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

Dominik Fuchß*, Stefan Schwedt†, Jan Keim*, Tobias Hey*

*Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{dominik.fuchss, jan.keim, hey}@kit.edu
†Heriot Watt University, Edinburgh, Scotland
schwedt@cruise-ac.net

***Abstract*—[Introduction] Requirements traceability is essential in software development, supporting tasks such as system understanding and change impact analysis. Traceability link recovery (TLR) methods, including those using large language models (LLMs) or retrieval-augmented generation (RAG), often rely on information retrieval (IR) to identify candidate artifact pairs. They are sensitive to hyperparameters (e.g., top-k, similarity thresholds) that require extensive, project-specific tuning.**

[Methods] We propose an inter-requirements TLR approach that uses an ensemble of small LLMs (or small language models (SLM)) to incrementally reduce the search space, aiming to replace IR-based candidate selection. We first analyze the sensitivity of IR methods to hyperparameters, then evaluate the ability of small LLMs to filter unrelated requirement pairs, and compare their performance when integrated into a TLR approach. The evaluation includes five projects from the requirements engineering community.

[Results] We find that IR performance heavily depends on project-specific hyperparameter tuning. Furthermore, small LLMs effectively reduce the candidate space with minimal loss of recall. While our LLM-based ensemble approach achieves comparable F2-scores to IR methods, it lags in precision.

[Conclusion] This work provides insights into the capabilities of small LLMs as a filter in inter-requirements TLR. Moreover, it provides insights into the performance of traditional IR techniques for TLR and their dependency on hyperparameters.

***Index Terms*—Traceability Link Recovery, Requirements Traceability, Requirements Engineering, Large Language Model, Small Language Model, Retrieval-Augmented Generation**

I. Introduction

In software development, a variety of interrelated artifacts are created. Understanding the relation between these artifacts is crucial for understanding the system and for supporting tasks such as change impact analysis and software reusability [1], [2]. To identify these relationships, traceability link recovery (TLR) approaches frequently employ information retrieval (IR) techniques. These range from traditional methods such as the vector space model (VSM) [3] and latent semantic indexing (LSI) [4] to approaches that combine different techniques and knowledge from different artifacts [5], [6]. More recent approaches use large language models (LLMs) and retrieval mechanisms [7], [8].

However, the effectiveness of these methods heavily depends on the quality of the retrieval, which is often constrained by parameters such as the top-k candidates or a predefined similarity threshold. Importantly, these hyperparameters are often project-specific and require careful tuning to achieve optimal performance [8].

To overcome these limitations, we explore an *ensemble approach* using LLMs to incrementally reduce the search space. This idea builds on successful applications in related domains. For instance, the Crowd and User Informed Suggestion Engine for Acceptance Criteria (CrUISE-AC) uses ensembles of LLMs to link bug reports to user stories [9]. Our approach adapts this concept for inter-requirements traceability.

The core idea is to use smaller, faster LLMs to pre-filter candidate pairs of high-level requirements (HLRs) and low-level requirements (LLRs), eliminating those unlikely to be related. In this context, *small LLMs* (also known as *small language models (SLM)*) refer to models with fewer than 15 billion parameters, which can typically be run on consumer-grade hardware. We introduce a layered architecture in which multiple LLMs are applied in succession. Using a simple prompt to identify potentially linked artifacts, the approach progressively narrows down the candidate space for more expensive evaluations. We integrate this filtering step into the Linking Software System Artifacts (LiSSA) framework [7], [8] as a replacement for its retrieval step. LiSSA then uses a single LLM to classify the remaining candidates.

Finding the optimal thresholds or k values for IR techniques is not always feasible in practice, as it requires data for tuning. As we expect non-optimal parameters to have a non-negligible impact, we see an advantage in a fully automated replacement. To back that up, we first assess the impact of thresholds or k values on the performance of IR techniques. Therefore, we formulate the first research question:

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

The combinatorial complexity based on the number of artifacts in a project can be high. Thus, the small LLMs have to be able to reduce the search space to save resources. To

assess the feasibility of our proposed approach, our second question investigates how well filtering can reduce the number of candidates while maintaining a high recall:

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

Finally, we assess the performance of our approach in comparison to state-of-the-art and baseline approaches:

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

Our contributions are insights into the performance of traditional and embedding-based IR techniques for TLR and their dependency on hyperparameters, an evaluation of the filtering capabilities of small LLMs for inter-requirements traceability, and a comparison of how this filtering combined with an LLM classifier performs against other methods in the context of inter-requirements traceability.

II. Related Work

A. Requirements Traceability Link Recovery

TLR between requirements has been widely studied over the past decades. Early work relied on IR methods such as VSM [3], LSI [4], and probabilistic network models [10], often enhanced with resources like glossaries [10], thesauri [3], [11], and ontologies [12]. Some approaches combined IR methods [13], included syntactic information combined with spreading activation [14], or used intermediate artifacts [15].

More recently, approaches applied word embeddings [16]–[18] and machine learning [19]–[22], though most require pre-existing links to train or fit their approach on.

Most of the latest approaches make use of LLMs. Lin et al. [23] compare domain- and task-adapted variants of the LLM BERT, showing that task-specific training on GitHub data yields the best trace link completion and expansion performance. Rodriguez et al. [24] demonstrate that prompt engineering, especially so-called chain-of-thought (CoT) prompting, impacts TLR results. Based upon the insights of Rodriguez et al., our retrieval-augmented generation (RAG)-based inter-requirements TLR approach [7] using GPT-4o and CoT showed significant improvements over baseline and related approaches. This approach, however, is dependent on the retrieval performance, which is limited by the selected top-k value. We target this issue in this work.

B. LLM Classifier and Ensemble Approaches

The use of classifier ensembles, where multiple base classifiers are combined to form a single predictive model [25], is a well-established strategy in classification tasks. Extensive research has shown that such ensemble methods often yield superior classification performance compared to individual classifiers (see, e.g., [26]). Recent work by Chen et al. [27] explores various ensemble techniques with LLMs, demonstrating that aggregating predictions from multiple LLMs through majority voting can outperform single-model approaches and, in some cases, even surpass more sophisticated methods such as CoT pipelines [28]. The approach of assembling multiple LLMs to improve classification performance has also been applied in requirements engineering. For example, Schwedt et al. [9] implemented an LLM ensemble to successfully establish links between bug reports and user stories.

III. Approach

This section presents our approach for inter-requirements TLR using an ensemble of LLMs as an alternative to classical IR methods. First, we give an overview of the LiSSA framework in Section III-A that we extended for our experiments. In Section III-B, we describe the ensemble of small LLMs and the configurations under investigation.

A. LiSSA Framework

LiSSA, the Linking Software System Artifacts framework [8], is designed to enable different TLR tasks, i.e., by identifying links between different artifacts like requirements, documentation, or code. The framework consists of two main steps: retrieval and mapping. In the retrieval step, LiSSA uses an embedding-based retrieval method to identify candidate pairs of artifacts that are likely to be linked. The retrieval is based on a pre-trained embedding model that maps the artifacts into a vector space, where similar artifacts are close to each other. In the mapping step, LiSSA uses an LLM to classify the candidate pairs as linked or not linked. In this paper, we focus on replacing the retrieval step with an ensemble of LLMs that reduces the search space. We describe the details of the ensemble in the next section.

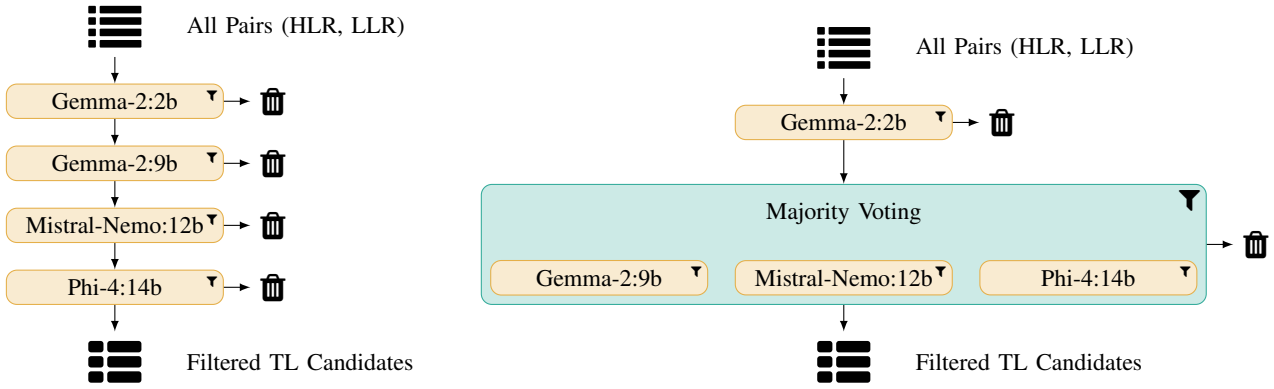
In previous work, LiSSA has been applied to the task of inter-requirements TLR [7] and outperformed state-of-the-art approaches. To ensure comparability with previous work, we use the same prompts. The first prompt is a simple classification prompt that asks the LLM to classify the candidate pairs as linked or not linked — the KISS prompt (Prompt 1). The second prompt is more complex CoT, asking the LLM to reason about the link between the artifacts (Prompt 2).

Prompt 1: KISS

Question: Here are two parts of software development artifacts.
{source_type}: '"{source_content}"'
{target_type}: '"{target_content}"'
Are they related?
Answer with 'yes' or 'no'.

Prompt 2: Chain-of-thought

Below are two artifacts from the same software system. Is there a traceability link between (1) and (2)? Give your reasoning and then answer with 'yes' or 'no' enclosed in <trace> </trace>.
(1) {source_type}: '"{source_content}"'
(2) {target_type}: '"{target_content}"'



(a) Chaining Approach

(b) Majority Voting Approach

Fig. 1. Two approaches to filter trace link candidates using a ensemble of LLMs. On the left, the chaining approach is shown, where each LLM filters the candidates and passes them to the next LLM. On the right, the majority voting approach is shown, where multiple LLMs vote on the candidates and the majority decides whether a candidate is kept or not.

B. Ensemble LLM-based Approaches for Retrieval

Figure 1 shows our two approaches to filter the search space for matching requirements, which we detail in the following.

1) Chaining Approach: Figure 1a depicts the chaining strategy for LLMs. This method uses the LLMs sequentially to consecutively filter the set of candidate requirements. Each LLM forwards only those identified as potential matches to the next LLM in the chain. The models are arranged in order of increasing parameter size, beginning with the smallest and concluding with the largest. This design aims to optimize both computational efficiency and overall performance: smaller LLMs, being faster and less resource-intensive, handle the initial broad filtering, while larger, potentially more capable models focus on a reduced subset of candidates. All LLMs in the chain use the same KISS prompt (Prompt 1), instructing them to classify whether a candidate pair is linked.

2) Majority Voting Approach: Figure 1b shows the majority voting strategy. Here, multiple LLMs independently classify each candidate pair, and their decisions are combined using a majority vote. Since every LLM processes the same set of candidates, computational costs scale with the number of involved models. To balance classification quality and efficiency, we adopt a layered approach: a small, fast LLM first filters the candidate pairs, and only the reduced set is passed to the other models. In the second layer, three LLMs evaluate the remaining pairs, and a candidate is accepted as linked if at least two out of three models agree. Like the chaining approach, all LLMs use the KISS prompt (Prompt 1) at each stage.

IV. Evaluation

In this section, we present the evaluation. We first introduce our setup, including the datasets in Section IV-A, the used LLMs in Section IV-B, and the applied metrics in Section IV-C. We then discuss the results in Section IV-D.

A. Datasets

To compare our results with results that are obtained from using embedding-based retrieval and other approaches, we utilize commonly used datasets from the TLR community. Table I gives an overview of the datasets. We took the datasets CM1-NASA, GANNT, MODIS, and WARC from the repository of the *Center of Excellence for Software & Systems Traceability* (CoEST) [29]. Additionally, we make use of the Dronology dataset [30] that we also used for evaluating our previous RAG-based approach [7]. All datasets comprise HLR and their respective LLR, i.e., their refinements.

TABLE I
Overview of the Datasets. Datasets Comprise High-level Requirements (HLR) and Low-level Requirements (LLR). Percentage of Linked Source and Target Artifacts in the Gold Standard Is Given in Brackets.

		Number of Artifacts				
Dataset	Domain	HLR		LLR		TLs
CM1-NASA	Aerospace	22	(86%)	53	(57%)	45
Dronology	Aerospace	99	(96%)	211	(99%)	220
GANNT	Proj. Mngmt.	17	(100%)	69	(99%)	68
MODIS	Aerospace	19	(63%)	49	(63%)	41
WARC	Archiving	63	(95%)	89	(89%)	136

It is worth noting that the gold standards for the CM1 and the MODIS dataset only cover a low number of LLR. This either means the set of source/target requirements is incomplete or the gold standard is incomplete, assuming that a refinement should have a corresponding high-level requirement.

B. Language Models

We use different LLMs in this evaluation. We decided to use models by OpenAI and locally deployed open-source models. As mentioned in Section III-B, we use smaller models for the filtering, as they are faster and require fewer resources: *Gemma-2:2b*, *Gemma-2:9b*, *Mistral-Nemo:12b*, and *Phi-4:14b*. For the final classification step, we use larger models that have already shown good performance for TLR tasks [7]: *GPT-4o mini* (*gpt-4o-mini-2024-07-18*) and *GPT-4o* (*gpt-4o-2024-08-06*). We ensured that the versions of OpenAI's models used in this work are identical to those used

TABLE II
VSM, LSI, AND EMBEDDING-BASED RETRIEVAL RESULTS ACROSS PROJECTS. GO DENOTES GLOBALLY OPTIMZED. PO DENOTES PROJECT OPTIMIZED.

Configuration	CM1-NASA			Dronology			GANNT			MODIS			WARC			Average			
	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_1	F_2
$Embeddings_{GO}$	.30	.73	.57	.32	.73	.58	.45	.56	.53	.13	.29	.23	.32	.75	.59	.30	.61	.40	.50
VSM_{GO}	.29	.53	.46	.08	.95	.31	.30	.40	.37	.34	.27	.28	.10	.65	.30	.22	.56	.27	.34
LSI_{GO}	.17	.60	.40	.10	.94	.35	.26	.44	.39	.26	.32	.30	.05	.77	.19	.17	.61	.23	.33
$Embeddings_{PO}$	.30	.73	.57	.39	.70	.60	.54	.54	.54	.09	.78	.30	.32	.75	.59	.33	.70	.42	.52
VSM_{PO}	.34	.51	.47	.37	.65	.56	.17	.65	.42	.16	.51	.35	.12	.52	.31	.23	.57	.32	.42
LSI_{PO}	.22	.53	.42	.36	.62	.54	.13	.78	.39	.12	.59	.33	.08	.52	.25	.18	.61	.27	.39

TABLE III
THRESHOLDS / TOP-K OPTIMIZED FOR HIGH F_2-SCORE BY PROJECT AND APPROACH. *Globally Optimized (GO)* SHOWS THE VALUES TO ACHIEVE THE BEST AVERAGE F_2-SCORE ACROSS ALL PROJECTS.

Project / Approach	Threshold for VSM	Threshold for LSI	k value for Embeddings
CM1-NASA	.12	.20	5
Dronology	.35	.43	4
GANNT	.06	.08	4
MODIS	.04	.07	19
WARC	.14	.29	5
Globally Optimized	.11	.17	5

in our previous work. We used a machine equipped with a Tesla V100S GPU, 32 GB of VRAM, 112 CPUs (2.7 GHz), and 256 GB of RAM to execute the experiments with the open-source models. However, the models can also be run on consumer hardware. For the GPT-based models, we used OpenAI's API.

C. Metrics

As our approach regards the task of recovering trace links as a classification task, we make use of classification metrics that are commonly applied to evaluate the performance of TLR approaches [4], [31]: precision, recall, and the F_β-score. We choose $\beta = 2$, as for semi-automatic TLR, recall should be valued higher than precision to prevent too many missed links. However, we also provide the F_1-score, since fully automating TLR requires both good precision and recall [4].

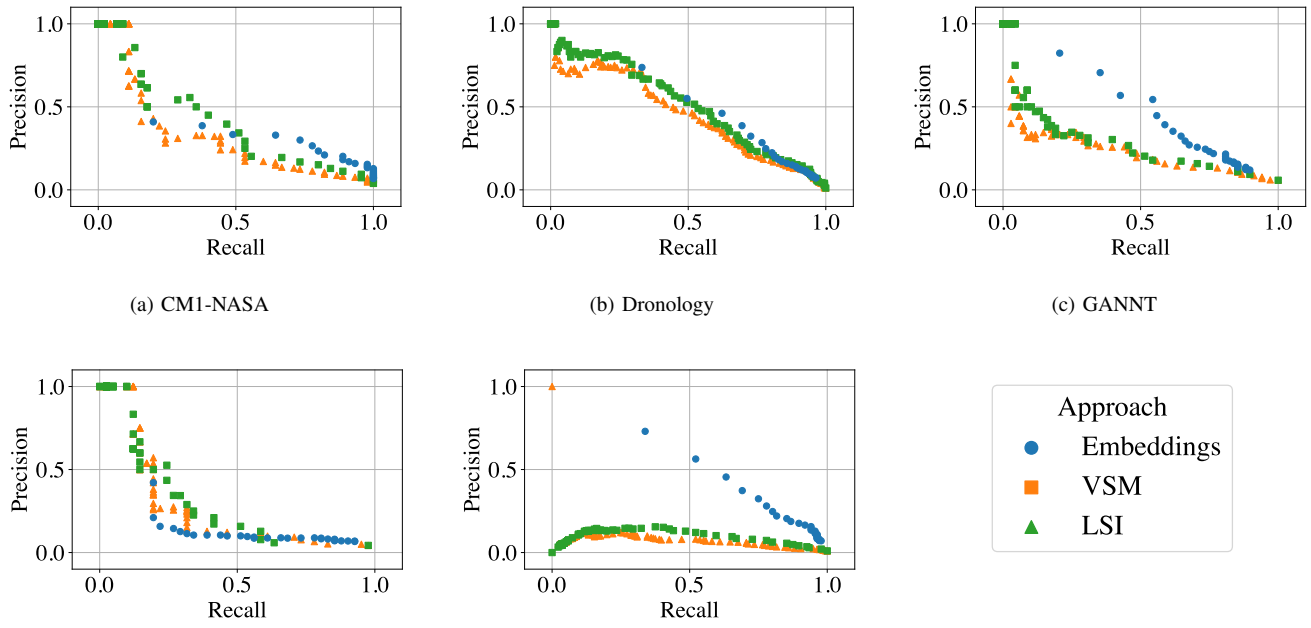
D. Results

In this section, we present our results structured by our research questions. Afterward, we discuss our threats to validity.

1) Impact of Top-K and Thresholds: This section covers the impact of thresholds, respectively, the value k for Top-k retrieval. To get corresponding insights, we analyze the results of our previous work [7] regarding different values for k and thresholds. For the baselines VSM and LSI, we analyze the impact of the similarity threshold, ranging from 0.01 to 1.0 in steps of 0.01. For the embedding-based TLR approach that relies on OpenAI's *text-embedding-3-large*, we analyze the impact of the k value, ranging from 1 to 30 in steps of 1. We calculated the precision and recall for each of these configurations. Figure 2 shows the results. The graphs show the relationship between precision and recall for the different configurations. The approaches' performance is dependent on the project. For example, the embedding-based retrieval performs better than VSM and LSI for WARC, as the curve is above the other two approaches. This is similar to GANNT. However, the performances of the approaches are more similar for the other projects. Lastly, the graphs show that the performance of the three approaches varies based on the different thresholds and values for k. This suggests that the choice of thresholds and k values impacts performance.

Based on the overall analysis of the results, our goal is to identify optimal configuration settings for each approach. We particularly focus on the F_2-score as the primary metric for optimization. We distinguish two kinds of configurations: The first type is the *Globally Optimized F_2* (GO) configuration, which returns the highest average F_2-score across all projects. The second kind are the *Project Optimized F_2* (PO) configurations, which are individually optimized for each project to return the best F_2-score. The specific values for k and the thresholds can be found in our replication package [32]. For our experiments, the average best threshold (i.e., *GO* configuration) for VSM was 0.11 and for LSI it was 0.17. The average best k value for the embedding-based retrieval was 5. If we optimize per project (*PO* configuration), the best thresholds for VSM range in $[0.04, 0.35]$ and for LSI in $[0.07, 0.43]$. For the embedding-based retrieval, the best k values per project range in $[4, 19]$. The thresholds and k values are summarized in Table III. The optimal configurations vary across projects, and the choice of thresholds and k values impacts the performance.

Table II shows the results of the optimal configurations, displaying the corresponding precision, recall, and F_2-score. Moreover, the table shows the average scores across all projects, including the F_1-score as well. It is important to note that the average scores for the *PO* configurations require tuning the thresholds and k values for each project individually, making them less realistic as users typically do not know the optimal configuration. However, these scores indicate an upper bound for the performance. The difference between the *GO* and *PO* configurations also demonstrates the importance of tuning the thresholds and k values. For example, VSM has a difference of 0.08 average F_2-score between the *GO* and *PO* configurations.



(a) CM1-NASA (b) Dronology (c) GANNT

(d) MODIS (e) WARC

Fig. 2. Precision-Recall Graphs for the different projects. The x axis shows the recall, the y axis the precision. The colors indicate the different approaches. Each dot corresponds to a different configuration.

TABLE IV
FILTERING CAPABILITIES OF THE DIFFERENT MODES

Mode	CM1-NASA			Dronology			GANNT			MODIS			WARC			Average			
	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_1	F_2
Gemma-2:2b	.04	1.0	.19	.01	1.0	.05	.08	1.0	.29	.08	.90	.31	.03	1.0	.12	.05	.98	.09	.19
Gemma-2:9b	.04	1.0	.17	.01	1.0	.06	.06	.99	.25	.06	1.0	.25	.03	1.0	.12	.04	1.0	.08	.17
Mistral-Nemo:12b	.08	1.0	.32	.04	.96	.17	.15	.99	.47	.16	.78	.45	.08	.92	.29	.10	.93	.18	.34
Phi-4:14b	.14	1.0	.46	.08	.97	.30	.18	.99	.51	.27	.59	.47	.12	.97	.41	.16	.90	.26	.43
Chaining	.15	1.0	.47	.08	.93	.30	.20	.97	.54	.32	.56	.49	.14	.91	.43	.18	.88	.28	.45
Majority Voting	.09	1.0	.32	.04	1.0	.17	.14	.99	.45	.17	.78	.45	.07	.98	.29	.10	.95	.18	.34

Conclusion RQ1: The performance of IR techniques for TLR is highly dependent on the choice of thresholds and k values. The optimal configurations vary across projects.

2) Filtering Capabilities of Small LLMs: To analyze the filtering capabilities of the small LLMs that we use in our ensemble, we first focus on the performance of the small LLMs in isolation. Then, we look into the performance of the two proposed ensemble modes. Table IV and Figure 3 summarize and visualize the results of the filtering capabilities.

In the Table IV, we show the precision, recall, and F_2-score for each of the small LLMs and the two ensemble modes. The filtering performance varies across the different models. While both Gemma-2:2b and Gemma-2:9b filter few candidates, Mistral-Nemo:12b and Phi-4:14b filter the most. However, the larger Gemma model does not filter more candidates than the smaller one. Overall, the LLMs can maintain a high recall except for MODIS, where there is a low recall using Mistral-Nemo:12b and Phi-4:14b.

Considering the ensemble modes, the chaining model, as expected, filters the most candidates. However, the average recall remains high, averaging 0.88 across all projects. Majority voting performs worse regarding F_1- and F_2-score, while the chaining mode achieves the best average F_2- and F_1-score.

Conclusion RQ2: Small LLMs can filter out many candidates while preserving high recall. The most effective mode is chaining, which eliminates the most candidates without compromising recall.

3) Ensemble of LLMs vs. Retrieval for TLR: In this final result section, we compare the performance of the best ensemble of LLMs to the retrieval-based approach from our previous

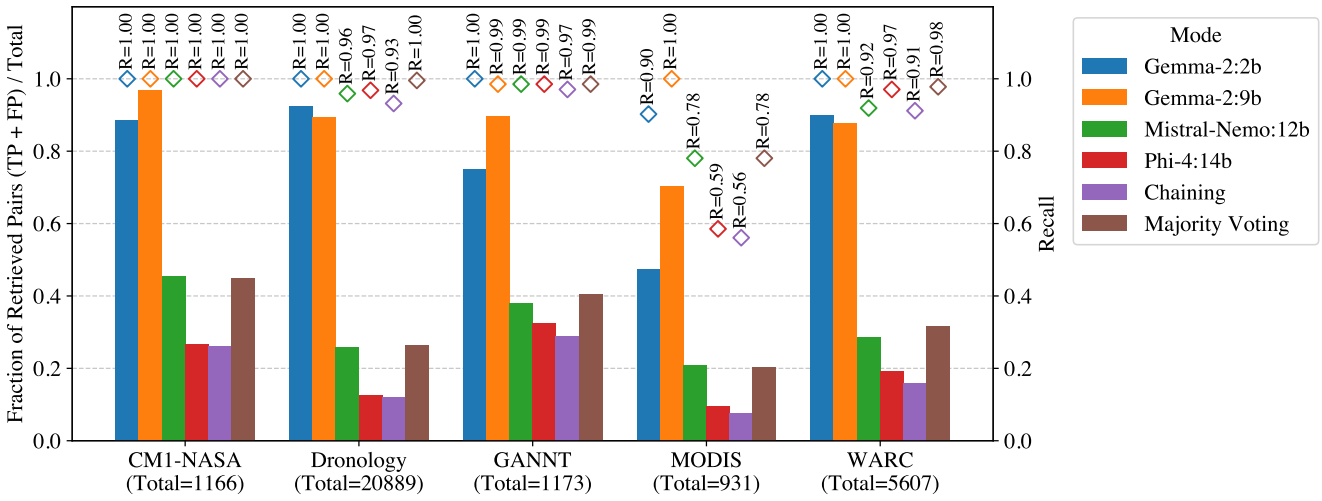


Fig. 3. Filtering capabilities of the different modes. The x-axis shows the different projects. The colors indicate the different modes. The y-axis shows the fraction of predicted positive candidates and all candidates. The diamond marks the recall. The lower the fraction, the better. The higher the recall, the better.

TABLE V
PERFORMANCE OF DIFFERENT CONFIGURATIONS ACROSS PROJECTS

Configuration	**CM1-NASA**			**Dronology**			**GANNT**			**MODIS**			**WARC**			**Average**			
	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_2	P	R	F_1	F_2
Embeddings [7]																			
$k = 4$	.33	.64	.54	.39	.70	.60	.54	.54	.54	.14	.27	.23	.37	.69	.59	.36	.57	.43	.50
+ GPT-4o (CoT)	.46	.60	.56	.51	.65	.62	.61	.54	.56	.50	.17	.20	.54	.64	.62	.52	.52	.50	.51
$\text{Embeddings}_{\text{GO}}$	.30	.73	.57	.32	.73	.58	.45	.56	.53	.13	.29	.23	.32	.75	.59	.30	.61	.40	.50
VSM_{GO}	.29	.53	.46	.08	.95	.31	.30	.40	.37	.34	.27	.28	.10	.65	.30	.22	.56	.27	.34
LSI_{GO}	.17	.60	.40	.10	.94	.35	.26	.44	.39	.26	.32	.30	.05	.77	.19	.17	.61	.23	.33
Chaining	.15	1.0	.47	.08	.93	.30	.20	.97	.54	.32	.56	.49	.14	.91	.43	.18	.88	.28	.45
+ GPT-4o mini (CoT)	.16	.98	.49	.09	.91	.33	.21	.97	.56	.34	.56	.50	.15	.89	.44	.19	.86	.29	.46
+ GPT-4o (CoT)	.27	.93	.62	.16	.86	.45	.24	.91	.58	.35	.29	.30	.21	.86	.53	.25	.77	.34	.50

work [7]. We focus on the chaining mode since it outperformed the majority voting mode in both F_1 and F_2-scores. Furthermore, we compare the performance to the globally optimized configurations of the VSM, LSI, and embedding-based retrieval.

Table V shows the results of our experiments. The table is structured into multiple sections. The first section shows the baseline results of our previous work [7] using RAG with embedding-based retrieval. Here, the table shows the results for the embedding-based retrieval with a fixed $k = 4$ and with subsequent classification using the CoT prompt (Prompt 2). The second section of the table shows the globally optimized configurations of the VSM, LSI, and embedding-based retrieval. Lastly, the third section shows the results of the chaining mode with subsequent classification using a CoT prompt. We applied the models *GPT-4o mini* and *GPT-4o* for the classification step, as we also used them in previous work.

On average, both ensemble modes with classification outperform the VSM and LSI baselines in F_1-score and F_2-score. Considering the F_2-score, the ensemble approach with GPT-4o performs comparably to the embedding-based approaches. However, the ensemble approaches have a lower precision than the best-performing embedding-based + GPT-4o approach.

Conclusion RQ3: The ensemble can outperform baselines based on VSM and LSI in F_1-score and F_2-score. It is unable to outperform the embedding-based retrieval approaches in terms of precision.

E. Threats to Validity

Based on the guidelines by Runeson and Höst [33] and the identified challenges of using LLMs in software engineering as discussed by Sallou et al. [34], we outline and address potential threats to the validity of our research and experiments.

The choice of prompts poses a potential threat to construct validity. In this work, we intentionally did not focus on prompt engineering, as our objective is to provide initial exploratory insights into the use of small LLMs as filters for TLR. To mitigate the risk, we reused prompt sets from prior studies [7], [8] that are based on the work by Rodriguez et al. [24].

Confounding factors may influence our findings and the conclusions drawn from them. To reduce this risk, we adhered to established practices. We utilized widely recognized benchmark datasets, thereby minimizing the possibility of selection bias, and clearly stated the origins of the included projects.

We evaluated our approach on a limited number of projects, which may limit the generalizability of our findings. To mitigate this threat, we selected benchmark datasets commonly used in TLR research that span different domains and project sizes. However, many of them are comparably old and may not fully capture the challenges of modern traceability scenarios.

Another limitation arises from the restricted set of evaluated LLMs that does not allow us to generalize to other LLMs.

A further threat stems from our use of *closed-source models*. Because their training data is not publicly disclosed, we cannot rule out potential *data leakage*.

Finally, LLMs are inherently non-deterministic. To reduce this threat, we fixed the random seed and set the model temperature to 0. Our results are therefore dependent on the specific seed 133742243 that we reused from prior work.

V. Discussion

Our evaluation reveals a trade-off between precision and recall between the ensemble methods and embedding-based IR techniques.

On the one hand, the ensemble approach, e.g., using chaining, consistently achieves high recall, which is valuable in scenarios where missing relevant links is particularly important. However, this comes at the cost of lower precision, resulting in a decreased average F_1-score, even though the average F_2-score remains comparable to other approaches due to the high recall. Still, there is potential for future improvements.

The ensemble approach is computationally more demanding and, thus, more expensive. It involves many calls to multiple LLMs, increasing both inference time and resource consumption. This makes it less practical in environments where computational efficiency or scalability are primary concerns.

On the other hand, embedding-based and traditional IR methods are generally more efficient and cost-effective. However, they rely on similarity thresholds or the selection of the top-k candidates, parameters that heavily influence performance. Due to the sensitivity of these parameters to the specific context or dataset, optimal configurations vary across projects and are not easily predictable without prior tuning. As such, these approaches may underperform in practical, real-world settings if default or suboptimal values are used.

Therefore, declaring one approach as universally superior is not straightforward. The choice depends heavily on the intended use case and available resources. In semi-automated settings, where human reviewers can refine candidate trace links, the ensemble approach may be more desirable due to its higher recall, which ensures fewer missed links. In contrast, for fully automated pipelines where manual validation is infeasible, embedding-based approaches might be more appropriate, offering a more favorable balance between precision and cost.

VI. Conclusion

In this work, we investigated the retrieval component of a state-of-the-art TLR approach for inter-requirements traceability. Our investigation was motivated by the limitations of traditional information retrieval (IR) techniques, which typically demand extensive, project-specific tuning of hyperparameters such as similarity thresholds and top-k values. These parameters impact performance but often lack generalizability across diverse project contexts.

We began by analyzing the impact of these hyperparameters on the performance of IR techniques. Subsequently, we evaluated an ensemble approach that uses multiple lightweight LLMs to incrementally reduce the search space for inter-requirements trace links. Our goal was to assess whether this ensemble could effectively replace the retrieval component of a TLR approach based on RAG with embedding-based retrieval. To this end, we first investigated the ensemble's filtering capabilities, followed by a comparison of the overall TLR performance against conventional retrieval methods. Our evaluation was conducted using five publicly available benchmark datasets widely adopted in the TLR community.

Our results demonstrate that small LLMs are effective in filtering a substantial number of non-linked candidate pairs while preserving high recall, particularly when using a chaining-mode where filters are applied sequentially across layers. In terms of effectiveness, the ensemble outperforms traditional baselines based on vector space model (VSM) and latent semantic indexing (LSI) in both F_1- and F_2-scores. However, it does not exceed the precision of recent embedding-based retrieval methods. While the ensemble's multi-model architecture incurs higher computational costs compared to single-model alternatives, it eliminates the need for hyperparameter tuning. This trade-off makes it particularly suitable when such tuning is difficult, costly, or infeasible.

Future research can build on this work in several directions. One natural extension is to evaluate the proposed approach on broader traceability tasks beyond inter-requirements links, such as *requirements-to-code* TLR, where search space reduction remains a key challenge. Our current method utilizes fixed prompt templates derived from earlier studies; future work could explore more adaptive prompt engineering techniques, specifically tailored to filtering and ranking in traceability contexts. Another promising direction involves a deeper investigation into ensemble design choices, including the number and sequencing of LLM layers, the used models, as well as strategies for combining their outputs (e.g., majority voting, weighted aggregation, or confidence-based filtering). Finally, we see potential in hybrid approaches that integrate LLM-based filtering with traditional IR techniques. By combining the semantic reasoning capabilities of LLMs with the efficiency and scalability of classical methods, while mitigating the limitations of each, such approaches may offer more robust and adaptable solutions to traceability link recovery across diverse software engineering artifacts and domains.

Data Availability Statement

The code, datasets, and results are publicly available in our replication package [32].

Acknowledgements

This work was funded by Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF) and supported by the German Research Foundation (DFG) - SFB 1608 - 501798263 and KASTEL Security Research Labs.

References

[1] J. Cleland-Huang, O. Gotel, A. Zisman *et al.*, *Software and Systems Traceability*. Springer, 2012, vol. 2.

[2] F. Tian, T. Wang, P. Liang, C. Wang, A. A. Khan, and M. A. Babar, "The impact of traceability on software maintenance and evolution: A mapping study," *Journal of Software: Evolution and Process*, vol. 33, no. 10, 2021.

[3] J. H. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, Sep. 2003, pp. 138–147.

[4] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods," *IEEE Trans. Softw. Eng.*, vol. 32, no. 1, pp. 4–19, Jan. 2006.

[5] J. Keim, S. Corallo, D. Fuchß, T. Hey, T. Telge, and A. Koziolek, "Recovering trace links between software documentation and code," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024.

[6] D. Fuchß, H. Liu, T. Hey, J. Keim, and A. Koziolek, "Enabling Architecture Traceability by LLM-based Architecture Component Name Extraction," in *2025 IEEE 22nd International Conference on Software Architecture (ICSA)*. Institute of Electrical and Electronics Engineers (IEEE), Apr. 2025.

[7] T. Hey, D. Fuchß, J. Keim, and A. Koziolek, "Requirements Traceability Link Recovery via Retrieval-Augmented Generation," in *Requirements Engineering: Foundation for Software Quality*. Cham: Springer, Apr. 2025.

[8] D. Fuchß, T. Hey, J. Keim, H. Liu, N. Ewald, T. Thirolf, and A. Koziolek, "LiSSA: Toward Generic Traceability Link Recovery through Retrieval-Augmented Generation," in *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, ser. ICSE '25. Institute of Electrical and Electronics Engineers (IEEE), May 2025.

[9] S. Schwedt and T. Ströder, "From Bugs to Benefits: Improving User Stories by Leveraging Crowd Knowledge with CrUISE-AC," in *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, ser. ICSE '25. Institute of Electrical and Electronics Engineers (IEEE), May 2025.

[10] X. Zou, R. Settimi, and J. Cleland-Huang, "Improving automated requirements trace retrieval: A study of term-based enhancement methods," *Empir Software Eng*, vol. 15, no. 2, pp. 119–146, Apr. 2010.

[11] A. Mahmoud and N. Niu, "On the role of semantics in automated requirements tracing," *Requirements Eng*, vol. 20, no. 3, pp. 281–300, Sep. 2015.

[12] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej, "Ontology-based multiperspective requirements traceability framework," *Knowl Inf Syst*, vol. 25, no. 3, pp. 493–522, Dec. 2010.

[13] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang, "Improving Trace Accuracy Through Data-driven Configuration and Composition of Tracing Features," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 378–388.

[14] A. Schlutter and A. Vogelsang, "Improving Trace Link Recovery Using Semantic Relation Graphs and Spreading Activation," in *Requirements Engineering: Foundation for Software Quality*, F. Dalpiaz and P. Spoletini, Eds. Cham: Springer International Publishing, 2021, pp. 37–53.

[15] A. D. Rodriguez, J. Cleland-Huang, and D. Falessi, "Leveraging Intermediate Artifacts to Improve Automated Trace Link Retrieval," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2021, pp. 81–92.

[16] T. Zhao, Q. Cao, and Q. Sun, "An Improved Approach to Traceability Recovery Based on Word Embeddings," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2017, pp. 81–89.

[17] L. Chen, D. Wang, J. Wang, and Q. Wang, "Enhancing Unsupervised Requirements Traceability with Sequential Semantics," in *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2019, pp. 23–30.

[18] T. Hey, F. Chen, S. Weigelt, and W. F. Tichy, "Improving Traceability Link Recovery Using Fine-grained Requirements-to-Code Relations," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2021, pp. 12–22.

[19] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically Enhanced Software Traceability Using Deep Learning Techniques," in *Proceedings of the 39th International Conference on Software Engineering*, ser. ICSE '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 3–14.

[20] W. Wang, N. Niu, H. Liu, and Z. Niu, "Enhancing Automated Requirements Traceability by Resolving Polysemy," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, Aug. 2018.

[21] C. Mills, J. Escobar-Avila, and S. Haiduc, "Automatic Traceability Maintenance via Machine Learning Classification," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2018, pp. 369–380.

[22] C. Mills, J. Escobar-Avila, A. Bhattacharya, G. Kondyukov, S. Chakraborty, and S. Haiduc, "Tracing with Less Data: Active Learning for Classification-Based Traceability Link Recovery," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2019, pp. 103–113.

[23] J. Lin, A. Poudel, W. Yu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Enhancing Automated Software Traceability by Transfer Learning from Open-World Data," no. arXiv:2207.01084, Jul. 2022.

[24] A. D. Rodriguez, K. R. Dearstyne, and J. Cleland-Huang, "Prompts matter: Insights and strategies for prompt engineering in automated software traceability," in *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, 2023, pp. 455–464.

[25] M. Mohandes, M. Deriche, and S. O. Aliyu, "Classifiers combination techniques: A comprehensive review," *IEEE Access*, vol. 6, pp. 19 626–19 639, 2018.

[26] Y. Bi, J. Guan, and D. Bell, "The combination of multiple classifiers using an evidential reasoning approach," *Artificial Intelligence*, vol. 172, no. 15, pp. 1731–1751, 2008.

[27] Z. Chen, J. Li, P. Chen, Z. Li, K. Sun, Y. Luo, Q. Mao, D. Yang, H. Sun, and P. S. Yu, "Harnessing multiple large language models: A survey on llm ensemble," *arXiv preprint arXiv:2502.18036*, 2025.

[28] J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye, "More agents is all you need," *Transactions on Machine Learning Research*, 2024.

[29] CoEST, "Center of Excellence for Software & Systems Traceability." [Online]. Available: http://sarec.nd.edu/coest/

[30] J. Cleland-Huang, M. Vierhauser, and S. Bayley, "Dronology: an incubator for cyber-physical systems research," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 109–112.

[31] Y. Shin, J. H. Hayes, and J. Cleland-Huang, "Guidelines for Benchmarking Automated Software Traceability Techniques," in *2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability*, May 2015, pp. 61–67.

[32] D. Fuchß, S. Schwedt, J. Keim, and T. Hey, "Replication Package for Beyond Retrieval: A Study of Using LLM Ensembles for Candidate Filtering in Requirements Traceability," 2025. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.15837231

[33] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131, 2008.

[34] J. Sallou, T. Durieux, and A. Panichella, "Breaking the silence: the threats of using llms in software engineering," in *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER'24. New York, NY, USA: Association for Computing Machinery, 2024, p. 102–106.