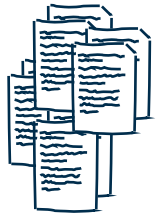
Beyond Retrieval: A Study of Using LLM Ensembles for Candidate Filtering in Requirements Traceability

Dominik Fuchß, Stefan Schwedt, Jan Keim, Tobias Hey

Requirements Traceability using Information Retrieval (IR)

High-Level Requirement

„The user shall be able to login with username and password.“



Low-Level Requirements
LLR1: …
LLR2: …
LLR3: …
…

Vectorize

$\mathbf{R}\rightarrow \begin{pmatrix} 0.1337 \\ \vdots \\ 0.4242 \end{pmatrix}$

Calculate Similarity

(HLR,LLR) $\rightarrow \angle$

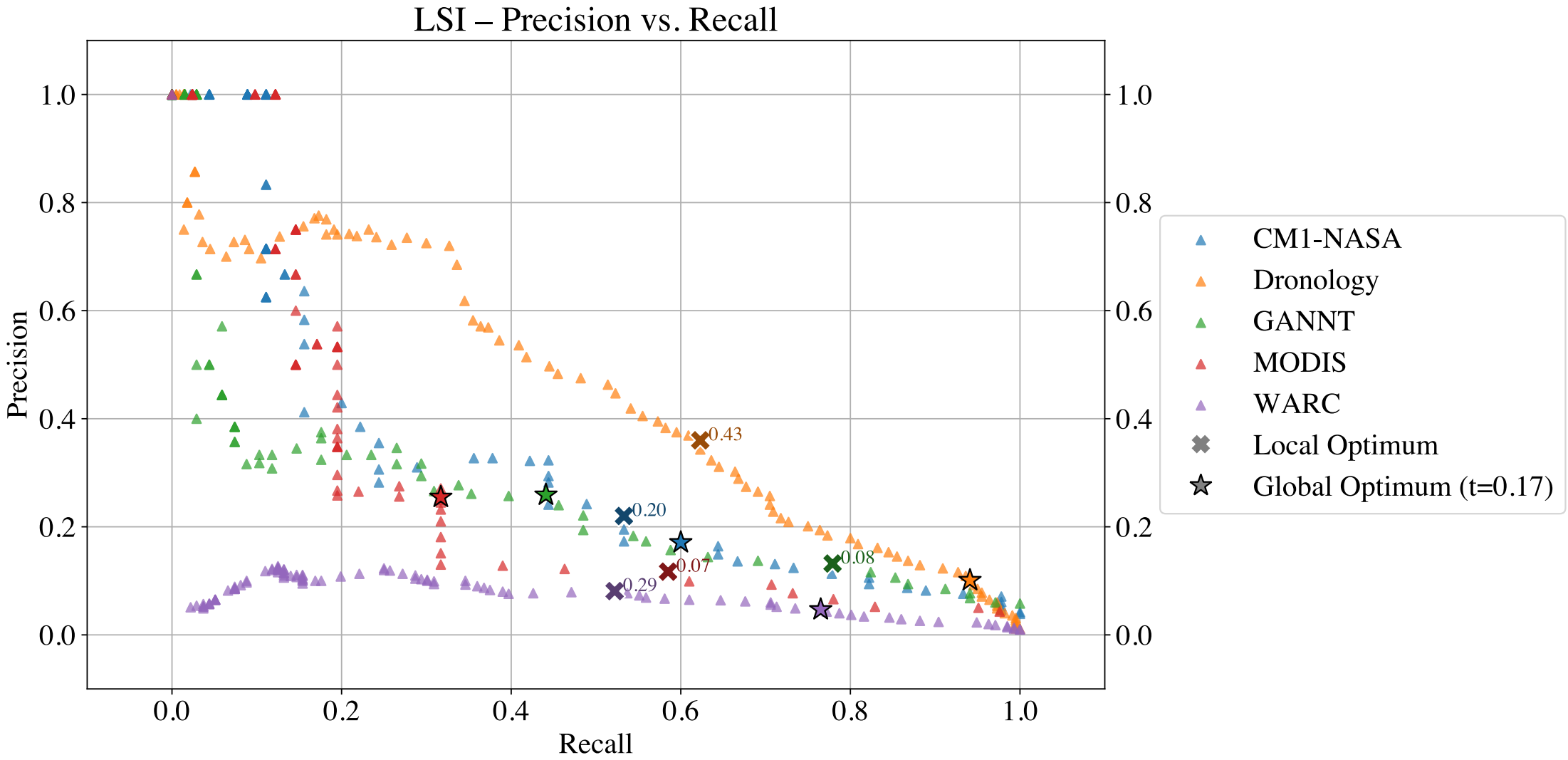
Ranked List	
Low-Level	Similarity
LLR3: …	0.9
LLR1: …	0.7
LLR2: …	0.15
LLR4: …	0.1
LLR9: …	0.042

t=0.5

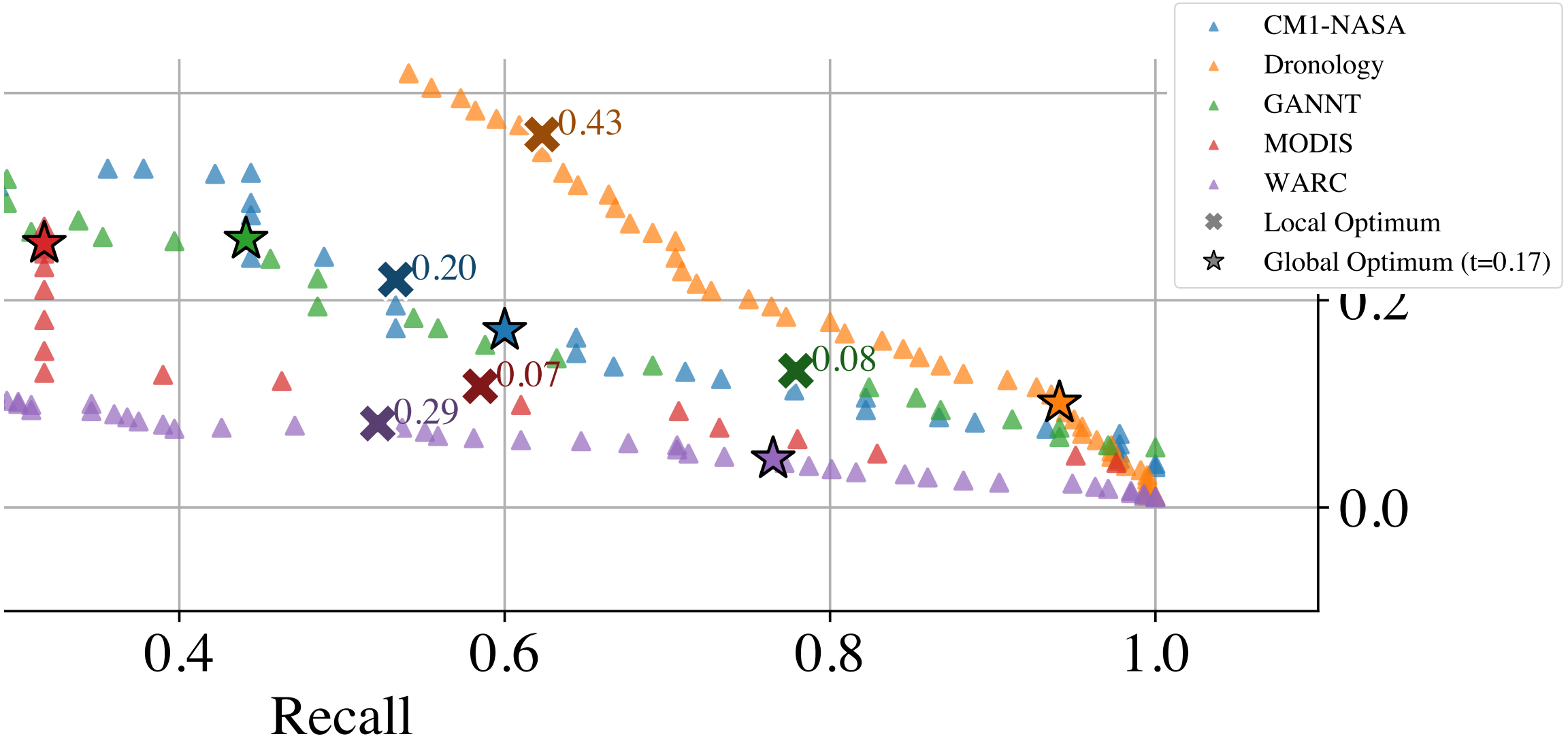
k=4

- Similarity Thresholds, e.g., t=0.5
- Top-K-Retrieval, e.g., k=4

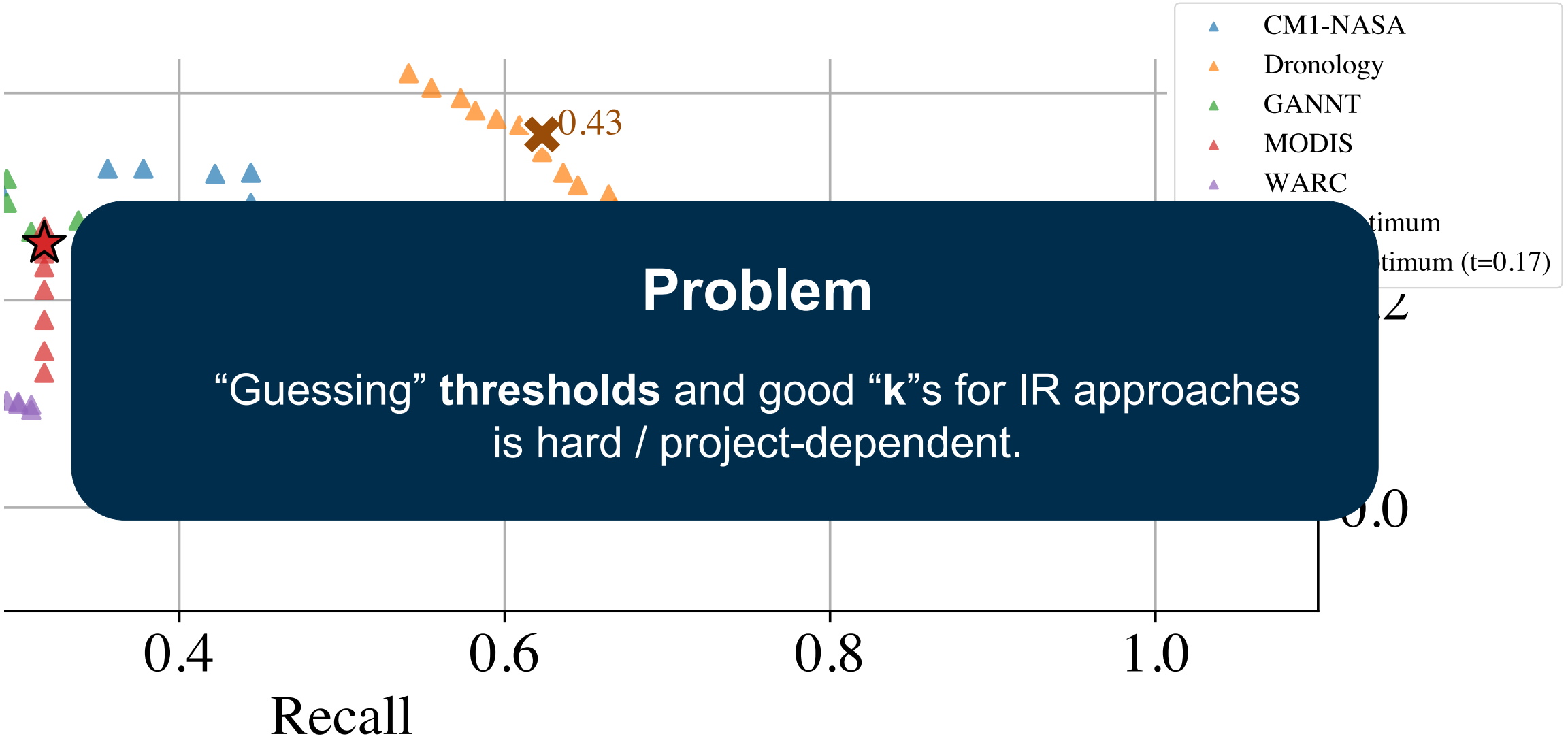
Requirements Traceability using Information Retrieval (IR)



Requirements Traceability using Information Retrieval (IR)



Requirements Traceability using Information Retrieval (IR)



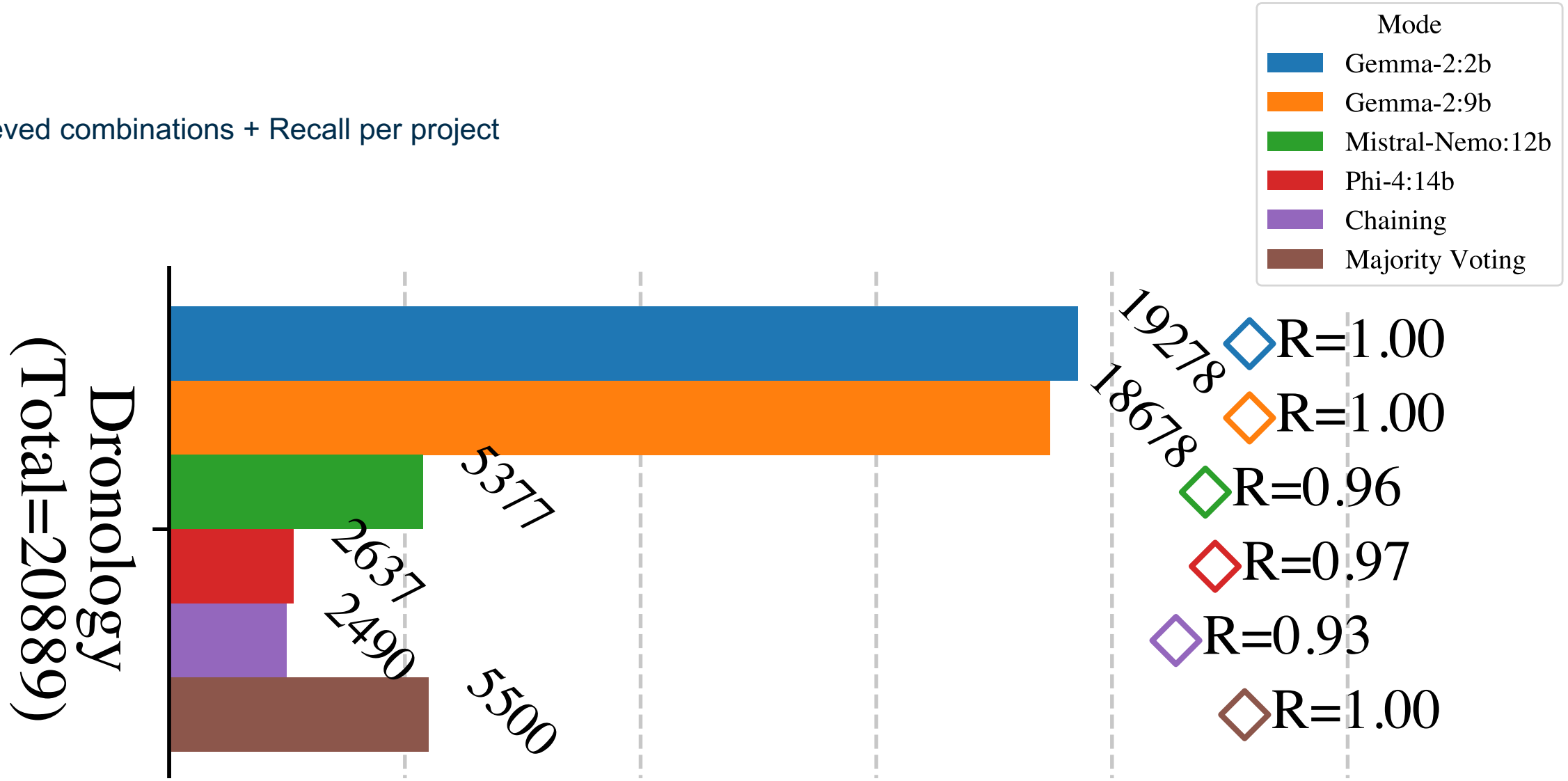
Problem

"Guessing" **thresholds** and good "**k**"s for IR approaches is hard / project-dependent.

Idea: Small LLMs to *lightweightly* filter all requirement pairs

Evaluation: Reduction of Search Space & Maintaining Recall

- Retrieved combinations + Recall per project



Evaluation: Comparison to State of the Art Approaches

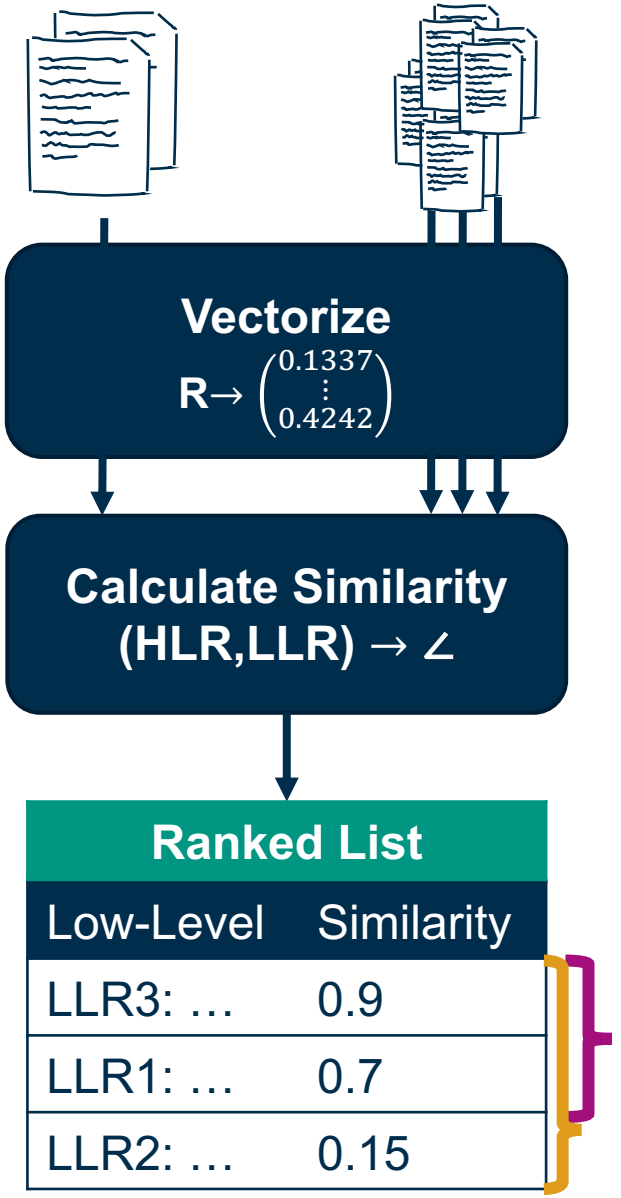
- Approaches for Comparison
  - VSM: IR with similarity threshold
  - LSI: IR with similarity threshold
  - Embeddings: IR with Top-K
  - LiSSA: RAG-based with Top-K
- Metrics
  - F_1-score (esp. full automation)
  - F_2-score (esp. semi-automatic)

Approach	F_1-score	F_2-score
VSM_{GO}	.27	.34
LSI_{GO}	.23	.33
$Embeddings_{GO}$	.40	.50
LiSSA (GPT-4o)	**.50**	**.51**
Majority Voting	.18	.34
Chaining	.28	.45
Chaining + GPT-4o	.34	.50

Conclusion

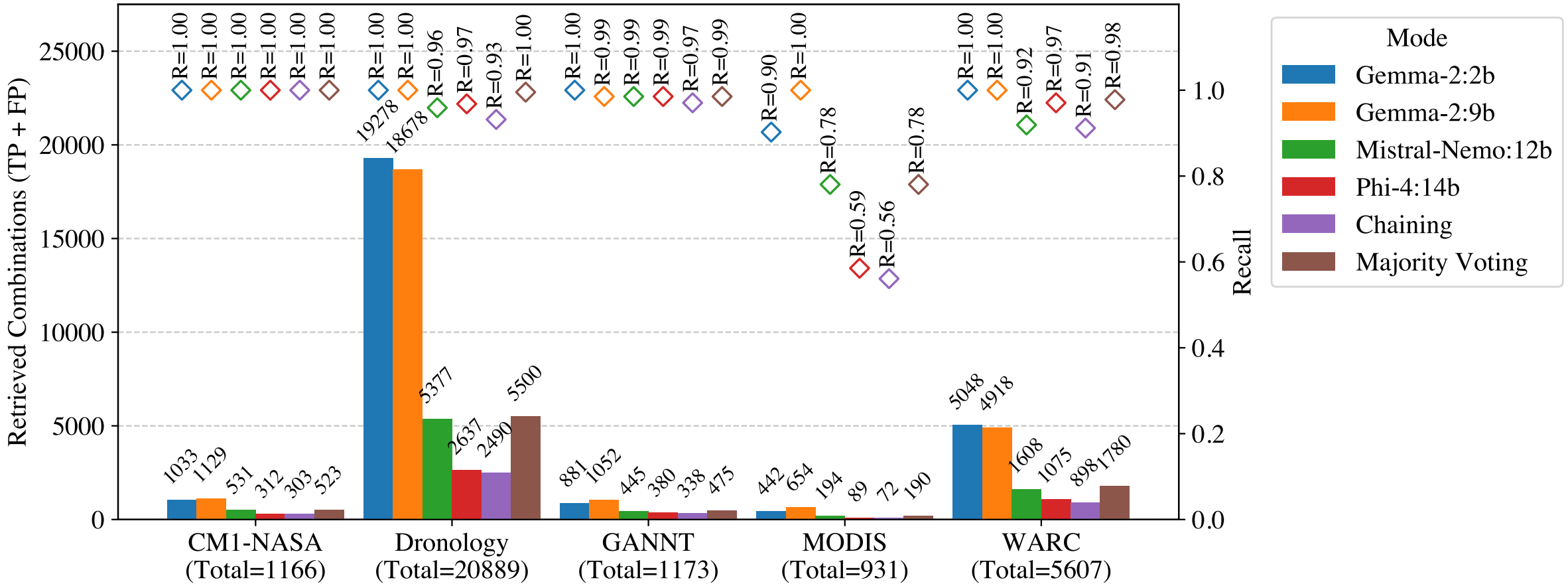
- Problem: “guessing” **thresholds** and good “**k**”s for IR is hard / project-dependent
- Approach: Use small LLMs to reduce search space while maintaining recall
- Results: Ensembles of small LLMs …
  - can filter non-linked candidate pairs
  - can outperform classical methods like VSM / LSI
  - do not outperform embedding-based top-k approaches
  - have higher computational costs, but no thresholds
    → trade-off decisions

ardoco.de/c/aire25



Backup

Evaluation: Reduction of Search Space & Maintaining Recall



Evaluation: Comparison to State of the Art Approaches

- Approaches for Comparison
  - VSM: IR with similarity threshold
  - LSI: IR with similarity threshold
  - Embeddings: IR with Top-K
  - LiSSA: RAG-based with Top-K
- Metrics
  - Precision
  - Recall
  - F_1-score (esp. full automation)
  - F_2-score (esp. semi-automatic)

Approach	Precision	Recall	F_1-score	F_2-score
VSM_{GO}	.22	.56	.27	.34
LSI_{GO}	.17	.61	.23	.33
$Embeddings_{GO}$	.30	.61	.40	.50
LiSSA (GPT-4o)	**.52**	.52	**.50**	**.51**
Majority Voting	.10	**.95**	.18	.34
Chaining	.18	.88	.28	.45
Chaining + GPT-4o	.25	.77	.34	.50

Research Questions

- RQ1:
  To what extent does the **performance** of IR techniques for TLR is **affected by thresholds** or **top-k**?

- RQ2:
  To what extent can **small LLMs** effectively **reduce the search space** for inter-requirements traceability?

- RQ3:
  How does an LLM **ensemble compare** to **existing** retrieval-based methods for TLR?