sources

A Better Way: FusionQuery



A Better Way: FusionQuery

① **Query stage** Process query over heterogenous data



② **Fusion stage**

Data fusion for query-related data

FusionQuery: Query Stage

Challenge: how to support unified queries across heterogenous data

Key idea: frame heterogeneous query as KG matching

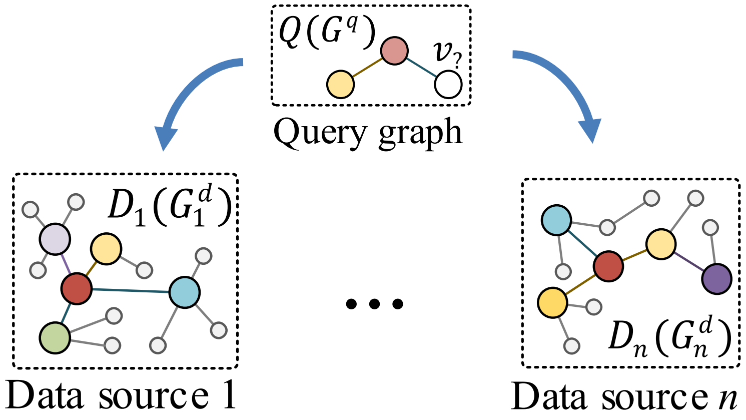
FusionQuery: Query Stage

Challenge: how to support unified queries across heterogenous data

Key idea: frame heterogeneous query as KG matching

However, KG matching is slow.

- Semantic and structural matching are iteratively performed.
- BFS takes $O(n(V^q + E^q)(V^d + E^d))$.



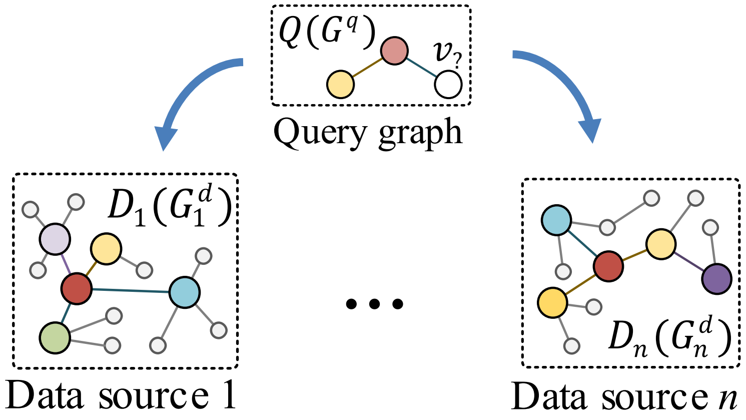
FusionQuery: Query Stage

Challenge: how to support unified queries across heterogenous data

Key idea: frame heterogeneous query as KG matching

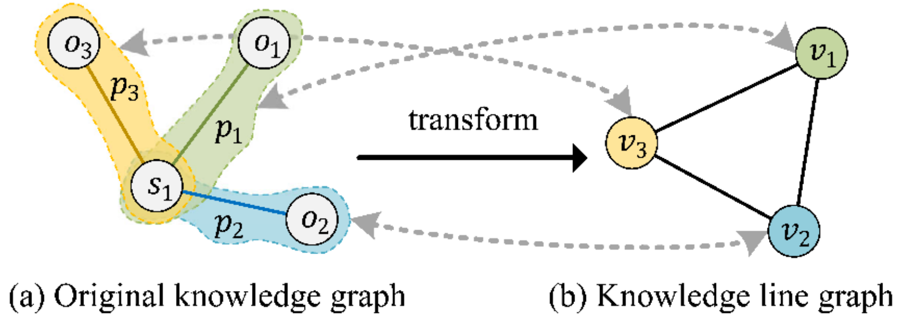
However, KG matching is slow.

- Semantic and structural matching are iteratively performed.
- BFS takes $O(n(V^q + E^q)(V^d + E^d))$.



Solution: Line Graph Transformation

- Convert triplets into nodes in a line graph.
- The time complexity is reduced to $O(E^q E^d)$

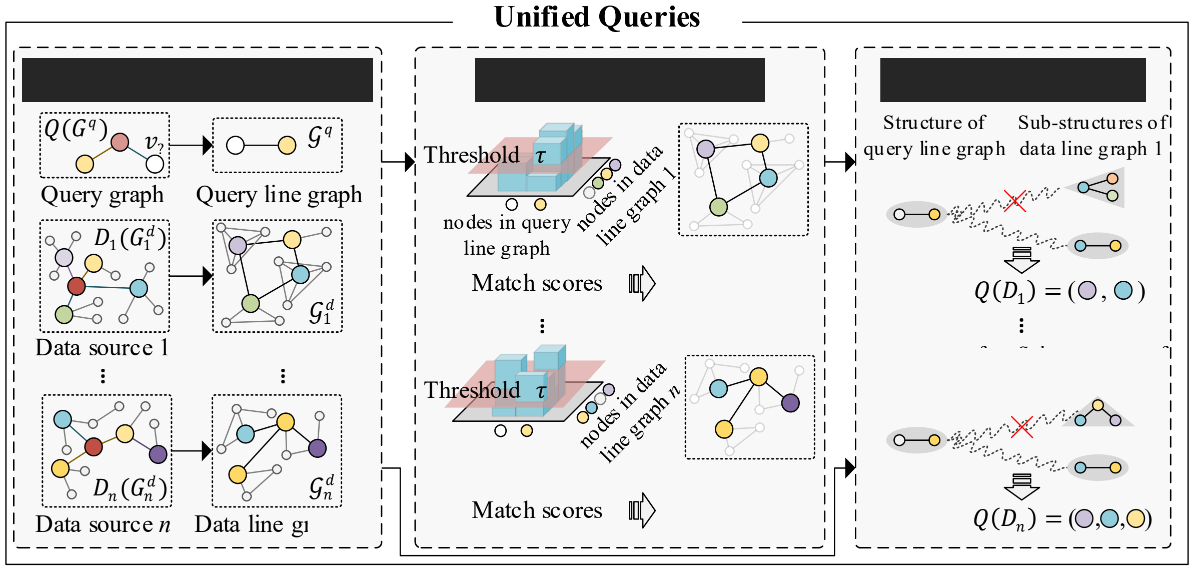


FusionQuery: Query Stage

By line graph transformation, the problem is reduced to sub-problems.

① Node-level semantic matching

- Nodes in line graphs are represented as embeddings by PLMs (e.g., BERT).
- Matching is determined by similarity of embeddings (decided by a threshold τ).



② Graph-level structure matching

- Leverage efficient off-the-shelf non-attributed graph matching algorithms.

FusionQuery: Query Stage

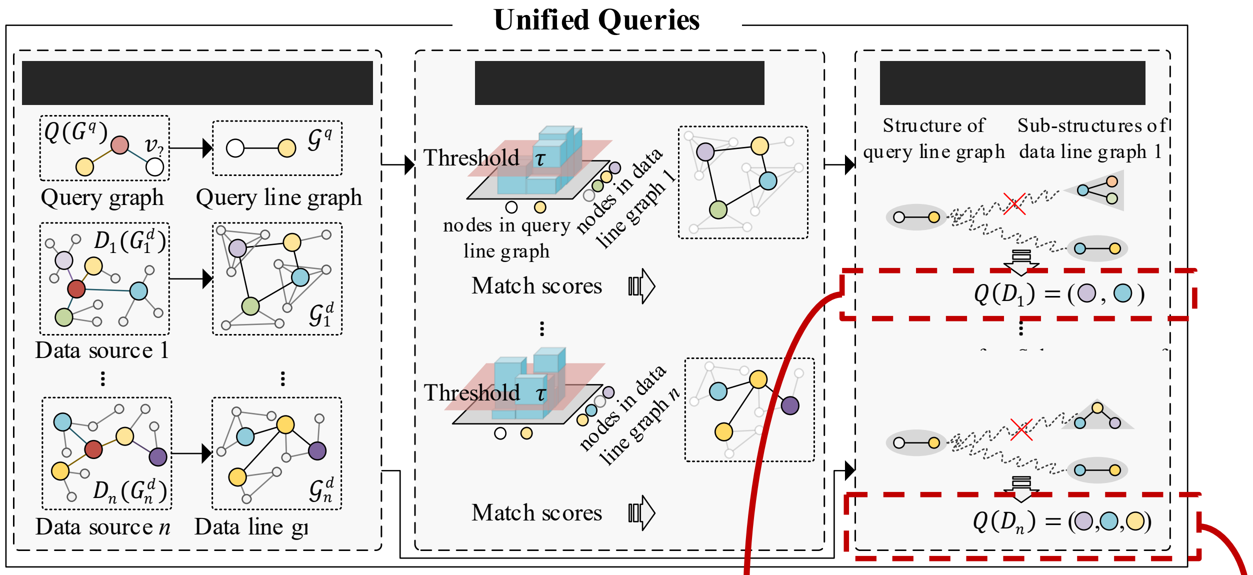
By line graph transformation, the problem is reduced to sub-problems.

① Node-level semantic matching

- Nodes in line graphs are represented as embeddings by PLM (e.g., BERT).
- Matching is determined by similarity of embeddings (decided by a threshold τ).

② Graph-level structure matching

- Leverage efficient off-the-shelf non-attributed graph matching algorithms.



query-related data is collected, where values from different sources may conflict with each other.

FusionQuery: Fusion Stage

Two key concepts in data fusion (truth discovery)

- **Data veracity** $\Pr(v)$: the veracity $\Pr(v)$ of a value v is the probability that the value v is a correct result to the query.
- **Source trustworthiness** $\Pr(D)$: the trustworthiness $\Pr(D)$ of a data source D is the probability that the source D provides true values for queries.
- A value has higher veracity score if it is provided by a more trustworthy data source, and vice versa (i.e., two scores are mutually relevant).

FusionQuery: Fusion Stage

Two key concepts in data fusion (truth discovery)

- **Data veracity** $\Pr(v)$: the veracity $\Pr(v)$ of a value v is the probability that the value v is a correct result to the query.
- **Source trustworthiness** $\Pr(D)$: the trustworthi is the probability that the source D provides true values for queries.
- A value has higher veracity score if it is provided by a more trustworthy data source, and vice versa (i.e., two scores are mutually relevant).

*Goal: find values in query-related data with **highest data veracity** as query results.*

FusionQuery: Fusion Stage

Two key concepts in data fusion (truth discovery)

- **Data veracity** $\Pr(v)$: the veracity $\Pr(v)$ of a value v is the probability that the value v is a correct result to the query.
- **Source trustworthiness** $\Pr(D)$: the trustworthi is the probability that the source D provides true values for queries.
- A value has higher veracity score if it is provided by a more trustworthy data source, and vice versa (i.e., two scores are mutually relevant).

*Goal: find values in query-related data with **highest data veracity** as query results.*

Drawback of existing data fusion methods

- Require a large amount of data to accurately estimate data veracity scores.
- However, what we can obtain is only query-related data (small amount).

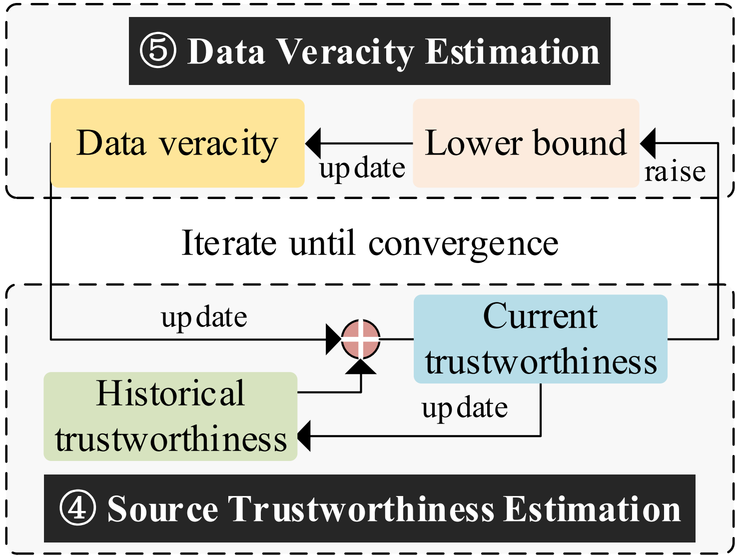
FusionQuery: Fusion Stage

On-demand data fusion

- The data veracity $\Pr(v)$ is approximated by its upper bound:

$$\log \Pr(v) \approx \sum_{D \in \mathcal{D}} \Pr(D|v) \log \frac{\Pr(v|D)\Pr(D)}{\Pr(D|v)}$$

$$\Pr(v|D) = \begin{cases} \Pr(D)\,, v \in D \\ 1 - \Pr(D)\,, otherwise \end{cases}$$



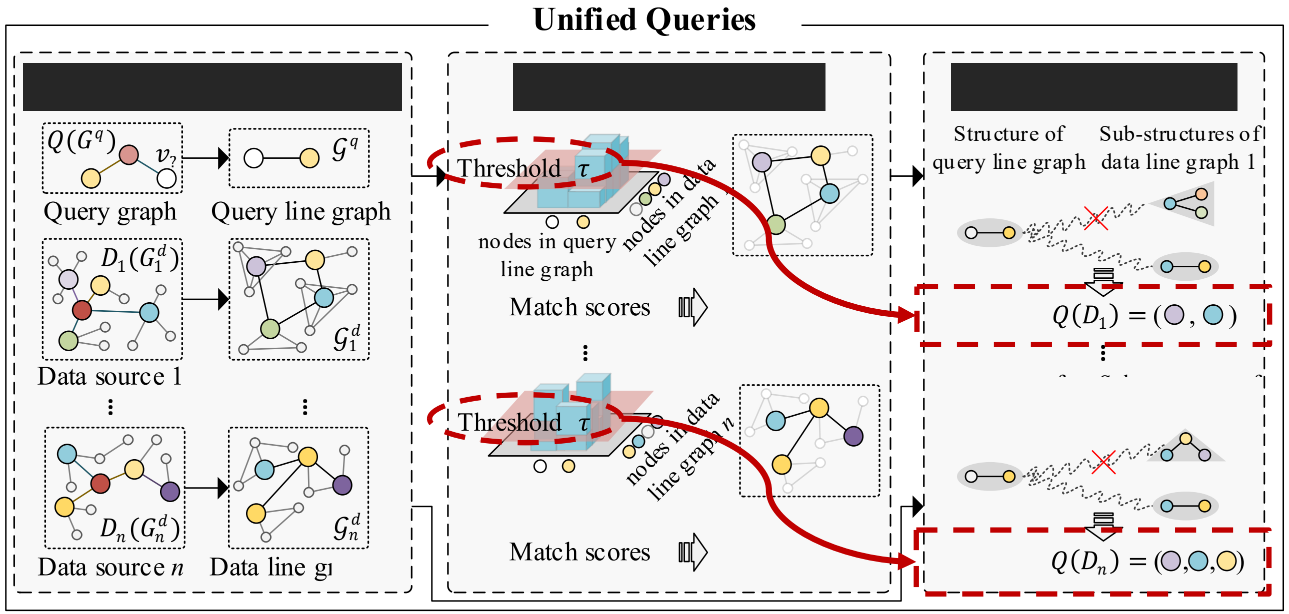
- The source trustworthiness $\Pr(D)$ is incrementally estimated:

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

$Data(Q,\mathcal{D})$: query-related data, $Pr^h(D)$: historical source trustworthiness, $\mathcal{H}$: the amount of historical query results

FusionQuery: Threshold Update

Threshold τ affects the quality and quantity of query results



- A low threshold τ results on a low precision; A high threshold τ results on a low recall.

FusionQuery: Threshold Update

Our solution: automatically adjust τ inspired by meta-learning

- Core idea: Adjust threshold τ by gradient descent.

*Goal: find value with **highest data veracity*** ⟶ Optimization goal: $\max \Pr(v)$

Do a transformation to the condition:

$$\Pr(v) \geq \tau \rightarrow \Pr(v) = \tau + \epsilon_v \ (\epsilon_v \geq 0)$$

Substitute $\Pr(v)$ in the estimation of $\Pr(D)$ and get the gradient of $\Pr(D)$:

$$\nabla_\tau \Pr(D) = |Data(Q, \mathcal{D})| + \sum_{v \in Data(Q,\mathcal{D})} \frac{\Pr(v) \cdot D_v[Q]}{\mathcal{H} + |Data(Q, \mathcal{D})|}$$

Update τ by the gradient:

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

Evaluation Setup

Datasets

- Four real-world datasets with heterogenous data types

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	

Baselines

- Offline batch data fusion methods:
  - ✓ Naïve method: Majority Voter
  - ✓ Iterative methods: TruthFinder, DART
  - ✓ Optimization method: CASE
  - ✓ Probabilistic method: LTM
- Adapt them to on-demand data fusion variants

Evaluation: Performance

FusionQuery achieves better accuracy and comparable runtime compare to on-demand data fusion baselines.

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.

Evaluation: Performance

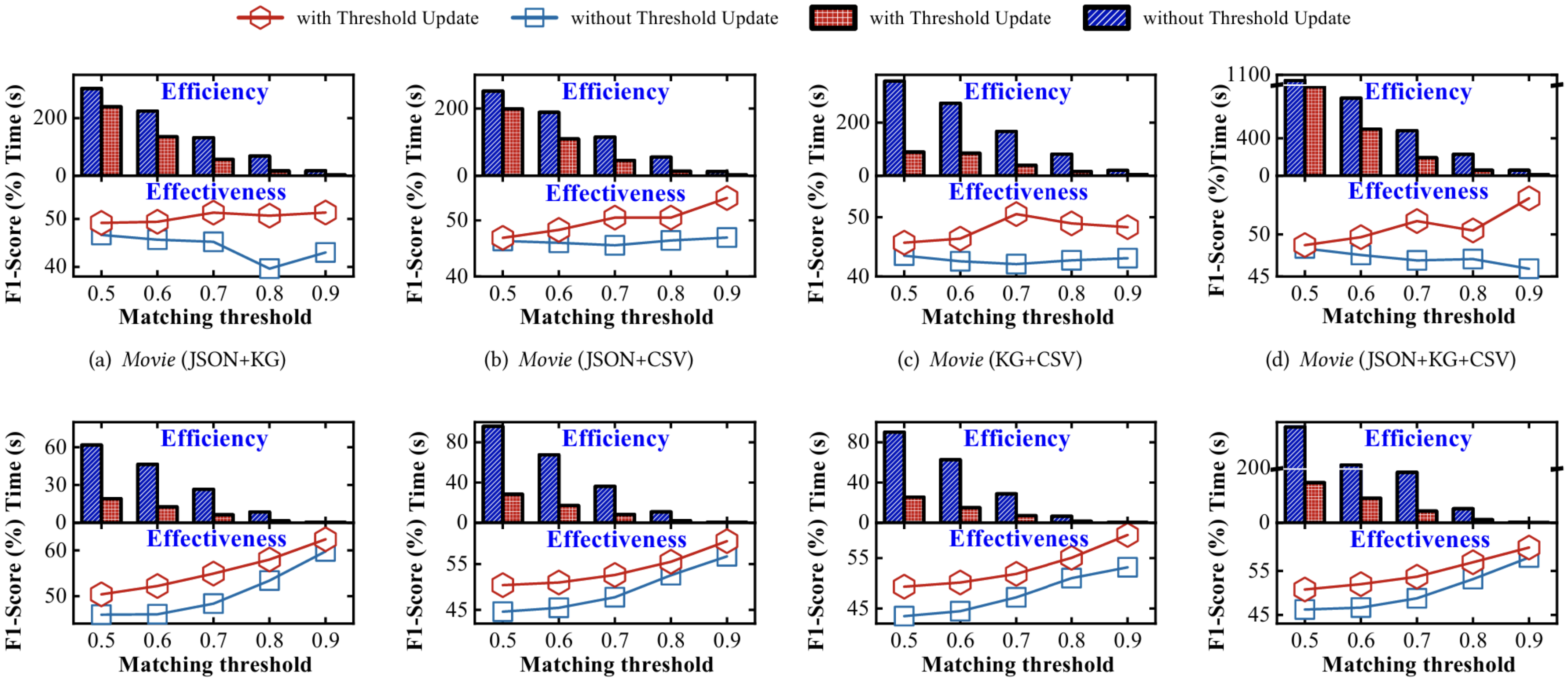
FusionQuery outperforms offline data fusion baselines in both effectiveness and efficiency.

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.

Evaluation: Ablation Study

Threshold update mechanism makes FusionQuery more robust.



(a) *Movie* (JSON+KG) (b) *Movie* (JSON+CSV) (c) *Movie* (KG+CSV) (d) *Movie* (JSON+KG+CSV)

(e) *Book* (JSON+CSV) (f) *Book* (JSON+XML) (g) *Book* (CSV+XML) (h) *Book* (JSON+CSV+XML)

Conclusion

Feel free to reach out for questions!

Contributions

- A framework for on-demand fusion queries over heterogenous data
- An efficient knowledge graph matching framework
- A convergence-guaranteed data fusion algorithm
- An autonomous threshold update mechanism

https://github.com/JunHao-Zhu/FusionQuery

Thank you! Any Question?