Comparing LLMs and Proposing an ML-Based Approach for Search String Generation in Systematic Literature Reviews

Diogo Adário Marassi
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
dmaraassi@inf.puc-rio.br

Juliana Alves Pereira
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil
juliana@inf.puc-rio.br

Katia Romero Felizardo
Federal University of Technology Paraná (UTFPR)
Cornélio Procópio, Brazil
katiascannavino@utfpr.edu.br

ABSTRACT

The formulation of an effective search string is a critical process in systematic literature reviews (SLRs), as it directly influences both the coverage and precision of the retrieved studies. Traditionally, this process relies on manual keyword selection and expert-driven refinements, making it laborious, susceptible to human bias, and often inaccessible to non-specialists. To address these limitations, this study explores the application of artificial intelligence (AI) to support the generation of search strings. We organized our ongoing investigation into two main phases. In the first phase, we evaluated the performance of search strings generated by different large language models (LLMs), specifically Llama-8B, Gemma-12B, and Mistral-Nemo-12B, using a previously published SLR as a benchmark. Our results suggest that, while LLMs can assist in search string formulation, their effectiveness is inconsistent and sensitive to input conditions. Motivated by these limitations, we propose a semi-automated pipeline based on Machine Learning (ML). Through our preliminary analysis, we proposed a standardized reproducible evaluation framework to assess and compare AI-based search string generation strategies, including our proposed ML-based approach.

KEYWORDS

Systematic Literature Review, Automation, Search String

1 Introduction

One of the main difficulties in conducting string-based searches in a systematic literature review (SLR) context lies in the considerable manual effort involved in formulating and refining search strings, filtering through large volumes of irrelevant articles retrieved, and managing the technical limitations of digital libraries. These factors can hinder the coverage and precision of the evidence-retrieval process. This challenge becomes even more pronounced for researchers and practitioners who are not fully familiar with the target domain. The crafting of effective search strings often requires expert knowledge of domain-specific terminology, synonyms, abbreviations, and evolving vocabulary trends. Without this familiarity, users may overlook key terms or rely on overly generic keywords, which can result in retrieving large numbers of irrelevant studies (low precision) or missing important works entirely (low coverage).

As an alternative, various researchers have proposed automation tools to support the creation and execution of search strings, aiming to retrieve relevant studies from the literature with higher precision and broader coverage [1, 10, 11, 16, 20, 22, 23, 27–30]. Together, these initiatives represent significant progress. However, important challenges remain, particularly in generating search terms that effectively retrieve relevant studies across digital libraries, thus minimizing the volume of irrelevant articles that must be manually screened and excluded during the selection phase.

Large Language Models (LLMs) have shown promise in a wide range of information retrieval and natural language understanding tasks, suggesting that they could serve as powerful tools for automating literature search. However, to the best of our knowledge, most studies using LLMs [2, 34] focus on their ability to generate reasonable queries or summaries but do not provide a comparative analysis that quantifies the quality of the retrieved articles, especially in controlled settings that use real-world SLRs. This gap highlights the need for a systematic and empirical evaluation of LLM-based search string generation. Such an investigation can clarify the strengths and limitations of LLM-based approaches, offering evidence-based guidance for researchers aiming to conduct efficient, reproducible, and thorough SLRs.

This ongoing study is organized into two main phases. In the first phase, we compare the search strings generated by different LLMs using the SLR from [18] as a benchmark to evaluate their precision and coverage. To ensure methodological rigor, we selected an SLR that followed established best practices and widely recognized guidelines [14] for conducting SLRs. This initial assessment revealed important limitations in the use of LLMs, motivating the development of our semi-automated ML-based approach, which is part of our second phase. The second phase begins with a user-provided seed article and applies a pipeline that includes TF–IDF-based keyword extraction, semantic clustering, Boolean string construction, and iterative refinement through automated snowballing. As shown in previous work [35], snowballing is an effective strategy for increasing coverage in SLRs, helping to discover relevant studies that keyword-based searches may overlook, especially those affected by terminology variation or indexing delays. This paper presents our current progress; validation is still underway.

The contributions of this work are as follows: *(i)* An empirical evaluation of how three different LLMs (LLaMA, Mistral, and Gemma) perform in the task of generating search strings for SLRs, using a real-world SLR as the ground truth. *(ii)* An in-depth analysis of prompt engineering strategies, examining how the varying complexity of prompts influences the quality, reproducibility, and robustness of LLM-generated search strings in response to different seed articles. *(iii)* A methodological framework for evaluating AI-assisted search strategies in SLRs, including standardized metrics (precision, coverage, relative precision of review, and F1-score) and a controlled experimental setup. *(vi)* A novel ML-based semi-automated approach to search string generation, offering an interpretable and lightweight alternative to LLMs while maintaining control and transparency in the search string construction process.

2 Related Work

In the context of SLRs, the search for evidence remains one of the most effort-intensive, time-consuming, and error-prone activities, often hindered by the difficulty of achieving both high-precision and broad-coverage search strings and adapting them to multiple digital libraries. Various studies have explored automation to reduce effort and increase reproducibility during SLR activities [3, 6, 35], including generating search strings and performing searches in multiple digital libraries [10, 16, 22, 28].

Visual text mining techniques have been adopted for the formulation of search strings to suggest new terms based on preliminary search results. Although search strings are effectively refined, their application is limited to abstract-level data and specific digital libraries, such as IEEE Xplore [16]. AI techniques, such as Hill Climbing algorithms, have also been applied to search string generation by automatically optimizing search terms based on a set of control studies [28]. These algorithms iteratively adjust synonyms and keyword combinations to improve coverage and precision. However, it remains limited to a single digital library and requires significant setup from the user. In terms of execution, several tools have been proposed to unify the search across multiple digital libraries by formatting the strings according to the library syntax [9].

More recently, researchers have evaluated the potential of LLMs to search for evidence in SLRs [2, 34]. An open issue for investigation is that LLMs are not exclusively trained in scientific databases, therefore, the terms they suggest may not accurately reflect the terminology commonly used by authors in peer-reviewed articles. As a result, the suggested keywords might not retrieve relevant articles. In this paper, we compare different LLM-based approaches and, on the basis of our findings, propose an interpretable and lightweight alternative to LLM.

3 Study Design

We evaluated the precision and coverage of LLM-generated search strings using a real-world SLR case study [18] as a benchmark. To this end, we vary the seed articles, testing five different options. Each search string is generated using only one seed article, in order to assess the potential of assisting novices who are beginning their first SLR. The strings will be executed in scientific databases such as the ACM Digital Library and Scopus, and the retrieved studies will be compared against those identified in the original SLR [18], which used the same databases as their primary sources.

3.1 Research Questions

This study is guided by two RQs as follows:

RQ_1 How do different LLM architectures and levels of prompt complexity affect the quality of search strings and the resulting evidence retrieval in SLR? In this question, we analyze how LLMs perform when generating search strings under varying conditions, specifically, comparing different model architectures (*e.g.*, LLaMA, Gemma, Mistral) and prompt designs (from simple to complex). This allows us to understand how both model choice and prompt engineering affect the coverage and precision of the returned articles.

RQ_2 How consistent are LLM-generated SLR outputs across different variations in seed articles? In this question, we examine the robustness of LLM performance when the input seed articles vary. We assess whether the outputs remain semantically stable and useful or if they are highly sensitive to input perturbations, which would limit their reliability in practical use.

3.2 LLM-Based Approach

This section details the methodology used for the automated generation of search strings using LLMs. We used Llama-8B, Gemma-12B, and Mistral-Nemo-12B due to their open-access availability, moderate computational requirements, and comparable parameter sizes. The LLM receives the title and abstract of the article as input. We chose to use this information instead of full text because, as reported by Van Dinter et al. [33], few automated approaches rely on full-text selection. Moreover, Portenoy and West [19], as well as Dieste and Padua [7], further argue that full-text selection tends to perform worse than approaches based solely on titles and abstracts.

The LLM input was combined with carefully designed prompts under a controlled temperature setting (0.2). Using a low temperature when generating search strings with LLMs reduces the randomness of the output, making the results more deterministic and focused on the most likely options. We made this decision to ensure consistency, reproducibility, and control in the comparison between different models and approaches. The instructions are organized into three levels of increasing prompt complexity (prompt0, prompt1 and prompt2), allowing the evaluation of the adaptive capacity of the model when exposed to different degrees of sophistication in string formulation.

The three prompts used share the same central goal but differ significantly in terms of task complexity, instructional detail, and level of semantic expansion. The main differences are the following.

Task depth and sophistication: `prompt0` presents a more direct and concise formulation, focusing solely on concept extraction and basic semantic expansion. It is suitable for models that already perform well with minimal instruction, but do not guide structured reasoning. `prompt1` introduces a more methodical approach, explicitly describing the steps to guide string construction. `prompt2`, in turn, is the most sophisticated of the three: In addition to describing a structured extraction and expansion process, it simulates an iterative refinement stage, as if the model were evaluating results and adapting the string to improve *coverage*, thereby inducing more intelligent and contextual behavior.

Instruction structure: `prompt0` relies on descriptive, continuous paragraph instructions, whereas `prompt1` and `prompt2` adopt step-by-step modularization, which helps decompose the task and increases reproducibility. This difference reflects a progressive advancement in prompt engineering, with the latter two encouraging the model to follow a more systematic reasoning process.

Emphasis on semantic expansion: Although all prompts encourage the use of related terms, the level of detail and the incentive for semantic expansion increase from `prompt0` to `prompt2`. `prompt0` uses expressions like *"underlying meaning"* and *"related terms"*, but without specifying how to reach them. `prompt1` reinforces this with terms like *"semantically rich"* and provides more guidance on variations, synonyms and common usage forms. `prompt2` stands out by introducing an iterative simulation component: after generating the initial string, the model must use the results retrieved and expand the string accordingly to improve coverage. This additional

layer brings the model's behavior closer to that of a human expert, promoting greater adaptability, and aligning more closely with the refinement technique implemented by the approach developed in this study (see Section 4.3).

All prompts in this study adopt a one-shot prompting strategy [5] with contextual framing. In this setup, the model is presented with a single, clearly defined task, a concise context describing the objective, accompanied by a single example of the expected output in Boolean syntax. No additional explanations or iterative feedback are provided. This single example plays a crucial role in guiding the model's structure and format, helping to standardize the outputs across different models and prompt variations. All prompt templates used in this study are publicly available in our supplementary material [15] to support reproducibility and future research.

With this setup, the LLMs generated the search strings, each of which was used to perform searches in the ACM Digital Library and Scopus databases. The set of articles retrieved from these searches will later be compared with the articles of the original SLR [18] retrieved using a manually created search string, with metrics such as precision, coverage, and F1 score, as defined in Section 3.3.

3.3 Evaluation

Figure 1 illustrates the proposed methodological framework for evaluating AI-assisted search strategies in SLRs. In this figure, we show the methodological flow for one case study, *i.e.* a previously conducted Systematic Review taken as input. For each case study, our evaluation randomly selected five seed articles from the pool of selected articles considered relevant in the original SLR. Each "seed article" is used to generate search strings using an AI-based model. In this study, we used three different LLMs: Llama 3.1:8b, Gemma:12b, and Mistral-nemo:12b. For each article, three prompt variations are tested per LLM, totaling nine distinct generations per seed article (3 models × 3 prompts), totaling 45 strings for a case study (9 generations x 5 seed articles).

Both ACM and Scopus databases were used to retrieve articles for each generated string, as they allow the complete set of search results to be exported. The articles retrieved for each string were collected and compared with the articles retrieved and included in the original SLR [18]. It is important to note that these searches were restricted to the time interval whose upper bound corresponds to the year of publication of the SLR used as reference.

The selected SLR used as a case study [18] provides comprehensive supplementary materials, including structured spreadsheets that document: (i) all articles retrieved using the manually crafted search string; and (ii) the full set of articles reviewed, along with those selected as relevant after peer review (following the application of inclusion and exclusion criteria). These resources enabled the construction of a precise and reliable ground-truth dataset. To support consistent and scalable comparisons, we developed Python scripts to process, normalize, and systematically compare the data exported from the searches with the ground-truth dataset defined by the original SLR. This allowed for automated, reproducible, and accurate computation of key evaluation metrics, based on the degree of overlap between the retrieved articles and the reference set established by expert reviewers.

We compute the metrics listed in Table 1 to quantify the degree to which our automated search string generation reproduces the

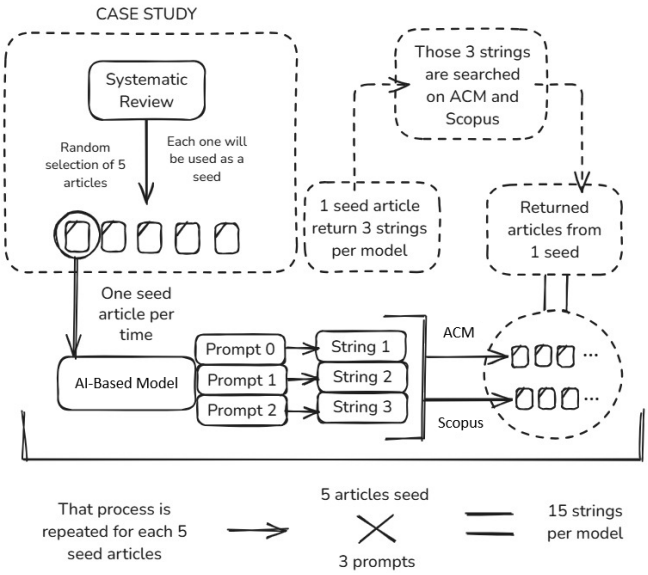


Figure 1: Methodological flow used for evaluation.

results of an expert-crafted search string. Let R: Set of relevant articles selected by the original review. A: Set of articles returned by automated search (AI-based). $R \cap A$: Articles found by both the original review and the AI search (*i.e.*, shared relevant articles). $R \cup A$: Articles from both the original review and the AI search. Next, we define the metrics:

Review-Relative Precision (RP): The fraction of articles selected from the original review that our automated AI search also retrieves.

Precision (P): The fraction of articles returned by the AI search that are truly relevant according to the original review. High precision indicates that the method excludes irrelevant work.

Coverage (C): The proportion of all articles retrieved from the original review (whether selected or not) that appear in the AI search. Reflects on how broadly the AI search spans the space of potentially relevant studies.

F1-Score (F1): The harmonic mean of precision and coverage, balancing accuracy (precision) against completeness (coverage). A strong F1 means an approach that is both selective and thorough.

Table 1: Metrics used to assess AI search strategies in SLRs.

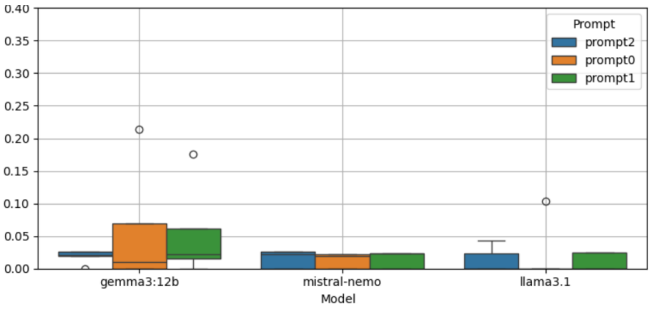
RP	P	C	F1
$RP = \frac{\lvert R \cap A \rvert}{\lvert R \rvert}$	$P = \frac{\lvert R \cap A \rvert}{\lvert A \rvert}$	$C = \frac{\lvert R \cup A \rvert}{\lvert A \rvert}$	$F_1 = 2 \cdot \frac{P \cdot C}{P + C}$

4 Results and Discussion

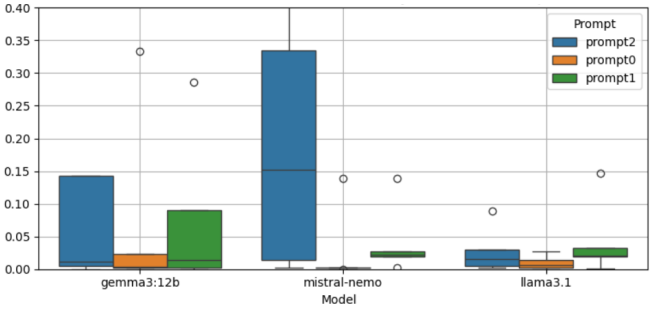
We present and discuss the results related to our RQs as follows. For conciseness, we focus on the best-performing results observed across all input combinations in our case study. An exhaustive set of experimental results, covering all combinations for five seed articles, three LLMs, and three levels of prompt engineering complexity, is available in our supplementary material [15].

4.1 Performance of Models and Prompts (RQ_1)

Figure 2 presents the distribution of F1-Scores obtained across different combinations of LLM models and prompt strategies for (a) the ACM database and (b) the Scopus database. Each model is represented by a group of three boxplots, corresponding to the performance results obtained with each of the three prompt formulations used in our study. Although all models showed poor results in terms of F1-Score, the figure reveals differences in performance depending on the prompt complexity and the model used.



(a) F1-Score – ACM database



(b) F1-Score – Scopus database

Figure 2: F1-Score results for ACM and Scopus databases.

In the ACM database (Figure 2a), the F1-Score between models is relatively low and less differentiated, with no prompt–model combination consistently demonstrating standing out. However, Gemma shows a modest advantage, particularly with prompt0 and prompt1, suggesting some sensitivity to prompt structure. Mistral-Nemo performs uniformly lower, with less variation between prompts. In contrast, the Scopus database (Figure 2b) reveals more pronounced differences. Mistral-Nemo, when combined with prompt2 (the most complex prompt), achieves the highest F1-Score, significantly outperforming other combinations. Conversely, simpler prompts (Prompt0) often failed to retrieve more than the seed article, resulting in near-zero F1-Scores for all models.

The main difference between Mistral-Nemo and LLaMA lies in their attention window. Although both models have a similar number of parameters and are based on the Transformer architecture, Mistral-Nemo:12b supports a context window of 128k tokens, while LLaMA 3.1:8b is limited to just 8k. Mistral-Nemo adopts optimized mechanisms to scale text processing to much larger windows[1], making it more suitable for tasks that involve handling large volumes of textual information. For Scopus, our results suggest that more complex prompts yield better results when the underlying model architecture can leverage that complexity. Simpler models or those not optimized for long-context reasoning fail to benefit similarly from increased prompt sophistication. This is further evidenced by the performance results of LLaMA for all prompts.

However, a notable distinction between the graphics is the limited variation in F1-Score between models and prompts within the ACM database, in contrast to the greater variability observed in Scopus. This raises the question whether such behavior is specific to these platforms or indicative of a broader pattern. To clarify this, we are performing additional experiments using a more diverse set of digital libraries.

Regarding execution time, all tests using Mistral and LLaMA were completed in under 10 seconds. The only exception was the Gemma model with *Prompt2*, which took 54 seconds to complete.

[1]https://mistral.ai/news/mistral-nemo

4.2 Performance with Different Seeds (RQ_2)

To evaluate the consistency of the LLM-generated output, we compare the best F1 obtained for each seed article across the ACM and Scopus databases. Table 2 summarizes these peak performances.

Table 2: Best results from each seed article and Database.

Seed	Base	Model	Prompt	F1 (%)	C (%)	P (%)	RP (%)
[24]	ACM	Mistral	prompt2	7.74	10.84	6.02	71.43
	Scopus	Gemma	prompt1	28.57	28.57	28.57	14.82
[31]	ACM	Gemma	prompt2	0.23	1.03	0.13	14.29
	Scopus	LLaMA	prompt0	0.35	1.44	0.20	11.11
[26]	ACM	Mistral	prompt1	0.20	0.91	0.11	14.29
	Scopus	Mistral	prompt1	1.87	2.34	1.56	29.63
[25]	ACM	Mistral	prompt0	0.19	0.78	0.11	14.29
	Scopus	Mistral	prompt2	48	60.00	40.00	7.41
[4]	ACM	Mistral	prompt2	0.23	1.14	0.13	14.29
	Scopus	Mistral	prompt1	2.72	3.10	2.41	25.93

In general, we observed substantial variability in retrieval quality depending on the chosen seed. In the ACM database, the seed of Sarkar et al. [24] produced the highest RP of 71.43% (with prompt2 in Mistral) and an F1 of 7.74% due to low coverage, indicating that although most articles selected by experts were retrieved, the search breadth was extremely narrow. In contrast, the same seed on Scopus achieved its best F1 of 28.57% (prompt1 on Gemma), with a coverage and precision of 28.57%.

For Siegmund et al. [25], the top Scopus run (Mistral with prompt2) yielded an F1 of 48%, coverage at 60%, precision at 40%, but RP of just 7.41%. In fact, four of the five seeds produced near-zero F1 values in ACM, reflecting extremely limited coverage despite occasional precision gains.

These results suggest that the effectiveness of LLM-based string generation is highly sensitive to the input seed. Seeds rich in domain-specific terminology and well-structured abstracts, such as [25], enable stronger pattern learning and broader retrieval in Scopus, but may still falter in more restrictive collections like ACM. In contrast, seeds with less distinctive keyword profiles produce poor coverage and F1 performance regardless of model or prompt.

This sensitivity poses a challenge for reproducibility: a method that excels in one seed but fails in another cannot be reliably deployed in practice without additional safeguards. Incorporating more seed articles, iterative refinement loops, or mixed-model strategies can help stabilize performance and ensure more robust, comprehensive coverage across diverse SLR topics (see Section 4.3).

Although high-capacity models, such as Mistral-Nemo, which appears seven times from a total of ten in Table 2, exhibit the potential for strong performance, their output consistency remains vulnerable to the choice of input, reinforcing the need for methodological safeguards in SLR automation.

4.3 Proposed ML-Based Approach

Given the previous results, proposing an ML-based search string approach is important because it offers a lightweight, transparent, and interpretable alternative to LLMs, which often operate as black boxes and require significant computational resources. In this section, we describe our proposed approach, which consists of six steps as shown in Figure 3. Our approach can be tailored to specific domains, is easier to reproduce, and allows users to trace and adjust each step of the search string generation process. Note that a refinement step based on automatic snowballing was added due to the significant variation in the LLM results as shown in Section 4.

Starting from a seed article provided by the user in PDF format, the process begins with metadata extraction (title and abstract), followed by keyword identification. An initial search string is then generated and validated by the user, who assesses the relevance of the suggested terms and synonyms. The search is executed on Semantic Scholar, and the five most relevant candidate studies are selected according to Semantic Scholar's relevance metric. Snowballing is then applied to the five selected articles, and the newly identified candidates are filtered again, with five additional studies selected, resulting in a total of ten included studies per iteration. The metadata (titles and abstracts) of these new articles is then used to extract additional keywords and generate a refined search string. This iterative process continues until the search string meets the user-defined criteria of quality and completeness. Once finalized, the search string is executed in databases such as ACM and Scopus.

(1) Preprocessing. The generation of the search string begins with a seed article and its metadata extraction. We used the open-source tool CERMINE[2] (Content ExtRactor and MINEr), which extracts metadata and content from scientific PDF articles. This tool employs techniques from Natural Language Processing (NLP), computer vision, and ML to identify and structure the topics addressed in the article. In this study, we extracted the title and abstract, as these sections offer the most relevant information about the core topic. Conventional and academic-specific stop words were then removed, using a list available in the project's GitHub repository [15].

(2) Keyword identification. With the title and abstract corpus, we extracted keywords using a three-stage algorithm:

(i) TF-IDF (Term Frequency–Inverse Document Frequency) application. TF-IDF is an NLP technique that assesses the relevance of a term within a specific corpus [21]. However, instead of applying the technique directly to individual words, we applied it to the most

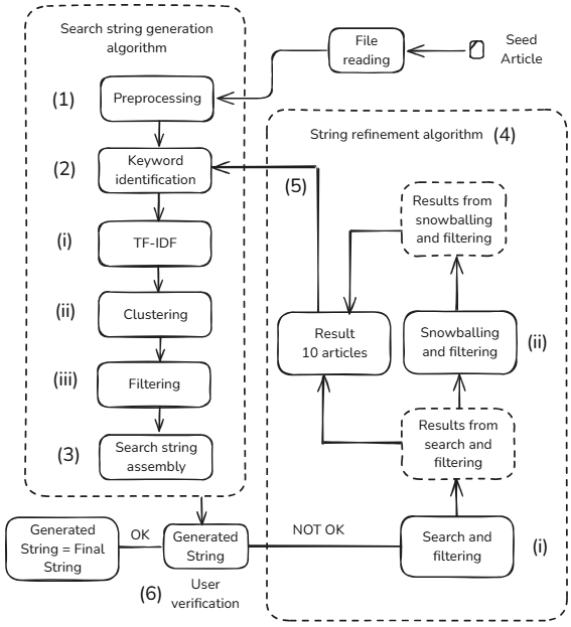


Figure 3: Overview of our proposed approach.

frequent bigrams and trigrams. The search string highlighting compound expressions such as *"machine learning"*, *"deep learning"*, and *"software product lines"* are more informative than isolated terms like *"machine"* or *"learning"* To limit string size without sacrificing context, we selected the 15 most frequent expressions. We tested lower and higher values; however, they had a significant reduction in the quality of the clusters generated in the next step.

(ii) Clustering The selected expressions were grouped into three clusters using the K-Means algorithm[12]. As we selected the top 15 most frequent terms, the choice of 3 clusters was made so that it would be possible to distribute the 15 words well among the clusters. Although we have tested the DB-Scan method [8], it showed disadvantages, such as the formation of too many clusters without clear semantic coherence.

(iii) Filtering To improve cluster quality, we applied a final filtering step based on semantic proximity of words, using the Nomic model [17]. Words with proximity scores below 0.5 were removed, ensuring that only the most relevant terms were retained. This threshold was chosen because, in our tests, words with a similarity above 0.5 already exhibited a meaningful relationship with others in the same cluster.

(3) Search string assembly. With the clusters defined, we constructed the search string connecting the terms with the appropriate logical operators (AND or OR). Terms within the same cluster were joined by OR, while different clusters were connected by AND.

(4) String refinement algorithm. To further enhance the search string, we developed a refinement step, divided into three substeps:

(i) Search and filtering. Using the generated string, we performed a scientific search using the Semantic Scholar API[3].

We applied a filtering metric that considers both total citations and "meaningful" citations, as provided by Semantic Scholar, which

[2]https://github.com/CeON/CERMINE

[3]https://api.semanticscholar.org/api-docs/

uses the meaningful citation metric to qualify the actual relevance of a citation within a scientific article. According to Valenzuela et al. [32], this metric is based on a supervised model that classifies citations as important, considering features such as the section of the article where the citation appears. The goal is to highlight contributions that substantially influence new work. Then, the five most relevant articles are selected. Although newer articles may have fewer citations, this bias was mitigated by using those meaningful citations instead of just the number of common citations. We also tested the use of conference impact factors, but this approach was discarded due to incomplete data availability in Semantic Scholar. A recent comparison of bibliographic tools [13] revealed complementary strengths between Semantic Scholar and OpenAlex, especially in metadata coverage and openness. As future work, we plan to combine both to improve data completeness and retrieval robustness.

(ii) Snowballing and filtering. With the five selected articles, we performed automated snowballing using the Semantic Scholar API. This search technique uses the references of each article (backward snowball) and cites articles (forward snowball), progressively expanding the set of relevant related articles. After we removed the duplicated articles, the result was a new list of articles different from the original five.

We applied the same filtering method to the new list, which yields the five most relevant articles from the snowballing application. Additionally, during the ranking process, we ensured diversity by selecting articles with varying publication dates. In the end, we obtained a total of ten articles (the initial five and the five from snowballing), a number aligned with the typical first-page return from a conventional search engine, such as Google Scholar. In our tests, selecting a large number of articles did not lead to better results and only increased the processing time of the filtering step. Therefore, only the top 10 articles were selected. It is important to note that, once an article is used in a refinement step, it is excluded from all subsequent iterations.

(5) Repeat of step 2 with 10 articles. Using the final list of ten relevant articles, we repeated the keyword identification step (Step 2), generating a new search string based on the titles and abstracts of these articles, retrieved directly from the Semantic Scholar API, without the need for further extraction through CERMINE. In this iterative step, the ten articles are always the newly returned results from the previous search, with no accumulation.

(6) User verification. The users may review and modify the newly generated string via an interface, adjusting clusters, removing irrelevant keywords, and adding terms to a negation cluster. This iterative process continues until the search string meets the user-defined criteria of quality and completeness, such as the saturation of new and relevant results across iterations. This human-in-the-loop approach ensures a balance between automation and control, making the method adaptable to both expert and non-expert users.

5 Conclusion and Future Plans

Based on the results obtained with the LLM-based approach, it becomes evident that there are still significant gaps in the use of LLMs to automate the search string process. Our goal was to take advantage of the insights gained from this study to propose an ML-based approach. As next steps, we intend to perform a comparative analysis of both approaches following the methodological framework detailed in Section 3.3. Moreover, we plan to expand the comparison between LLMs by increasing the number of SLRs used as benchmarks and incorporating more variations, such as additional models, different prompt strategies, temperature settings, and using more databases for searches, in addition to ACM and Scopus. Furthermore, we plan to conduct a qualitative evaluation of the user verification step of the quality and completeness of the generated ML-based search string.

First, we will perform a deeper quantitative analysis across multiple previously published SLRs, following the same methodology as in RQ_1 and RQ_2. We will investigate whether LLMs can match or outperform our proposed ML-based approach for generating search strings. The goal is to assess whether ML is a viable alternative to automate this critical step of the SLR process. Metrics such as precision, coverage, and F1-score will be computed at each iteration to assess how quickly and reliably users converge on high-quality queries. We will also request expert feedback to rate the relevance and completeness of retrieved results and to identify any critical omissions or false positives introduced during refinement.

Second, we will evaluate the iteration-wise quality of the refinement process itself by looking at the evolution of the keyword clusters, tracking how many new relevant terms are added, how many irrelevant terms are removed, and how the exclusion cluster affects final retrieval performance. This will reveal whether there are diminishing returns after a certain number of cycles or thresholds where user effort yields maximum benefit.

Finally, to understand usability for non-expert researchers, we will conduct qualitative user studies with participants from adjacent fields who lack deep domain knowledge. Through think-aloud protocols and post-task interviews, we will calculate how intuitively users interact with the clustering interface, how effectively they can identify and correct misclustered terms, and how confident they are with the final generated search string. Together, these evaluations will provide actionable insights into both the efficacy and human factors surrounding iterative search string refinement, guiding future enhancements to make the system robust, transparent and accessible to all levels of SLR practitioners.

ARTIFACT AVAILABILITY

The authors declare that the research artifacts supporting the findings of this study are accessible at https://doi.org/10.5281/zenodo.17055783. The corresponding GitHub repository (version v1.7.0) is also publicly available at https://github.com/aisepucrio/llm-search-string/tree/v1.7.0.

ACKNOWLEDGMENTS

This research was partially funded by the Brazilian funding agencies CAPES (Grant 88881.879016/2023-01), CNPq (Grant 302339/2022 – 1), FAPESP (Grant 2023/00811-0), and the Binational Cooperation Program CAPES/COFECUB (Ma1036/24). We also acknowledge the Brazilian companies Stone[4] and Flopo[5] for their financial support.

[4]https://www.stone.com.br/
[5]https://flopo.com.br/

REFERENCES

[1] Nauman bin Ali and Binish Tanveer. 2022. A comparison of citation sources for reference and citation-based search in systematic literature reviews. *e-Informatica Software Engineering Journal* 16, 1 (2022).

[2] Ahmad Alshami, Moustafa Elsayed, Eslam Ali, Abdelrahman E. E. Eltoukhy, and Tarek Zayed. 2023. Harnessing the power of ChatGPT for automating systematic review process: methodology, case Study, limitations, and future directions. *Systems* 11, 7 (2023), 1–7.

[3] Ahmed Al-Zubidy and Jeffrey C. Carver. 2019. Identification and prioritization of SLR search tool requirements: an SLR and a survey. *Empirical Software Engineering* 24, 1 (2019), 139–169.

[4] Marwa Assim, Qasem Obeidat, and Mustafa Hammad. 2020. Software Defects Prediction using Machine Learning Algorithms. In *2020 International Conference on Data Analytics for Business and Industry (ICDABI)*. IEEE, 1–6.

[5] Prashant Bansal. 2024. Prompt Engineering Importance and Applicability with Generative AI. *Journal of Computer and Communications* 12 (2024), 14–23.

[6] Jeffrey C. Carver, Edgar Hassler, Elis Hernandes, and Nicholas A. Kraft. 2013. Identifying barriers to the systematic literature review process. In *7th International Symposium on Empirical Software Engineering and Measurement (ESEM'13)*. IEEE, 203–213.

[7] Oscar Dieste and Anna Griman Padua. 2007. Developing search strategies for detecting relevant experiments for systematic reviews. In *1st Symposium on Empirical Software Engineering and Measurement (ESEM'07)*. ACM, 215–224.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 226–231.

[9] Luyi Feng, Yin Kia Chiam, Erma Rahayu Mohd Faiza, and Unaizah Obaidellah. 2017. Using suffix tree clustering method to support the planning phase of systematic literature review. *Malaysian journal of Computer Science* 4, 30 (2017), 311–332.

[10] Mohammad Ghafari, Mortaza Saleh, and Touraj Ebrahimi. 2012. A federated search approach to facilitate systematic literature review in software engineering. *International journal of Software Engineering & Applications* 2, 3 (2012), 1–13.

[11] Andreas Hinderks, Francisco José Domínguez Mayo, Jörg Thomaschewski, and María José Escalona. 2020. An SLR-tool: Search process in practice: A tool to conduct and manage systematic literature review (SLR). In *42nd Conference on Software Engineering: Companion Proceedings (ICSE-Companion - ICSE'20)*. IEEE Press, 81–84.

[12] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. 2023. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences* 622 (2023), 178–210. doi:10.1016/j.ins.2022.11.139

[13] Jailma Januário, Maria Isabel Nicolau, Katia Romero Felizardo, and Juliana Alves Pereira. 2025. Toward Reliable Forward Snowballing in Systematic Literature Reviews: A Comparative Study and Framework Proposal. In *Brazilian Symposium on Software Engineering, Insightful Ideas and Emerging Results Track (SBES IIER)*. SOL, 1–7.

[14] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Report. EBSE Technical Report, EBSE-200701, Keele University and University of Durham, Staffordshire, UK and Durham, UK. https://www.cs.auckland.ac.nz/~norsaremah/EBSE-2007-01.pdf

[15] Diogo Adário Marassi, Juliana Alves Pereira, and Katia Romero Felizardo. 2025. Comparing LLMs and Proposing an ML-Based Approach for Search String Generation in Systematic Literature Reviews. https://github.com/aisepucrio/llm-search-string. Accessed: 2025-07-15.

[16] Germano Duarte Mergel, Milene Selbach Silveira, and Tiago Silva da Silva. 2015. A method to support search string building in systematic literature reviews through visual text mining. In *30th Annual ACM Symposium on Applied Computing (SAC'15)*. ACM DL, 1594–1600.

[17] Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. *Nomic Embed: Training a reproducible long context text embedder*. Technical Report. Nomic AI.

[18] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. 2021. Learning software configuration spaces: A systematic literature review. *Journal of Systems and Software* 182 (2021), 111044.

[19] Jason Portenoy and Jevin D. West. 2020. Constructing and evaluating automated literature review systems. *Scientometrics* 125, 3 (2020), 3233–3251.

[20] Heri Ramampiaro, Daniela Cruzes, Reidar Conradi, and Manoel Mendonça. 2010. Supporting evidence-based Software Engineering with collaborative information retrieval. In *6th Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'10)*. IEEE Press, 1–5.

[21] Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. 133–142.

[22] Rasmus Ros, Elizabeth Bjarnason, and Per Runeson. 2017. A machine learning approach for semi-automated search and selection in literature studies. In *21st Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. ACM DL, 1–10.

[23] Rasmus Ros, Elizabeth Bjarnason, and Per Runeson. 2017. A machine learning approach for semi-automated search and selection in literature studies. In *21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. ACM DL, 118–127.

[24] Atrisha Sarkar, Jianmei Guo, Norbert Siegmund, Sven Apel, and Krzysztof Czarnecki. 2015. Cost-Efficient Sampling for Performance Prediction of Configurable Systems. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 342–352. doi:10.1109/ASE.2015.45

[25] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. 2012. SPL Conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal* 20, 3–4 (2012), 487–517. doi:10.1007/s11219-011-9152-9

[26] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, and Gunter Saake. 2010. Measuring Non-functional Properties in Software Product Lines for Product Derivation. In *Proceedings of the 14th International Software Product Line Conference (SPLC)*. IEEE.

[27] Paramvir Singh and Karanpreet Singh. 2017. Exploring automatic search in digital libraries: A caution guide for systematic reviewers. In *21st Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. ACM DL, 236–241.

[28] Francisco Carlos Monteiro Souza, Alinne Cristinne Corrêa dos Santos, Stevão Alves de Andrade, Rafael Serapilha Durelli, Vinícius Humberto Serapilha Durelli, and Rafael Alves Paes de Oliveira. 2017. Automating search strings for secondary studies. In *Information Technology – New Generations*, Shahram Latifi (Ed.). Advances in Intelligent Systems and Computing, Vol. 558. Springer, Cham, 839–848.

[29] Mariusz Sośnicki and Leszek Madeyski. 2021. ASH: A new tool for automated and full-text search in systematic literature reviews. In *21st Conference Computational Science (ICCS'21)*. Springer, 362—-369.

[30] Yueming Sun, Ye Yang, He Zhang, Wen Zhang, and Qing Wang. 2012. Towards evidence-based ontology for supporting Systematic Literature Review. In *16th Conference on Evaluation Assessment in Software Engineering (EASE'12)*. IEEE Press, 171–175.

[31] Paul Temple, José Angel Galindo, Mathieu Acher, and Jean-Marc Jézéquel. 2016. Using Machine Learning to Infer Constraints for Product Lines. In *Proceedings of the 20th International Software Product Line Conference (SPLC)*. 209–218. doi:10.1145/2934466.2934472

[32] Marco Valenzuela, Vu Ha, and Oren Etzioni. 2015. Identifying Meaningful Citations. In *Scholarly Big Data: AI Perspectives, Challenges, and Ideas: Papers from the 2015 AAAI Workshop*. Association for the Advancement of Artificial Intelligence, Austin, Texas, USA.

[33] Raymon van Dinter, Bedir Tekinerdogan, and Cagatay Catal. 2021. Automation of systematic literature reviews: A systematic literature review. *Information and Software Technology* 136, C (2021), 16 pages.

[34] Shuai Wang, Harrisen Scells, Bevan Koopman, and Guido Zuccon. 2023. Can ChatGPT write a good boolean query for systematic review literature search?. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'23)*. ACM, 1426—-1436.

[35] Claes Wohlin. 2014. Writing for synthesis of evidence in empirical software engineering. In *8th International Symposium on Empirical Software Engineering and Measurement (ESEM'14)*. ACM, 1–4.