Adaptive Search Query Generation and Refinement in Systematic Literature Review

Maisie Badami[a,*], Boualem Benatallah[b,a], Marcos Baez[c]

[a] *University of New South Wales (UNSW) Australia*
[b] *Dublin City University Ireland*
[c] *Bielefeld University of Applied Sciences Germany*

Abstract

Systematic literature reviews (SLRs) are a central part of evidence-based research, which involves collecting and integrating empirical evidence on specific research questions. A key step in this process is building Boolean search queries, which are at the core of information retrieval systems that support literature search. This involves turning general research aims into specific search terms that can be combined into complex Boolean expressions. Researchers must build and refine search queries to ensure they have sufficient coverage and properly represent the literature. In this paper, we propose an adaptive query generation and refinement pipeline for SLR search that uses reinforcement learning to learn the optimal modifications to a query based on feedback from researchers about its performance. Empirical evaluations with 10 SLR datasets showed our approach achieves comparable performance to queries manually composed by SLR authors. We also investigate the impact of design decisions on the performance of the query generation and refinement pipeline. Specifically, we study the effects of the type of input seed, the use of general versus domain-specific word embedding models, the sampling strategy for relevance feedback, and number of iterations in the refinement process. Our results provide insights into the effects of these choices on the pipeline's performance.

Keywords:
Systematic Reviews, Query Enrichment, Query Adaptation, Reinforcement Learning, Word Embedding

*Corresponding author
Email address: `m.badami@unswalumni.com` (Maisie Badami)

1. Introduction

Systematic literature reviews (SLRs) offer robust and transferable evidence for evaluating and interpreting relevant research on a particular topic [1]. They establish the groundwork for future research by enabling researchers to systematically identify, evaluate, and synthesize all relevant research evidence on a particular topic, in order to answer research questions in a transparent and systematic manner [2]. Given their demonstrated value, SLRs are becoming a popular type of publication in empirical research domains such as evidence-based software engineering [1]. Identifying relevant studies is a crucial step in the SLR process, as it can impact the overall *quality* and *workload* of the review. This step is guided by the definition of research questions that set the scope for the entire SLR [1, 2]. Researchers capture this scope by creating Boolean *search queries* for scientific digital libraries [1].

The appropriateness of the search query to capture relevant studies is crucial for ensuring that the SLR accurately synthesizes the literature downstream of the process [3]. It also greatly impact the workload as researchers need to screen the volume of returned results to filter out irrelevant work. Consequently, a significant portion of the SLR effort goes into identifying relevant studies [4]. In addition, SLR authors might have limited knowledge of the review topic and the search terms at the beginning of the process, and primary studies themselves may use different terminology to refer to similar concepts [5]. These factors make building proper SLR search queries a challenging task and a potential point of failure, which has led to research on automating query generation for SLR search. Existing solutions have focused on automatically building search queries (e.g., [6, 7]) or automatically refining existing search queries (e.g., [3, 8]), but these methods are often limited to specific research domains (e.g., medical domain). This limitation imposes challenges for generalizing and adopting these methods in different SLRs and research domains. In addition, these methods rely on machine learning algorithms (e.g., classifiers) that require domain-specific training data. Furthermore, these methods require authors to compose and provide an initial query, which poses challenges for researchers who have limited knowledge about building search queries, especially those who are new to a research topic [5]. Therefore, there is a need for solutions that can adapt better to different domains and SLRs while supporting authors in the early stage of the review process when knowledge is still limited.

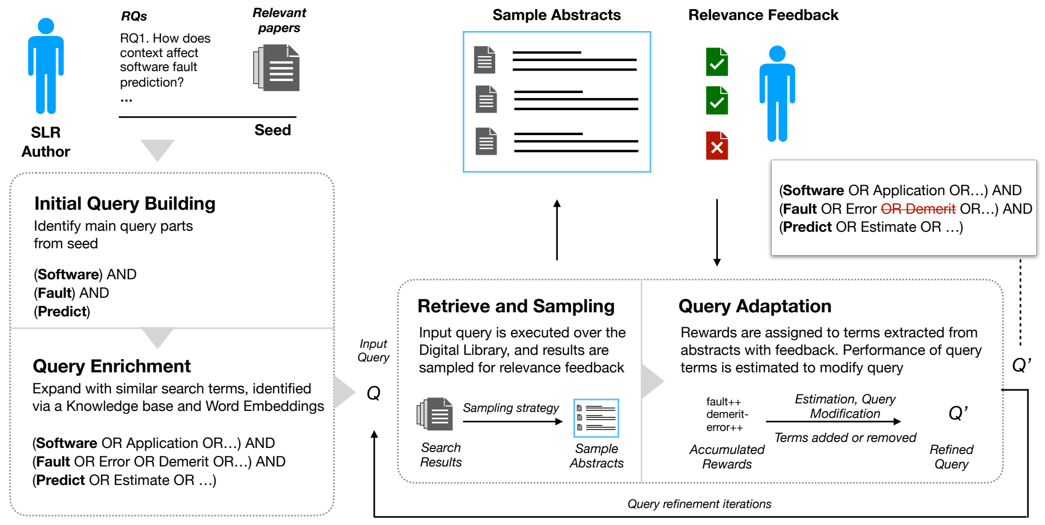

RQs
RQ1. How does context affect software fault prediction?
...
Relevant papers
Seed
SLR Author
Initial Query Building
Identify main query parts from seed
(Software) AND
(Fault) AND
(Predict)
Query Enrichment
Expand with similar search terms, identified via a Knowledge base and Word Embeddings
(Software OR Application OR…) AND
(Fault OR Error OR Demerit OR…) AND
(Predict OR Estimate OR …)
Input Query
Q
Sample Abstracts
Relevance Feedback
(Software OR Application OR…) AND
(Fault OR Error ~~OR Demerit~~ OR…) AND
(Predict OR Estimate OR …)
Retrieve and Sampling
Input query is executed over the Digital Library, and results are sampled for relevance feedback
Sampling strategy
Search Results
Sample Abstracts
Query Adaptation
Rewards are assigned to terms extracted from abstracts with feedback. Performance of query terms is estimated to modify query
fault++
demerit-
error++
Accumulated Rewards
Estimation, Query Modification
Terms added or removed
Q'
Refined Query
Q'
Query refinement iterations

Figure 1: High-level overview of the query building and refinement pipeline. The pipeline takes a *seed* as input, representing the SLR scope. The initial step builds a query based on the seed, enriched with similar terms. The second step iteratively refines the query using author feedback on search result samples.

We addressed gaps in effectively constructing SLR search queries by devising an adaptive query *building* and *refining* pipeline that relies on a reinforcement learning approach to incrementally refine a generated search query based on authors' feedback (see Figure 1). More precisely, given a seed expressing the scope of the literature review (e.g., research questions or a set of relevant abstracts), the pipeline automatically generates a search query from the initial seed. Through an interactive process, the pipeline then leverages author feedback on the query search results to incrementally improve the generated query. The aim is to maximize recall while minimizing the workload in later screening steps. The rationale behind our approach is to leverage SLR authors' knowledge about the scope of the review and relevance of selected studies to refine SRL search queries. The contributions of this paper are as follows:

- We propose a data-driven pipeline that exploits a natural language description of the SLR scope to generate an initial search query and enrich it with semantically related and diverse terms;

- We devise an incremental and adaptive process to refine search queries for SLRs. The proposed reinforcement learning approach learns to modify and adjust the search queries by observing the relevance feedback provided by researchers on query search results;

- We empirically show, in an evaluation with 10 SLR datasets, that the proposed pipeline can generate effective search queries (in terms of recall and workload) that have a performance comparable to the queries that domain experts manually compose;

- We devise experiments and provide evidence for the impact of meaningful design decisions in the query generation and refinement process. This includes the impact of the type of input seed and the use of general and domain-specific word embedding models in the initial query generation step. We also assess the impact of sampling strategies adopted for relevance feedback and the number of iterations in the refinement process.

The present paper is an extended journal version of our previous research paper [9], which expands on our previous work in the following ways: (i) We provide a more comprehensive review of the literature, which includes a background section with an overview of the SLR query generation process and related concepts. We expanded the related work with a more detailed review of the research on SLR query building support and introduced a new discussion on Human-in-the-Loop approaches that are meaningful to position the design decisions in the pipeline. (ii) We present a more detailed description of our approach, including the description of adopted techniques, algorithm definitions, and more examples to accompany the SLR query generation process. (iii) We performed a new experimental evaluation that assesses the impact of general versus domain-specific word embedding models in the query building process. This is an important dimension that contributes with new insights into the generalization of our approach and findings and sheds new light into the implications for pipeline design.

The rest of this paper proceeds as follows. Related work is given in Section 2. Section 3 presents our proposed approach. The experiments and evaluation are presented in Section 4. And finally, Section 7 concludes the paper.

2. Background and Related Work

This section reviews the existing literature on SLR search query building support. We provide a general background on the query definition process for SLRs and then address two specific tasks that are central to ongoing research in SLR query building support: i) automatic techniques for *generating* search queries, and ii) automatic *refinement* of SLR search queries. Additionally, we discuss Human-in-the-loop techniques in SLR automation, which inform the approach adopted in our proposed pipeline.

2.1. Background

Researchers use available information about the scope of the review to build SLR search queries. This information can be found in the review protocol or a collection of known relevant studies (i.e., quasi gold standard (QGS) [10]) [11, 12]. They extract high-level concepts that describe the topic and scope of the review from available knowledge, such as review questions. One example of a technique used in medical domain and software engineering [1] is PICO, which stands for Population (or Problem), Intervention, Comparison, and Outcomes. Each component of PICO defines specific aspects of the research questions, which can be used to define search queries and data extraction coding schemes. Researchers use these queries to search for relevant studies on SLR information retrieval (IR) systems, such as digital libraries, which are built upon standard Boolean retrieval models [13, 14]. As shown in Figure 2, researchers refine initial queries by adding more synonyms or relevant terms as they inspect search results and identify relevant studies. The goal of the query building process is to produce a search query that can maximize recall while minimizing the workload in later screening steps.

While most of the literature on SLR automation focuses on study selection (see [15] for a review), interest in SLR search query building support has recently sparked. These efforts can be categorised into two main groups: i) automatic techniques for *generation* search queries, and ii) automatic *refinement* of SLR search queries. Next, we discuss these categories.

2.2. SLR Search Queries Generation

Several studies explored approaches to support researchers in building search queries for SLRs. Existing approaches use information from the review protocol (e.g., review questions, inclusion and exclusion criteria) or a set of relevant abstracts to extract relevant terms that can help with building

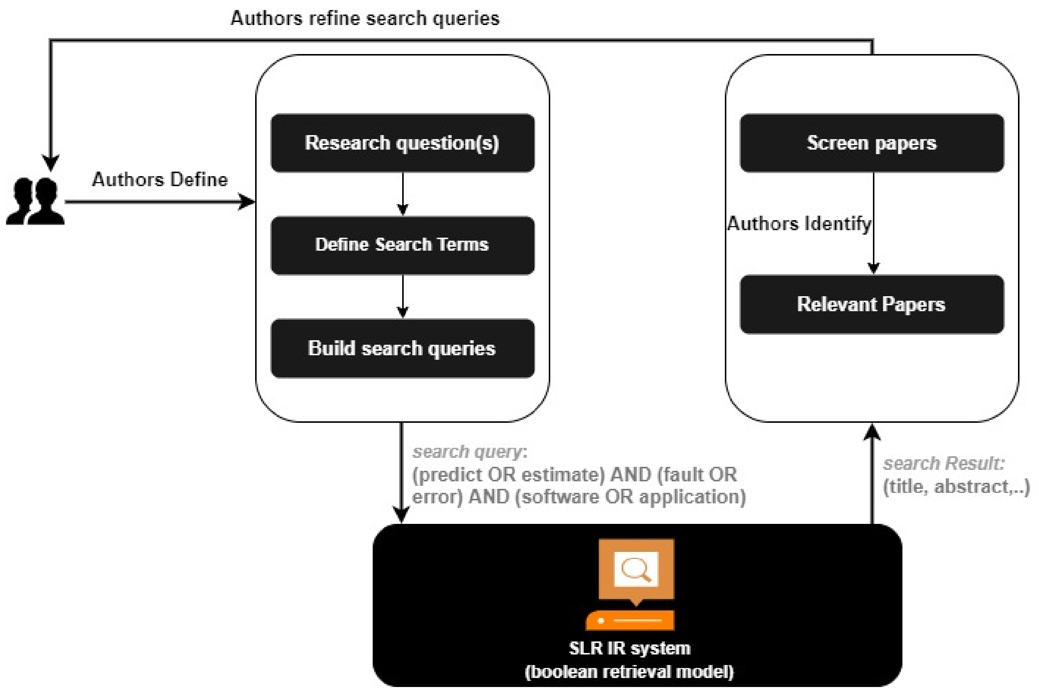


Figure 2: The search query building and refining process, currently a manual SLR task.

search queries [7, 16, 17, 6]. These techniques mostly rely on text analysis and mining techniques (e.g., terms TF-IDF) to find the most relevant terms from given corpus [6, 7]. Some techniques leverage visualization to suggest terms that researchers can use to manually refine search queries [7].

Recently, tools have also been developed based on the above techniques [18]. For instance, 2dSearch[1] [18] provides a method to build search queries leveraging visualisation techniques. In 2dSearch the search query is represented in a visual query canvas, where search terms are objects and the relationship between them is expressed in terms of nesting or co-occurrence. In enabling researchers to build queries using this visual paradigm, the aim is to guide them towards the correct definition of queries, provide them a clear semantic representation of search expressions, and enable the reuse query definitions over different scientific databases. While formulating and reusing queries is simpler in this approach, a researcher new to the topic (and to the process)

[1]https://app.2dsearch.com/

is still required to formulate the query, with the challenges that this poses.

Another tool is QueryVis [19] which focuses on characterizing contributions of query clauses to the accuracy of a search query. It annotates the query with the number of relevant studies retrieved with each query clause.

As described above, these approaches and tools focus mainly on suggesting terms to help researchers build queries and do not provide an end-to-end solution for query adaptation and refinement. Yet, the techniques serve as an inspiration for building the query generation component.

2.3. SLR Search Queries Refinement

Existing automatic query refinement techniques comprise two main steps: i) generating alternative queries from an initial query, and ii) predicting the performance of generated queries and selecting queries with the best performance. To generate alternative queries from an initial query, these techniques leverage query refinement techniques such as query expansion, query reduction and query rewriting [3, 20, 8, 21].

To predict the performance of generated queries, they leverage machine learning algorithms (e.g., classification models) that learn to rank generated queries based on their performance [3, 20, 8]. These studies used different query performance predictors as features to learn machine learning models and measure the performance of generated queries [8, 22, 3].

Kim et al. [8] proposed an approach for building alternative queries from an initial query. Their system contains two main tasks: generating boolean queries and ranking generated queries. They used information from pseudo-labeled citations (i.e., the top-k documents retrieved by a baseline system) to build a decision tree that can decide whether a document is relevant or not. Each path from the root to the node in the tree defines an alternative query. Afterward, each path from the root to a positive leaf node (indicating possible relevance) formulates a boolean query. For learning a model to rank and suggest the best query from the set of alternative queries, they utilized features from pre-retrieval QPPs (e.g., the similarity of the generated query to the original query) and post-retrieval QPPs (e.g., the query result set including the items in the baseline query retrieval set). The authors compared this query expansion technique to the contemporary query expansion techniques and experimentally showed the formers' superiority.

For instance, Scells et al. [3] proposed a query refinement technique that relies on query expansion and reduction. They focus on building alternative queries from a query composed by expert users. The alternative queries are automatically ranked based on the predicted performance, without user intervention, in a one-step refinement process. They used heuristics (e.g., relevance of a new citation to a query) to train their query ranking models. They found that their proposed approach could generate boolean queries with better recall and precision than previous baseline queries. They also demonstrated that the nearest neighbour model outperformed other models in ranking queries with better precision than the original queries.

While valuable, existing proposed solutions for query building and refinement in SLRs, they have important limitations: i) adopting these techniques for different research domains and SLRs is complex and time-consuming, as they often rely on machine learning techniques that require domain-specific training data; and ii) they require authors to provide an initial query, which poses challenges to researchers who have less knowledge about building search queries and especially who are new to the research topic [23, 24].

2.4. Human-in-the-loop Automation Techniques for SLRs

Our systematic literature review results in [25] revealed that developing automation techniques for SLRs face important obstacles. One significant obstacle is the absence of sufficient training data for learning machine-based algorithms at the beginning for different SLRs or different research domains. To overcome this limitation, some existing approaches leverage human-machine collaboration techniques (e.g., active learning techniques [26, 27, 4, 28, 29, 30]). These techniques attempt to continuously learn to classify or rank citations using feedback from oracles (experts or crowds). The active learning approach has shown significant success in reducing citation screening workload and cost [29, 31, 32]. Due to this success, some tools (e.g., Rayyan[2]) have also been implemented to support researchers in citation screening.

Outside SLR automation techniques, reinforcement learning is widely used in automating complex tasks [33, 34, 35, 36]. Multi-armed bandit models have shown success in various domains such as content recommendation [33], gaming [34], online learning [35] and data curation [36]. For instance, using a bayesian multi-armed-bandit, Tabebordbar et al. [36] proposed an approach that dynamically adjusts rules that annotate data in social media such as Twitter and Facebook. Williams et al. [35] leveraged a combination of crowdsourcing and bayesian multi-armed-bandit algorithm to generate expla-

[2]https://www.rayyan.ai/

nations to present to learners in problem solving settings (e.g., math problem solving).

Building upon the finding of empirical studies (e.g., [37]) on SLR automation, We build up previous research in automatic query expansion [38, 39] and reinforcement learning [36, 35] to propose a novel pipeline for building and refining SLR search queries. This pipeline learns to refine automatically generated queries using researchers feedback about the query retrieved results. The pipeline is usable in different domains and SLRs. Moreover, we devised novel search term enrichment algorithm leveraging both knowledge-based and embedding-based techniques. This alleviates an important challenge reported by survey participants in an empirical study on SLR automation [25], regarding the difficulties they face in building search queries. These difficulties stem mostly from inconsistent and often overlapping terminologies used by different research communities to describe concepts.

The authors of a recent study[40] examined the effectiveness of reinforcement learning models, using ChatGPT, in creating Boolean queries for systematic review literature searches. Their findings revealed that ChatGPT can generate queries that result in high search precision but at the cost of the recall. Ultimately, this study highlights ChatGPT's potential to generate efficient Boolean queries for systematic review literature searches. As ChatGPT can produce queries with high precision, it is a valuable resource for researchers performing systematic reviews, particularly for time-sensitive rapid reviews where sacrificing recall for increased precision is acceptable.

3. Incremental Query Building And Refining Pipeline

In our proposed query generation and refinement approach, the aim is to utilize the minimal information available to the researchers regarding the scope of a review to build initial search queries. Subsequently, through a refinement process based on researcher feedback, the initially generated queries are incrementally refined and improved. In our approach, the performance of a search query depends on the effectiveness of a query to i) retrieve relevant literature as defined by the scope of the review (recall), and ii) minimise the (unnecessary) screening effort incurred by the number of studies in the retrieved results. It is worth noting that striking this balance is important as very "open" search queries may be effective in retrieving the majority of relevant studies but may also yield a large number of search results. Conversely, narrow search queries may be more manageable but risk missing relevant studies.

To accomplish this goal, we devised the query building and refinement pipeline as illustrated in Figure 3. In summary, the pipeline receives a seed representing the scope of the SLR (e.g., research questions). Then, the *Initial Query Builder* component utilizes the seed to extract candidate terms for constructing an initial query. The *Query Enrichment* component expands the initial query by enriching the terms using knowledge and embedding-based techniques. The generated query is subsequently automatically executed on a digital library (DL) search engine to retrieve the search results. The *Query Adaptation* component then employs relevance feedback from researchers on the search results to assess the performance of the executed query. Finally, this component uses these observations to refine the query employing a reinforcement learning approach. In the following sections, we elaborate on each component of our proposed pipeline.

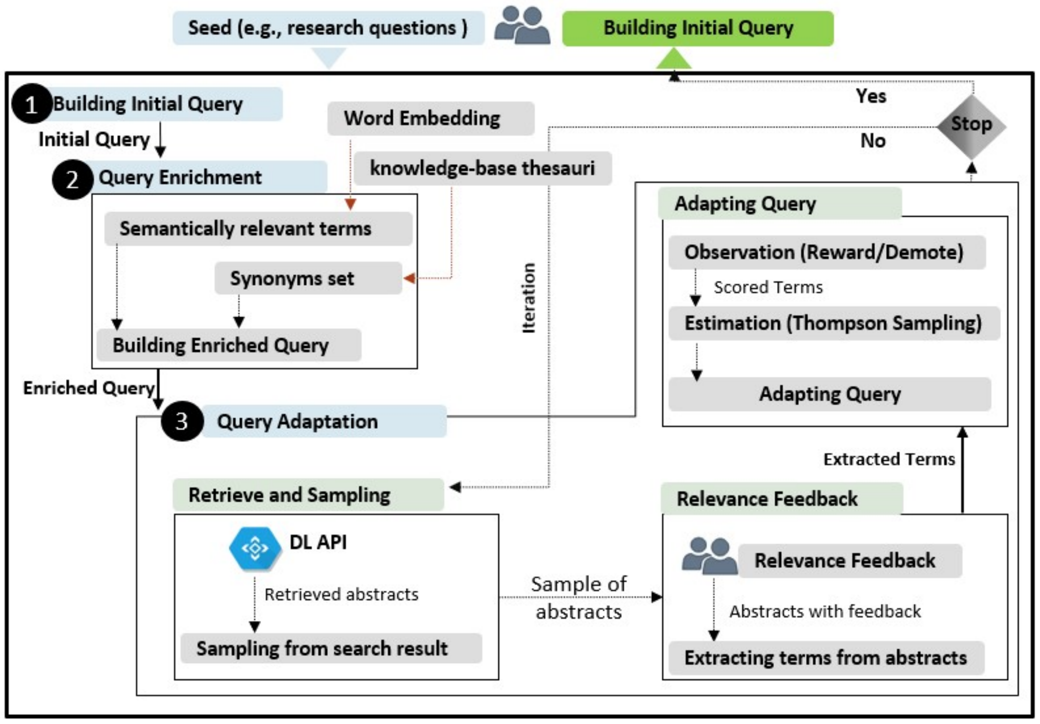


Figure 3: Architecture of the Query Building and Refinement Pipeline

3.1. Initial Query Builder

The first component of the pipeline is the *Initial Query Builder* which utilizes a high-level expression of the scope of an SLR to construct an initial query. The input to this component is a *seed*, which can be partially defined research question(s) or multiple relevant abstracts. This component first removes non-contributing terms (e.g., stop words and special characters). Next, it extracts all the terms (nouns, verbs) from the given seed using Stanford's CoreNLP library [41]. This component relies on TF-IDF to select terms for constructing an initial query when the seed contains more than one document. When the input seed contains only one document (e.g., one relevant abstract), the term frequency (TF) is used to select top-n relevant terms. The selected terms at this stage represent concepts that should be present in relevant studies (i.e., matching results). We refer to these terms as *main terms* of the query. Therefore, *Initial Query Builder* constructs the initial query by joining the main terms using the *'AND'* operator. For instance, for given research question *RQ:"Which techniques perform best when used to predict software fault?"* as the seed, the *Initial Query Builder* extracts the main terms and generates an initial query denoting: $q =$ (software AND fault AND predict).

The generated initial query search results will be narrow and not suitable for recall-oriented SLR search. The reason is that authors of scientific literature use different terminologies to express the same concept [42]. To address this, we devised the pipeline with a query enrichment component which we explain next.

3.2. Query Enrichment

We devised the pipeline with two query enrichment techniques: i) a knowledge-based approach for finding the synonyms of main query terms [43], and ii) an embedding approach to find alternative terms that are relevant to the query terms but may not be synonyms to the main terms [43]. Algorithm 1 shows the sketches of these two query enrichment techniques.

Knowledge-base Enrichment. This component identifies all the synonyms of a given query term using WordNet [44] and constructs a *synonym set* for each query term, containing the term and its synonyms. WordNet contains English words into groups of synonyms, called *synsets* [44]. One word in WordNet may have more than one synset. WordNet records the semantic relations between the synsets that describe the specific concept (hyponym) or generalized concept (hypernym) of a synset [44].

Algorithm 1: Search Query Enrichment

```
QueryEnrichment
    inputs : Initial Boolean Query ($q$);
    output: Enriched Boolean Query ($q^*$);
    // Define variables
    wordnet_synonyms: Synonyms from WordNet;
    word_embeddings: Word embedding terms set;
    $q = t_1 \wedge t_2 \wedge ... \wedge t_n$;
    $w \in word\_embeddings$ ;
    $T \leftarrow \{t_1, t_2, ..., t_n\}$;
    // Iterate through main terms
    foreach $t \in T$ do
        // Find synonyms from WordNet
        $set_s.Add(s_t)$ where $s_t \in wordnet\_synonyms$;
        // Find relevant terms from word embeddings
        foreach $w \in word\_embeddings$ do
            if $cos(\vec{set_s}, \vec{w}) > \alpha$ then
                $candidate\_set_t.Add(w)$
        // Select top-n enriched terms
        $enriched\_set \leftarrow top_n\ w \in candidate\_set_t$ in decreasing order
          by $cos(\vec{q}, \vec{w})$;
    // Construct enriched query
    foreach $t \in T$ do
        foreach $w \in enriched\_set$ do
            $Component\_t \leftarrow \vee w$
        $q^* \leftarrow \wedge Component\_t$;
    // Return enriched query
    return $q^*$;
```

To enrich a search term, this component selects a synset from WordNet that has a hyponym similar to other terms in the query. For instance, in the initial query q (from our ongoing example), the term "fault" has a synset representing the concept of "geography" and another representing the concept of "programming". The synset representing "programming" is selected because it has a similar hyponym to the term "software" in the query. As a result, for each term in query q, the following synset is selected for fault:{fault, defect, error,...}.

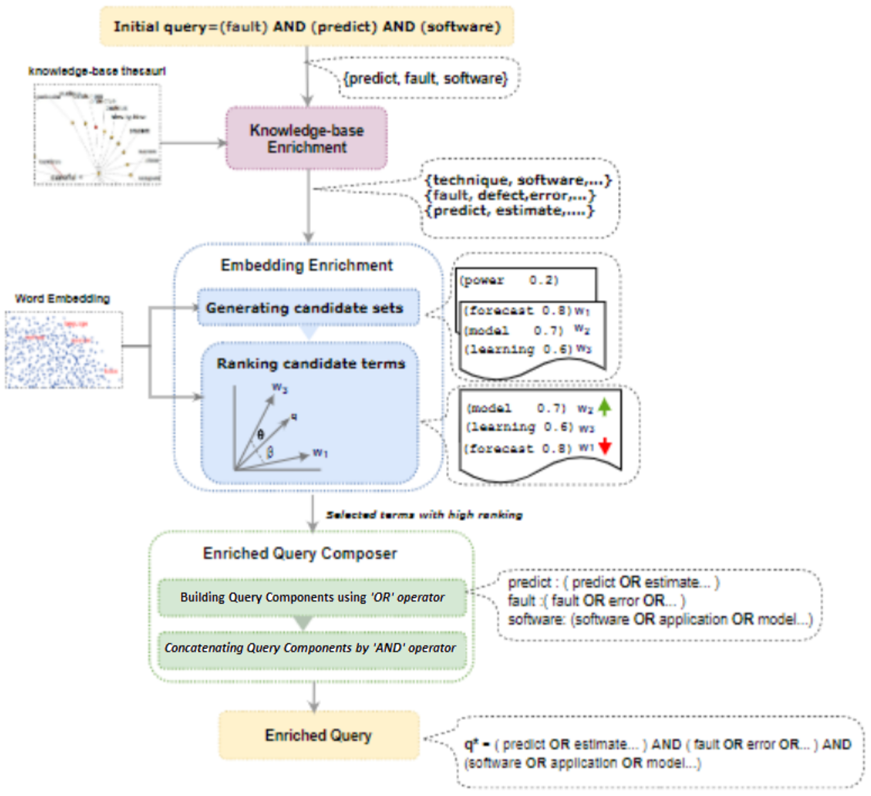

Initial query=(fault) AND (predict) AND (software)
{predict, fault, software}
knowledge-base thesauri
Knowledge-base Enrichment
{technique, software,...}
{fault, defect,error,...}
{predict, estimate,....}
Embedding Enrichment
Word Embedding
Generating candidate sets
(power 0.2)
(forecast 0.8) w1
(model 0.7) w2
(learning 0.6) w3
Ranking candidate terms
(model 0.7) w2
(learning 0.6) w3
(forecast 0.8) w1
Selected terms with high ranking
Enriched Query Composer
Building Query Components using 'OR' operator
Concatenating Query Components by 'AND' operator
predict : (predict OR estimate...)
fault :(fault OR error OR...)
software: (software OR application OR model...)
Enriched Query
q* = (predict OR estimate...) AND (fault OR error OR...) AND (software OR application OR model...)

Figure 4: Query enrichment process

This enrichment approach improves the query's retrieval performance [43]. However, the knowledge-based thesauri often do not contain all the semantically relevant terms that are not synonyms [45]. Therefore, to further improve the query performance in retrieving relevant studies, we introduced an additional enrichment technique. This technique enhances the query performance in retrieving studies that use alternative terms to express similar concepts [46]. Figure 4 illustrates how related terms are chosen and selected by this component.

Embedding-based Enrichment. This enrichment component builds upon a word embedding approach [45]. It uses a word embeddings model to identify the most relevant terms for each *synonym set* and collects these terms into an *enriched set*. First, it calculates the mean vector of each synonym set using the vectors of all terms within the synonym set ($\vec{set}_s = 1/|set_s| \sum_{s \in set_s} \vec{s}$), where set_s denotes a synonym set. Next, the component uses the cosine similarity score between embedding terms and the mean vector of each synonym set ($\vec{set}_s$) to find the top-n candidate terms in the embedding that have similarity to the synonym set. These top-n selected candidate terms form a candidate set (W) for each synonym set. Finally, to select the most similar terms from the candidate set, the *Embedding Enrichment* ranks the terms in the candidate set (W) based on their cosine similarity to the mean vector of the *initial query*. We chose to rank the terms in the candidate sets (W) based on their similarities to the query rather than their synonym sets (set_s).In doing so, we ensured that terms more relevant to the query are ranked higher and selected for enrichment [39]. A query's mean vector is calculated by averaging the vectors of all terms in the query ($\vec{q} = 1/|q| \sum_{t \in q} \vec{q}$), where q denotes a query. Therefore, the score of the term w from the candidate sets (W) is calculated as $score(w, q) = cos(\vec{q}, \vec{w})$. $\vec{w}$ denotes vector of a term w in a candidate set (W). In our experiment, a minimum similarity threshold (α) is used to select the top-n terms. These top-n terms form an *enriched set* for each corresponding synonym set.

Query Composer. Once all the *enriched sets* are generated, the *Query Composer* component constructs an enriched query using Boolean logic operators. The terms in the enriched sets serve as alternatives to the *main terms* in the initial query. Therefore, the *Query Composer* assembles the alternative terms in the enriched set using *OR* operator, forming a *query Component* for each *enriched set*. For instance, as shown in Figure 5, the main term *predict* and its enriched set {predict, estimate, model} result in a Boolean expression: (predict OR detect OR model). The final query is then formed by concatenating all the components, using the *'AND'* operator. For the initial query q, the enriched query q^* is generated as: q^* = (software OR program...) AND (fault OR defect OR...) AND (predict OR detect...). At this stage, the queries are built using only OR and AND operations. Incorporating other operations (e.g., NOT) is left to future extensions.

3.3. Query Adaptation

Query adaptation is a common practice in SLR searches. Researchers modify search queries based on the knowledge gained by screening more

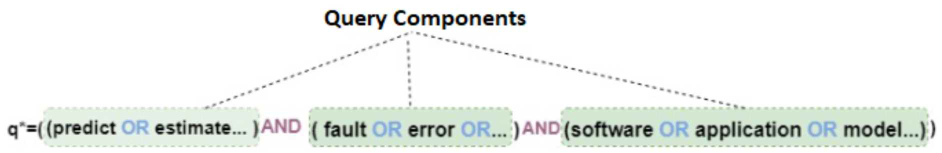

Query Components
q*=((predict OR estimate...) AND (fault OR error OR...) AND (software OR application OR model...))

Figure 5: The enriched query is formed by concatenating the query parts using *AND* operator

candidate papers over time. If a new semantically similar term is found in a newly discovered relevant paper, they modify the search query by adding the new term to the query or replacing the former insufficient terms. In practice, this process can require many iterations and is prone to errors. Moreover, due to the large and ever-expanding number of publications, researchers may spend significant time evaluating and appraising search results and adapting the queries accordingly [14].

To assist researchers in this time-consuming task, we formulated the query adaptation problem in the form of a reinforcement learning model, which learns to adapt queries by observing changes in the retrieved studies over time. Our approach relies on a multi-armed algorithm [47, 36]. In a multi-armed bandit, the algorithm continually chooses which action to take and each action results in a reward based on an underlying probability distribution [48, 33]. The algorithm learns to take the actions that maximise the accumulative rewards by repeating the action and observing the results [48, 35].

In the following, we provide a detailed description of the components of our proposed query adaptation approach.

Observation. The first step in our proposed query adaptation approach is *observation*, which initially executes the enriched query and retrieves the search results. It then gathers relevance feedback from researchers on a sample of search results to later inform the estimation component about its observations. This step consists of three components: i) **search and retrieve**; ii) **sampling**; and iii) **relevance feedback**.

3.3.1. Search and Retrieve.

This component provides an interface that facilitates searching across various digital libraries. It contains adapters for executing search queries by calling digital library APIs.[3] *Search and Retrieve* executes the queries and retrieves the results for further processing.

3.3.2. Sampling

In the systematic review, researchers examine the query performance by screening a subset of retrieved papers. Moreover, they use a set of relevant studies that are already known (QGS) and check whether the search could retrieve the QGS studies [49]. We aim to improve the retrieval performance of the search query by learning about the relevance of a sample set from the query search results. For this, we introduce a sampling mechanism to the pipeline. The *Sampling* component is used to select S items from the search results based on a learning strategy introduced next.

To select samples from the retrieved result, this component first extracts all the terms (nouns, verbs) from the retrieved abstracts using Stanford's CoreNLP library [41]. Next, it generates the corresponding word vectors for these terms using an embedding model (e.g., Glove [50]). *Sampling* then averages all these vectors to generate the mean vector of each abstract. The *Sampling* component also calculates the mean vector of the query by averaging corresponding vectors of all its terms. It uses the cosine similarity metric to calculate the similarity between the vector of each abstract and the vector of the query: $Score(a, q*) = Sim_a = cos(\vec{q*}, \vec{a})$ (see Figure 6). Here, $\vec{a}$ represents the vector of the abstract, and $\vec{q*}$ is the vector of the executed query. The *sampling* then ranks the abstracts in descending order based on their similarity score.

Finally, *Sampling* selects a set of samples based on a sampling strategy. The sampling strategy is applied either by seeking feedback on *uncertain* or *certain* abstracts. To this end, a lower η and a higher ζ similarity score thresholds define the sampling strategy. In sampling based on uncertainty, abstracts are those that the pipeline is less confident about their relevance. In other words, those that fall between the similarity score thresholds ($\eta < Sim_a < \zeta$) are selected for receiving feedback. In contrast, sampling based on certainty seeks confirmation on abstracts with the highest similarity scores ($Sim_a > \zeta$). The choice for the sampling method is a configuration parameter of the pipeline.

[3]Application Programming Interface

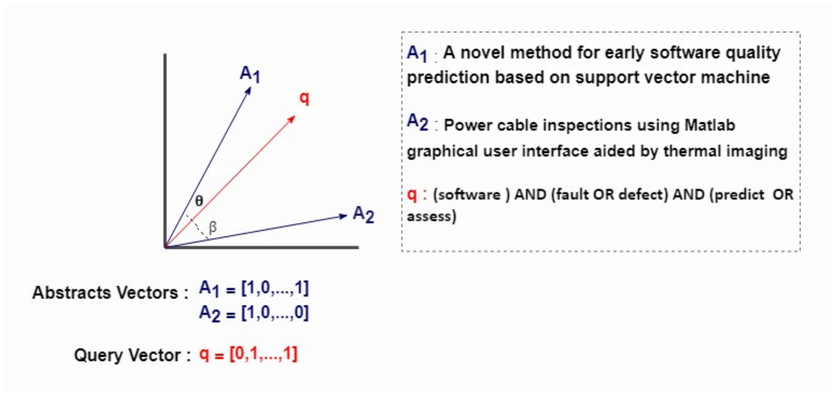

A_1
q
θ
β
A_2
A_1 : A novel method for early software quality prediction based on support vector machine
A_2 : Power cable inspections using Matlab graphical user interface aided by thermal imaging
q : (software) AND (fault OR defect) AND (predict OR assess)
Abstracts Vectors : A_1 = [1,0,...,1]
A_2 = [1,0,...,0]
Query Vector : q = [0,1,...,1]

Figure 6: On how the relevance of an abstract is computed as the cosine similarity between the vectors of abstracts and the query. In this figure, the distance between A_1 and q (θ) is smaller than A_2 and q distance (β). A_2 could be considered as an irrelevant abstract to the query as the distance is large.

3.3.3. Relevance Feedback

The *Relevance Feedback* component presents the selected samples to researchers for receiving feedback. Leveraging a binary relevance feedback method [51, 52], researchers can screen the sample abstracts and label them as *relevant* or *irrelevant*, based on whether they fit the scope of the review or not. Researchers can also add samples of QGS relevant abstracts that were not present in the retrieved result (or in the sample set). They can also screen additional abstracts from the search results. The abstracts with feedback are then used in the next component of the pipeline to modify the search query in order to improve its retrieval performance.

Estimation. The next step in our proposed query adaptation is *estimation*, which computes the retrieval performance of the terms within the query. A term's performance is estimated based on its accumulated rewards and demotes over time. The query adaptation process is thus iterative, i.e., a term that is deemed as uncertain based on its current performance, might still be incorporated into the query in future iterations - depending on changes in its estimated performance. In our query adaptation approach, which we formulated as a multi-armed bandit problem, each term in a query represents an arm of the bandit. Therefore, the Bayesian algorithm will afford more confidence for either keeping or replacing a term. Next, we elaborate on the steps of estimation: i) reward/demote schema and ii) terms performance estimator.

Reward/Demote Scheme. The *Reward/Demote Scheme* component keeps records of the accumulated rewards and demotes for each query term. It also finds candidate terms in the relevant abstracts that are not yet part of the query but have shown positive performance (candidate terms). This component makes use of the following sub-components:

- **Reward/Demote Calculator.** Each time new relevant/irrelevant abstracts are received, the *Rewards/Demotes calculator* extracts all terms (nouns, verbs) from each abstract, using Stanford's CoreNLP [41]. It then updates the rewards and demotes of the query terms, based on their presence in relevant and irrelevant abstracts. The query terms have a default value of 1 for reward and demote in the first iteration. Each time a query term appears in a new irrelevant abstract, it is demoted ($d_t = +1$). In turn, every time a query term appears in a new relevant abstract, it is rewarded ($r_t = +1$). This algorithm updates the term rewards and demotes in each iteration. Therefore, after n iterations the accumulated rewards and demotes for query term t would be: $r_t = \sum_{i=1}^{n-1} r_{ti}$ and $d_t = \sum_{i=1}^{n-1} d_{ti}$.
- **Candidate Terms Finder.** This sub-component identifies new candidate search terms. These are the terms that frequently appear in relevant abstracts but are not part of the query yet. When all the terms are extracted from relevant abstracts, *Candidate Terms Finder* uses the TF-IDF of those terms as a filtering mechanism and selects candidate terms when their TF-IDF score is above a minimum threshold. Terms above this threshold are added to a *candidate terms* set. These candidate terms are then used by the *Query modifier* component when a term in a query must be replaced with a more suited term. The terms accumulated rewards and demotes are used by the *Terms Performance Estimator* to estimate the retrieval performance of each term.

Terms Performance Estimator. Having the terms with rewards and demotes, the only remaining question is: "how to choose a term for adding to or removing from a query?". We could address this question by relying on heuristic rules, such as adding the term with the highest rewards or removing one with the highest demotes. However, heuristic rules raise the questions: "how many relevant abstracts should contain a term to determine the term as a suitable choice for a query?" or " how many papers should be reviewed to find the optimal number of relevant papers?".

We used Thompson Sampling [53], in which the above question is answered by representing uncertainty as probability distribution [53, 35]. In our query adaptation problem, Thompson Sampling relies on a *policy* for deciding which term should be modified in a query. We utilized Thompson Sampling because it has provided near-optimal regret[4] bound in previous research [54, 36]. It has been successfully utilized in a range of applications, such as educational systems [35], gaming platforms [34], website optimisation [55], recommendation systems [56] and rule-based data annotation system [36].

The algorithm estimates a probability distribution, $\theta{=}Beta(r_t, d_t)$, for candidate term t using its accumulated rewards (r_t) and demotes (d_t). This distribution shows the expected reward when a particular term is chosen and also how variable reward is [35]. These features can affect the action that is taken (removing or adding a term). Each distribution has an initial value set based on a prior. The prior indicates our opinion about the performance of a term that has not yet been used in a query (candidate term). This distribution probability value is updated based on the algorithm observation (i.e., whether the term appears on relevant or irrelevant abstracts).

Each time the algorithm receives terms (with rewards and demotes), it updates their θ value based on their rewards and demotes.

When the algorithm obtains a set of candidate terms $C = \{t_1, t_2, ..., t_n\}$ along with their accumulated rewards and demotes, it updates the terms probability distributions, $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$, where $0 < \theta < 1$ using Bayes rule as:

$$P(\theta|t) = \frac{P(t|\theta)P(\theta)}{P(t)}. \tag{1}$$

It is assumed that each term t in iteration i has an initial prior of $\beta(1, 1)$. If $\beta(r_t, d_t)$ is the β value for term t in iteration i, then after observing a win $r = 1$, its β value would be $\beta(r_t + 1, d_t)$, conversely after observing a loss $d = 1$ its β value will be: $\beta(r_t, d_t + 1)$. The candidate term is then selected according to its probability (θ_t) that satisfies:

$$\text{argmax}_t\, P(\theta|t) = E[reward|\theta_t]p(\theta_t|A) \tag{2}$$

[4]The per-iteration regret is the mean of rewards of a choice with the best rewards and the action taken by the algorithm [53].

where A is the set of observed abstracts with feedback (relevant and irrelevant abstracts) and θ_t is the parameters of the Beta distribution for term t. It is worth noting that our query adaptation problem is a Bernoulli problem, meaning that the generated random variable by each term has only two possible outcomes: 0 or 1 and the value of $Beta(\alpha, \beta)$ is within the interval [0, 1]. Therefore, instead of the result varying per term, the probability of term generating rewards varies [57].

Query Modifier The final component of our query adaption is the *Query Modifier*, which modifies the query by adding to or removing terms from the query based on their observed performance. A Boolean query is a conjunction of query components, where each expression can have K terms that are connected using the *OR* operator. For instance, as illustrated in Figure 7, query q has two components and each expression has two terms (e.g., (t_1 OR t_2)). We refer to terms in an expression as sibling terms (e.g., t_1 is a sibling to t_2, and t_3 is a sibling to t_4). We define a query clause cl as a conjunction of a set of terms that are joined with *OR* operator(e.g., (t_1 AND t_3)). In other words, clauses are built by rewriting [58, 59] the query that contains the query components. We define a performance value p for each query clause cl as the accumulated rewards and demotes of all the terms within the query clause.

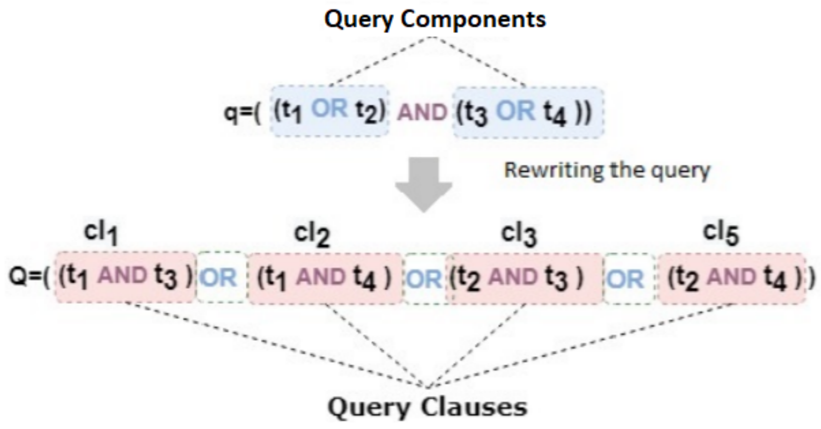

Query Components
q=((t1 OR t2) AND (t3 OR t4))
Rewriting the query
cl1
cl2
cl3
cl5
Q=((t1 AND t3) OR (t1 AND t4) OR (t2 AND t3) OR (t2 AND t4))
Query Clauses

Figure 7: Query components and query clauses

To adapt a query, the *Query Modifier* first identifies query clauses with performance values below a minimum threshold γ. The threshold characterizes the minimum accumulative rewards and demotes that a query clause should have to be considered efficient for query retrieval performance. After identifying inefficient query clauses, the *Query Modifier* finds the term in the clause that makes the clause inefficient. It compares the performance of the term with its sibling terms' performance. The *Query Modifier* will replace a term in the query clause if the number of relevant abstracts in which the term appears is below the average number of relevant abstracts in which its sibling terms appear. This indicates that the term is not useful in retrieving relevant abstracts. Moreover, the *Query Modifier* replaces this term with a *candidate term* estimated to yield the highest probability distribution (θ) by the Bayesian algorithm.

For instance, as illustrated in Figure. 8, suppose after i iterations the *Query Modifier* identifies that t_3 is insufficient in retrieving relevant papers. Thus, the *Query Modifier* removes t_3 and replaces it with t_6, a candidate term that has the highest probability value θ in iteration i.

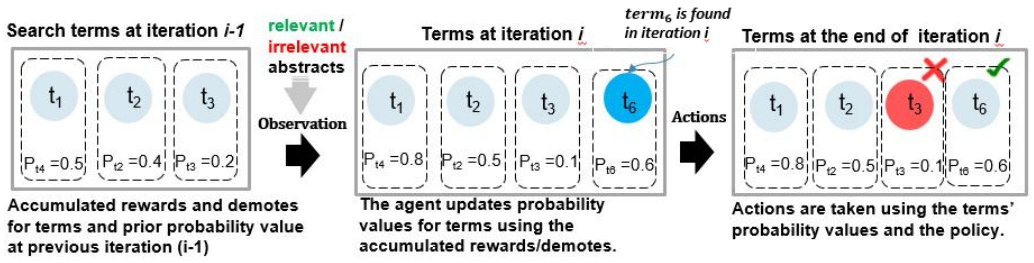

Search terms at iteration i-1
relevant / irrelevant abstracts
Observation
Terms at iteration i
term6 is found in iteration i
Terms at the end of iteration i
Actions
t1
t2
t3
t6
Pt4 =0.5
Pt2 =0.4
Pt3 =0.2
Pt4 =0.8
Pt2 =0.5
Pt3 =0.1
Pt6 =0.6
Accumulated rewards and demotes for terms and prior probability value at previous iteration (i-1)
The agent updates probability values for terms using the accumulated rewards/demotes.
Actions are taken using the terms' probability values and the policy.

Figure 8: Query modifier removes and adds terms in iteration i

3.4. Stopping Condition

The issue of determining the appropriate stopping point for the adaptation process is pertinent to the learning of RL algorithms. While there are currently no standard stopping rules for RL, some ad-hoc guidelines exist. However, these can lead to suboptimal results, such as overly long circuits or premature termination [60]. The optimal stopping problem is a mathematical framework that can be used to address the stopping issue in a stochastic system [61]. In this problem, at each interval, the decision maker must decide when to halt the process and receive a reward based on the current state or continue without immediate reward and potentially stop at a later time.

The stopping problem in the SLR query refinements problem can be tackled through various strategies, such as monitoring the query until it stabilizes, indicating that no further improvements can be made. Alternatively, the decision to end the adaptation can be left to the human, in this case, the SLR author. In our experiments, we have adopted the latter approach.

4. Evaluation

The main goal of the evaluation was to assess the two primary design decisions in our proposed pipeline: (i) *automatic generation* of the search query from the initial user input, and (ii) *incremental refinement* of the initial search query by leveraging user feedback. To this end, we designed three experiments that evaluated the *initial* and *refined* according to relevant performance metrics.[5].

In particular, (i) we investigate the effect of the information within the initial seeds on the automatic generation of search queries. We sought to evaluate the ability of the *Initial Query Builder* to construct search queries similar to those in our gold standard dataset, which consists of search queries from published SLRs. (ii) we analyze the impact of using general and domain-specific embedding models, assessing their impact on query enrichment and the effects on the performance of the generated queries. (iii) We evaluate the query adaptation and refinement in terms of its contribution to query performance while examining the impact of important decisions such as the number of iterations and sampling strategy.

4.1. Methods

Datasets. We performed the evaluation on SLR datasets that were made publicly available by their authors. To identify these datasets, we looked for SLRs in computer science that published their search and screening data, by systematically searching datasets at Zenodo and Figshare.[6] As a result, we identified 10 SLRs that included research questions, the SLR search query, the search result dataset, and the final relevance assessment. Of these SLRs, 5 also included the relevance assessment from the title and abstract screening phase. We consider the SLR authors' relevance assessment to be the gold

[5]For full details about the datasets, experimental details, and in-depth results, please refer to our supplementary material at `https://tinyurl.com/496zuar3` and implementation details on `https://tinyurl.com/2rp4m5cs`

[6]Popular dataset repositories, at `https://zenodo.org` and `http://figshare.com`

label in our experiments. In Table 1 we list the datasets employed in our experiments, along with some descriptive statistics.

Table 1: SLR datasets adapted in our experimental evaluation, along with descriptive statistics

Reference	Years Searched	#Candidate Papers	#Relevant Papers	Experiment No.
SLR_1[62]	2000-2013	7002	59	1,2,3
SLR_2[63]	2000-2013	8911	104	1,2,3
SLR_3[64]	1963-2011	6000	48	1,2,3
SLR_4[65]	2005-2013	1351	27	1,2,3
SLR_5[66]	2004-2007	1705	45	1,2,3
SLR_6[67]	2009-2014	384	36	1,3
SLR_7[68]	1996-2015	54	34	1,3
SLR_8[69]	2001-2019	5177	19	1,3
SLR_9[70]	2006-2018	244	75	1,3
SLR_{10}[71]	2014-2019	1585	49	1,3

Experiment 1- Query generation, and impact of seed. In this experiment, we assessed our approach to generating the search query directly from an initial seed, by applying the query generation and enrichment components. To understand the impact of the amount of information in the seed on these components, we tested three conditions having:**i)** only research questions (*GEN-RQ*), **ii)** abstracts from (1 or 3) relevant papers (*GEN-ABS(1,3)*); **iii)** and a combination including the research questions and one abstract (*GEN-RQA*). Notice that we only used the abstracts of the relevant papers for building the seeds, as they capture a meaningful summary of the paper. We generated search queries for the 10 SLR datasets, taking the seed from each SLR. For comparison, we took as baseline (*BASE-OG*) the performance of the original search queries from each SLR adapted and scoped to our target digital library and the fields currently supported (title, abstract). In these experiments we adopted a general word embedding model (Glove) in the query enrichment process. The detailed list of conditions compared during this experimental evaluation is shown in Table 2.

Experiment 2- Query generation, and impact of Word Embeddings models. Another important dimension to analyse when assessing the performance of the query generation, is the impact of the underlying word embedding model used for the query enrichment. We focus in particular on the

Table 2: Type of queries built using different seeds in the experiments

Seed	**Description**
GEN-RQ	These queries are built using only the research questions as the seed.
GEN-RQA	These queries are built using the research questions and only one randomly selected relevant abstract as the seed.
GEN-ABS	These queries are built using only one randomly selected relevant abstract as the seed.
GEN-ABS-3	These queries are built using three randomly selected relevant abstracts as the seed.

impact of using *general word embedding models* versus *domain-specific word embedding models*. One of the concerns in using general word embeddings is that models might not contain enough similar words for specific words from technical documents. For instance, the word "fork" in a software engineering context will have a specific meaning and set of similar words. Using a word-embedding model that is trained on the software engineering context would result in similar words such as {forking, fork/exec, forks, /execve, vfork} [72] that reflect the technical meaning of the word. However, using a general word-embedding model such as Google News word2vec [73], would result in similar words such as {pancake_turner, forking, wooden_skewer, ricer}, alluding to the utensil. In this experiment, we aim at comparing the performance of these two types of models, as they can lead to implications for the design, generalisation and required investment (e.g., in training) on the pipeline, when working with SLRs in new domains. Thus, for the general embedding model condition, we take the results from the first experiment which generated results using Glove. For the domain-specific word embedding model condition, we rely on a pre-trained domain-specific embedding for software engineering, namely SO embeddings [72][7]. This model is a word2vec model that is pre-trained on 15GB of Stack Overflow[8] posts. To make both conditions comparable, we adopt the same experimental conditions as in Experiment 1, except for the word embedding model as explained before.

[7]The pre-trained model is stored in a .bin file (of approximate size 1.5 GB) which can be accessed at this link: http://doi.org/10.5281/zenodo.1199620

[8]https://stackoverflow.com

Experiment 3- Query adaptation and refinement. Here, we evaluated the performance of the query adaptation based on author relevance feedback when incrementally refining the initial search query. To do so, we took the initial queries generated with research questions (*GEN-RQ*) in Experiment 1. Then we simulated the authors' feedback by leveraging the relevance assessment already present in the SLR datasets, i.e., screening relevance feedback made strictly based on the title and abstract.

We assessed the changes in query performance within 5 iterations, where for the sampling method we tested two different approaches: i) *uncertainty sampling*, where abstracts for feedback are drafted from those where the pipeline is less confident about their relevance (low similarity score); ii) *certainty sampling*, where abstracts are drafted from those the pipeline is most certain about their relevance (high similarity score). The idea of these two methods was to test the impact of the class distribution (ratio relevant to irrelevant papers) in authors' feedback since these contribute differently to the query adaptation. The uncertainty sampling returns predominately irrelevant papers, while the certainty sampling returns predominantly relevant papers.

For both sampling methods, we used 0.80 as the highest and 0.30 as the lower similarity thresholds for estimating relevance. The sampling size in the experiment was five papers for each SLR and in each iteration. We compared the performance of the query adaptation conditions against the initial generated search query (*GEN-RQ*) and the original search query (*BASE-OG*).

Data processing and analysis. In our analysis, we relied on metrics to help us capture how effective search queries are in identifying relevant papers while lowering the workload of the screening process. To capture the **effectiveness** we relied on standard precision and recall metrics applied to this context. *Precision* measures the proportion of retrieved relevant papers to the total number of retrieved papers. *Recall* computes the proportion of relevant papers retrieved to the total relevant papers in the relevance assessment dataset.

As a proxy for **workload**, we assess the *number of retrieved papers*. This number gives us an indication of the effort that authors would need to put into screening the search results for identifying relevant papers. As previously mentioned, a large number of retrieved studies will significantly impact the cost and effort required by the authors and could dictate the feasibility of performing the review. Notice that we take the number of relevant studies

Table 3: Performance of generated queries based on different seed input, compared to manually formulated queries by SLR authors (#Ret= #Retrieved; #Rel=#Relevant).

DS	#	BASE-OG			GEN-RQ			GEN-RQA			GEN-ABS-3		
	Rel	#Ret	Rec.	Prec.	#Ret	Rec.	Prec.	#Ret	Rec.	Prec.	#Ret	Rec.	Prec.
SLR_1	71	44,431	**0.59**	0.0009	996	0.37	0.0261	1045	0.40	0.0268	1,220	0.42	0.0246
SLR_2	208	5,852	**0.50**	0.0178	1,650	**0.50**	0.0630	1,780	0.51	0.0596	540	0.54	0.2074
SLR_3	89	3605	**0.47**	0.0117	552	0.45	0.0725	1,525	0.45	0.0262	1,049	0.37	0.0315
SLR_4	23	101	**0.95**	0.2178	260	0.83	0.0731	260	0.83	0.0731	146	0.87	0.1370
SLR_5	160	1,886	**0.28**	0.0239	13,300	0.24	0.0029	13,300	0.24	0.0029	2,019	0.28	0.0223
SLR_6	99	11,713	**0.93**	0.0079	350	0.80	0.2257	352	0.80	0.2244	220	0.59	0.2636
SLR_7	34	1,462	**0.76**	0.0178	169	0.35	0.0710	261	0.44	0.0575	1,245	0.38	0.0104
SLR_8	19	1,652	**0.95**	0.0109	800	0.74	0.0175	800	0.74	0.0175	205	0.42	0.0390
SLR_9	75	144	0.43	0.2222	730	**0.63**	0.0644	45	0.21	0.3556	79	0.16	0.1519
SLR_{10}	49	82,009	0.71	0.0004	604	0.82	0.0662	27,618	**0.84**	0.0015	785	0.49	0.0306
Median		2,746	0.66	0.0531	667	0.56	0.0653	923	0.48	0.0421	663	0.42	0.0357
Relative median performance					24.3%	85.6%	122.9%	33.6%	72.8%	79.3%	24.1%	64.1%	66.3%

from each SLR dataset as the gold standard. However, since most SLR datasets include results from multiple libraries, we recalculated the number of relevant papers to those that could be identified by the original query on our target digital library (IEEEXplore).

4.2. Results

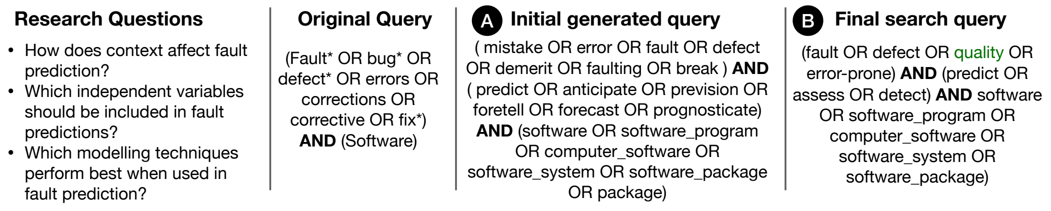
Experiment 1. The results of query generation are presented in Table 3[9], and the full details of the generated queries are available in the online Appendix. In comparison to the baseline (BASE-OG), queries generated with the best variant of our approach (GEN-RQ), achieve 85.6% of the median recall, and 122.9% of the accuracy of the expert queries, with only 24% of the results.

When comparing the impact of the three types of seed, we see the results to be similar across the various SLRs. However, the most consistent performance was achieved by GEN-RQ, i.e., it shows higher mean values of precision and recall with lower-to-comparable numbers of retrieved papers. These results suggest that, with our current approach, RQs are generally better for identifying key concepts for the search query than having one or three relevant abstracts. An inspection of the results tells us that, in addition to the type of seed (e.g., RQs or relevant abstracts), the information presented in the seed also impacts the quality of generated queries. For example, in our

[9]We only included the results of three seed types in the table, the full list is available in Appendix at https://tinyurl.com/496zuar3

experimental scenario, RQs are collected from published SLRs. Thus they are more likely to be properly formed and representative of the scope of the reviews. In contrast, the randomly selected relevant abstracts in the GEN-ABS and GEN-RQA approach may contain more repeating words, possibly introducing noise and reducing the relevance of important terms. However, going from a seed with one to three abstracts does show some performance increase in mean recall albeit with a higher number of results (*GEN-ABS-1*: Recall 58.8%, #Ret 21.5%; *GEN-ABS-3*: Recall 64.1%, #Ret 24%).

A closer look at the generated queries gives us hints into the characteristics of produced queries. As illustrated in Figure 9**A**, our approach is effective at identifying the main query components (e.g., fault, prediction, software). Yet some refinement (e.g., removing non-relevant terms) could improve its performance.



Research Questions	Original Query	A Initial generated query	B Final search query
• How does context affect fault prediction? • Which independent variables should be included in fault predictions? • Which modelling techniques perform best when used in fault prediction?	(Fault* OR bug* OR defect* OR errors OR corrections OR corrective OR fix*) **AND** (Software)	(mistake OR error OR fault OR defect OR demerit OR faulting OR break) **AND** (predict OR anticipate OR prevision OR foretell OR forecast OR prognosticate) **AND** (software OR software_program OR computer_software OR software_system OR software_package OR package)	(fault OR defect OR quality OR error-prone) **AND** (predict OR assess OR detect) **AND** software OR software_program OR computer_software OR software_system OR software_package)

Figure 9: Example queries generated for SLR$_2$ after the (A) initial query generation step (GEN-RQ) and (B) final query refinement iteration I_5 using certainty sampling

The above tells us that our approach provides a good foundation for generating queries and improving upon them. Having properly formulated research questions might not be the case in the early stages when planning a literature review process. Therefore, the possibility of having relevant abstracts as seeds provide a solid base for composing and refining search queries.

Experiment 2. Next, we investigated how the use of a domain-specific embedding model in the query enrichment component may affect the generation of initial queries. For this, we compare the outcomes from Experiment 1 generated using Glove (general embedding model) with the performance of the queries generated with SO embeddings, a domain-specific embedding model for software engineering. The results of this experiment are presented in Table 4[10]. In comparison to the baseline (BASE-OG), queries generated with

[10]The details of the generated queries is presented in the online appendix available at

the best variant of our approach (GEN-RQ), achieve 76.8% of the median recall of the manual approach, with 65% of the results (number of retrieved articles).

Table 4: Performance of generated queries based on different seed input, compared to manually formulated queries by SLR authors (#Ret= #Retrieved; #Rel=#Relevant) using SO embeddings. We include the median performance across SLRs for the domain-specific embeddings condition (Mean SO) as well as the mean for the general embeddings condition (Mean Glove) from Experiment 1

DS	**# Rel**	**BASE-OG**			**GEN-RQ**			**GEN-RQA**			**GEN-ABS-3**		
		#Ret	**Rec.**	**Prec.**	**#Ret**	**Rec.**	**Prec.**	**#Ret**	**Rec.**	**Prec.**	**#Ret**	**Rec.**	**Prec.**
SLR_1	71	44,431	**0.59**	0.0009	2,602	0.32	0.0088	7,024	0.45	0.0046	11,195	0.45	0.0029
SLR_2	208	5,852	0.50	0.0178	2,469	0.43	0.0360	43	0.44	0.1395	552	**0.51**	0.1920
SLR_3	89	3605	**0.47**	0.0117	634	0.42	0.0584	835	0.44	0.0467	1,074	0.39	0.0326
SLR_4	23	101	**0.95**	0.2178	1,007	0.83	0.0189	1,007	0.83	0.0189	2,060	0.87	0.0097
SLR_5	160	1,886	**0.28**	0.0239	20,738	0.24	0.0018	13,326	0.20	0.0024	2,019	**0.28**	0.0223
SLR_6	99	11,713	**0.93**	0.0079	351	0.74	0.2080	351	0.80	0.2251	4,011	0.23	0.0090
SLR_7	34	1,462	**0.76**	0.0178	91	0.29	0.1099	262	0.44	0.0573	80	0.59	0.7250
SLR_8	19	1,652	**0.95**	0.0109	1,657	0.68	0.0078	800	0.74	0.0175	120	0.42	0.0667
SLR_9	75	144	0.43	0.2222	1,033	**0.63**	0.0455	32	0.21	0.5000	79	0.13	0.1266
SLR_{10}	49	82,009	0.71	0.0004	23,235	**0.82**	0.0017	24,965	0.54	0.0016	770	0.53	0.0338
Med. SO		2,746	0.66	0.0531	1,345	0.53	0.0275	818	0.45	0.0328	922	0.44	0.0332
Med. Glove		2,746	0.66	0.0531	667	0.56	0.0653	923	0.48	0.0421	663	0.42	0.0357

Looking at the impact of the amount of information in the seed, we can see that the queries generated with the domain-specific model exhibit a relative performance that is consistent with Experiment 1. This suggests that the amount of information in the seed has a visible and consistent impact on the performance of the query, for the conditions under study, independently of the model.

When comparing the impact of the model on the performance, we can see that - surprisingly - the general embedding condition performs consistently better than the domain-specific embedding model, for GEN-RQ and GEN-RQA, and partially for GEN-ABS-3 (except for recall). A qualitative inspection of the generated queries revealed some potential explanations for these results, which we describe by referring to the search terms in Table 5. SO embeddings were better at expanding on technical terms. As seen in the table, technical terms such as "fault" were expanded into terms such as "bug", "debug" and others, which are in this context closer than the ones produced by the general model (e.g., "guilt" or "fracture"). However, when it

https://tinyurl.com/496zuar3

came to general terms such as "predict", the general model provided a much richer set of options. Ultimately, the ability to produce domain-specific terms was overshadowed by the inability to work with general concepts, which are also present in query components. It should be noted, however, that the SO embedding model is trained on StackOverflow, and not on software engineering papers.

The implications of these results for pipeline design pipeline are two-fold. i) General models can provide solid performance in the generation of the initial query, despite their inability to capture domain-specific terms. Relying on domain-specific models does not necessarily guarantee better performance, especially if the training data does not reflect the use case. ii) There is an opportunity to combine general embedding models and domain-specific models to make up for the limitations of both approaches. However, the goal at this step is the generation of an initial query that can be refined based on user feedback.

Table 5: Example of search terms along with the enriched list of words using Glove and SO embeddings

Term	**Search terms using Glove**	**Search terms using SO**
fault	guilt, fracture, defect, mistake, demerit	error, defect, bug, mistake, debug, break
predict	anticipate, foresee, prevision, foretelling, forecasting, prognostication	guess, prognosis, forecast
model	role_model, modelling, poser, exemplary, good_example	framework, simulation, pattern, simulate, mock_up

Experiment 3 The results of the query adaptation and refinement step, for the uncertainty and certainty sampling, are shown in Table 6. Compared to the **input query** generated by GEN-RQ (I_0), the final refined queries consistently improved on the recall when applying either of the two sampling methods, though, the improvements are more pronounced for certainty sampling. However, the improvements came at the cost of an increase in the number of retrieved results for 3 of the 5 SLRs. Precision was also improved only in 2 of the 5 SLRs (refer to the supplementary materials for precision results). Looking at performance through the iterations (see Figure 4.2, we can see that uncertainty sampling improves more slowly, or stalls in terms of

Table 6: Performance of the query refinement approach for two competing sampling methods. Values are shown for the first five iterations (I_1 - I_5). Performance is compared to the input query ($I_0^{\uparrow}$) generated by GEN-RQ, and the baseline query (Q$^{\uparrow}$) BASE-OG.

DS	**Uncertainty**							**Certainty**						
Recall	I_1	I_2	I_3	I_4	I_5	$I_0^{\uparrow}$	$Q^{\uparrow}$	I_1	I_2	I_3	I_4	I_5	$I_0^{\uparrow}$	Q$^{\uparrow}$
SLR_1	0.37	0.38	0.44	0.44	0.56	51%	-5%	0.46	0.51	0.56	0.61	0.66	78%	12%
SLR_2	0.51	0.51	0.52	0.52	0.54	8%	8%	0.93	0.95	0.97	0.99	**1.00**	100%	100%
SLR_3	0.45	0.45	0.48	0.55	0.60	33%	28%	0.83	0.88	0.92	0.96	**1.00**	122%	113%
SLR_4	0.83	0.87	0.87	0.87	0.87	5%	-8%	0.85	0.85	0.88	0.91	0.95	14%	0%
SLR_5	0.24	0.24	0.24	0.26	0.26	8%	-7%	0.35	0.35	0.35	0.44	0.53	121%	89%
#Ret	I_1	I_2	I_3	I_4	I_5	$I_0^{\uparrow}$	$Q^{\uparrow}$	I_1	I_2	I_3	I_4	I_5	$I_0^{\uparrow}$	Q$^{\uparrow}$
SLR_1	996	862	813	756	952	-4%	-98%	1320	992	855	767	705	-29%	-98%
SLR_2	2052	2038	1904	1760	2362	43%	-60%	6893	6861	5528	5477	5161	213%	-12%
SLR_3	1050	890	1473	1563	1420	157%	-61%	6904	7120	7311	7366	7542	1266%	109%
SLR_4	315	420	421	495	452	74%	348%	1955	1777	1687	1674	1561	500%	1446%
SLR_5	12754	9835	8920	8635	7432	-44%	294%	4667	3733	3237	3352	2423	-82%	28%

recall, compared to certainty sampling. The increase in the number of papers in search results is also more conservative in the uncertainty sampling approach. This can be attributed to the fact that negative samples (irrelevant abstracts) contribute more to removing irrelevant terms.

To contextualise the practical implications of the higher recall at the cost of more search results, we compare refined queries to the queries formulated by the SLR authors. When comparing the performance to the **baseline query** (BASE-OG), the difference is more noticeable. Certainty sampling was significantly improving (or matching) the recall of the expert-formulated queries BASE-OG (Q). As seen in the grayed-out cells, this was achieved while reducing the number of retrieved papers in two instances (SLR_1, SLR_2) and with a modest increase in other cases. The outlier with a significant increase is due to the unusual case of having only 101 search results in the baseline (SLR_4). What these results tell us is that our approach is **able to match or improve the performance of expert formulated queries** when it comes to recall, while adapting to different SLRs by learning from author feedback. All of this, reducing at times the number of results – though results in this regard are inconclusive. Notice that achieving a high recall, while reducing the risk of missing relevant work, is an important aspect of SLRs, and our approach is able to build and refine queries to support this goal.

As illustrated in Figure 9, the refinement process is indeed able to reduce the non-relevant search terms (e.g., for "fault", removing "demerit") and add

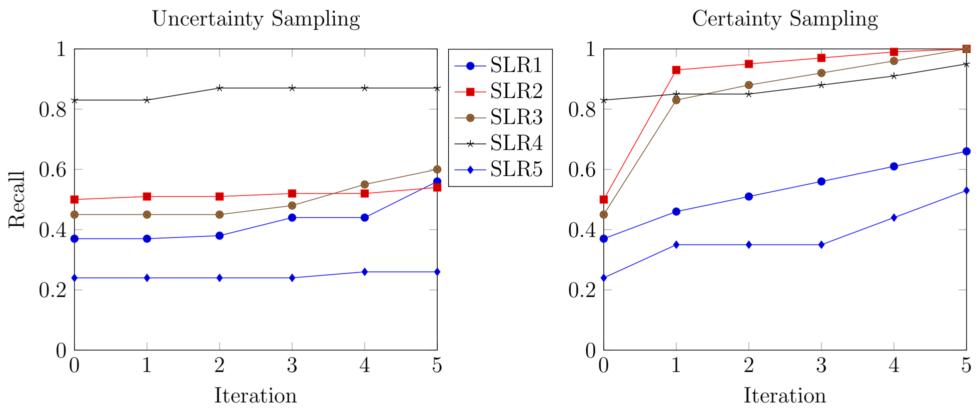


Figure 10: Performance of sampling methods on recall, over iterations I_0-I_5

relevant terms (e.g., for "fault" adding "quality") to the query, contributing to better performance. However, we should also note that the generated query is currently not the optimal expression of the query, as we can still find redundant terms (e.g., having "software" makes the term "computer software" redundant). To inspect all the generated queries, please refer to the online Appendix.

5. Discussion

In this discussion section, we delve into key insights derived from our experimental results and their implications for SLR query search formulation. By examining the impact of different seed types, word embedding models, and query refinement strategies, our findings contribute to a better understanding of the potential benefits and limitations of our approach in addressing the challenges of the SLR search process.

Importance of properly formulated research questions. The results highlight the importance of properly formulated research questions in the early stages of planning a literature review. If available, RQs can achieve a median recall of 86% with 24% of the results compared to the expert-formulated queries. Thus, the type of seed does have an impact, and it motivates further exploration of the specific factors determining the increase in performance.

Value of abstracts as seeds for researchers with limited context. While queries generated using abstracts as seeds may not be as effective as those generated from research questions, they still offer a solid starting point for the query refinement process. This is particularly valuable for users with limited domain expertise, allowing them to iteratively compose and refine search queries while benefiting from the improvements observed during the refinement process. Our analysis also indicates that increasing the number of input abstracts can lead to better query performance, which could improve the starting point for refining their search queries if authors have more sample papers available.

Impact of Domain-specificity of Word Embeddings on Query Generation. We recognize that a model trained on academic papers in a specific domain (e.g., software engineering, medicine) might provide better semantic representations, capturing the nuances of academic research in the a specific domain. Developing such a model from scratch would be challenging due to the need for a large corpus of academic papers and computational resources for each domain. Nevertheless, our findings are encouraging, demonstrating the feasibility of leveraging general models. This is particularly relevant with the advent of powerful large language models (LLMs) like GPT-4, which are pre-trained on massive datasets from diverse sources.

Added value of query adaptation and refinement. The experimental results show that our pipeline is able to produce refined queries that can match or improve the performance of expert-formulated queries in terms of recall while adapting to different SLRs by learning from author feedback. Certainty sampling resulted in a better approach, featuring a more pronounced improvement over the 5 iterations of our experimental settings. Indeed, the rate of improvement we observed suggests that there is an opportunity for further improvement beyond the 5 iterations, which should be an aspect to address when studying stopping conditions.

6. Limitations

While this approach and its evaluation showed some initial promise in generating search queries, there are limitations to the current pipeline and the experimental evaluation that we address below.

Query expressiveness limitations. The current pipeline generates and refines queries based on the *AND* and *OR* boolean operations. We do not currently support the *NOT* operator, and we acknowledge that this impacts the query expressiveness and is likely to affect the performance of the formulated queries. However, our approach was able to match, and in some cases, improve the performance compared to expert-formulated queries even with this simple setup, indicating room for improvement. We leave the exploration of the impact of using the NOT operator for generating and refining queries to future research.

Relative importance of research studies. Our approach does not consider the relative importance of papers based on factors such as type and source. Considering the relative importance of papers could allow authors to give more authority to journal papers or top venues. While interesting, investigating these aspects was deemed outside the scope of our current paper.

SLR datasets as gold-standard for query performance. The adopted SLR datasets could have potentially missed relevant work that was correctly retrieved by the generated queries but deemed as false positives in our experimental setting. To address this, it is important to consider more comprehensive studies that include actual assessment of relevance by humans (e.g., authors) or running actual pilots with ongoing SLRs. We leave these more comprehensive studies to future work.

Simulation of query refinement with SLR datasets. The query refinement step was simulated by leveraging the relevant assessment already made by SLR authors and present in the dataset. While this was a valuable strategy to explore the performance of the refinement approach, we acknowledge that it has potential limitations. For example, the simulation might not reflect the actual behavior of authors in refining the search and interacting with search results. Therefore, pilots with real users in an actual refinement process will be required to analyze the impact on real settings and user behavior.

7. Conclusion and Future Work

In summary, our evaluation showed that our approach is able to generate effective queries from high-level expressions of the scope of a review. The initial query generation and enrichment process is able to generate search queries that deliver 87.7% of the median recall of the manual approach, with only 24% of the results (number of items retrieved). While a solid approximation, we observed this process to be sensible to the amount of information provided in the seed and limited by the query enrichment methods that might introduce non-semantically relevant terms to the query. On this, the proposed query adaptation is able to significantly improve on the initial generated query and archives the performance (and in some cases improve) of expert formulated queries in the course of 5 iterations with 5 relevance feedback each. Sampling methods were shown to have a significant impact, with a sampling strategy biased toward positive examples found to be the most effective. Overall, our empirical evaluation has shown the potential of the reinforcement learning approach to adapt search queries based on author feedback, without the need for domain-specific training.

We consider the following areas for future work as particularly critical to guarantee quality and sustainability in the longer term:

Understanding the acceptance of the community of SLRs that are partially automated using the available support tools and techniques. Understanding the readiness of the research community to launch a machine-based SLR has been discussed in other works. We believe it would also be an interesting research direction to investigate the acceptance of the community of SLRs that are partially automated using the available support tools and techniques. Thus, as we develop more effective automated solutions, the question still remains whether and to what extent we should embrace automation in our community, calling us to reflect on the consequences and revisit traditional notions such as authorship, accountability, and ethics in general. A starting point is to provide guidelines to tool developers and researchers. The guidelines could propose how to design and validate such automation techniques and set the desired governance models for the responsible use of automation, especially AI-enabled automation in the SLR process.

A future work direction for us is to evaluate the acceptance of the proposed approach by SLR authors. We believe that providing the proposed pipeline in the form of a service to conduct a study with researchers who plan to conduct an SLR is a major step to improve the SLR process.

Exploration of techniques that have shown success in automation of SLR tasks to automate unexplored tasks. The majority of adopted automation techniques for SLR tasks focus on adopting the usual machine learning approach (e.g., text classification). However, these techniques may not hold enough capabilities to aid complex tasks(e.g., summarizing research findings). The reason behind this might be that these tasks are highly cognitive and require acute human judgment. Moreover, humans are superior to machines in fulfilling these tasks, whereas in other tasks that require consistent iterations (e,g., study selection) the machines outperform human capacity. We believe that future research should address these gaps through cognitive computing systems, to promote collaboration between humans and machines [74, 2].

These human-machine techniques have shown promising results in obtaining efficiency and effectiveness in automating specific SLR tasks (e.g., active learning for citation screening [4, 75]). We believe that leveraging these techniques to support other complex research tasks (e.g., coding in qualitative data analysis, summarizing research findings, seeking evaluation of research techniques) is clearly transformable. This can provide researchers with similar experience that search engines and social media platforms provide for Web search and people interactions, but for performing research tasks and collaborating with others, reducing information overload and automating repetitive tasks. The challenge is to support researchers in improving research processes including the SLR process with automation while ensuring compliance with policies and rules (e.g., research ethics and privacy). This capability can be powered by a number of AI-enabled techniques such as query intend discovery, entity mention discovery, knowledge graphs and deep learning algorithms (e.g., to support entity and relationships-based indexing over research studies).

Gathering empirical evidence on the comparative performance of existing models. Our literature review on automation techniques for facilitating SLR, indicates that there is not enough evidence on how machine learning models compare in terms of their performance when reporting the results. Empirical studies that look at systematically comparing models or tools under relevant domain-specific conditions for SLR automation are necessary. These studies could help to have a clear understanding of the strengths and limitations of these techniques. Besides, researchers' perspectives about adopting these techniques are missing. A more human-centered approach to the development and evaluation of the models could help to shape more efficient and realistic SLR automation techniques.

Evaluation frameworks and reporting guidelines to ensure the strength of evidence, transparency, and replicability of results. One limitation that affects the appraisal of the proposed techniques (e.g, machine learning models) in automating SLR tasks is that the models are treated as a 'black box' when presented in the studies(e.g., [76, 77, 78, 27]). For instance, the data preparation steps (e.g., selecting the train datasets and feature extractions) are not clearly described. Even though the majority of the studies provide evaluation results, they do not explain how and why they used a certain evaluation method. One potential research direction that could benefit SLR automation research is developing benchmarks and frameworks for evaluating, reporting, and comparing SLR automation techniques.

These frameworks should take into account the specifics of the SLR process, current practices, barriers to proper experimental evaluation, and opinions of the community, so as to provide value and facilitate adoption. For example, frameworks could include datasets for training machine learning algorithms and standard evaluation metrics for reporting the results. The goal is to facilitate the adoption of the developed techniques, standardise the sharing of outcomes, models, and datasets when reporting the results. It could also help to conduct more experimental studies for appraising SLR automation techniques across different research domains (e.g, medical and software engineering).

References

[1] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering (2007).

[2] M. Badami, M. Baez, S. Zamanirad, et al., On how cognitive computing will plan your next systematic review, arXiv preprint arXiv:2012.08178 (2020).

[3] H. Scells, G. Zuccon, B. Koopman, Automatic boolean query refinement for systematic review literature search, in: WWW, 2019, pp. 1646–1656.

[4] B. C. Wallace, K. Small, C. E. Brodley, et al., Who should label what? instance allocation in multiple expert active learning, in: SDM, SIAM, 2011, pp. 176–187.

[5] H. Li, H. Scells, G. Zuccon, Systematic review automation tools for end-to-end query formulation, in: Proc. 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2020, pp. 2141–2144.

[6] S. Marcos-Pablos, F. J. García-Peñalvo, Decision support tools for slr search string construction, in: Proc. of TEEM'18, 2018, pp. 660–667.

[7] G. D. Mergel, M. S. Silveira, T. S. da Silva, A method to support search string building in systematic literature reviews through visual text mining, in: Proc. of the 30th Annual ACM Symposium on Applied Computing, 2015, pp. 1594–1601.

[8] Y. Kim, J. Seo, W. B. Croft, Automatic boolean query suggestion for professional search, in: Proceedings of SIGIR, 2011, pp. 825–834.

[9] M. Badami, B. Benatallah, M. Baez, Systematic literature review search query refinement pipeline: Incremental enrichment and adaptation, in: International Conference on Advanced Information Systems Engineering, Springer, 2022, pp. 129–146.

[10] H. Zhang, M. A. Babar, P. Tell, Identifying relevant studies in software engineering, Information and Software Technology 53 (6) (2011) 625–637.

[11] J. Clark, Systematic reviewing, in: Methods of clinical epidemiology, Springer, 2013, pp. 187–211.

[12] E. Hausner, C. Guddat, T. Hermanns, U. Lampert, S. Waffenschmidt, Development of search strategies for systematic reviews: validation showed the noninferiority of the objective approach, Journal of clinical epidemiology 68 (2) (2015) 191–199.

[13] D. A. Buell, A general model of query processing in information retrieval systems, Information Processing & Management 17 (5) (1981) 249–262.

[14] H. Scells, G. Zuccon, B. Koopman, A comparison of automatic boolean query formulation for systematic reviews, Information Retrieval Journal 24 (1) (2021) 3–28.

[15] R. van Dinter, B. Tekinerdogan, C. Catal, Automation of systematic literature reviews: A systematic literature review, Information & Soft. Tech. (2021) 106589.

[16] S. Karimi, S. Pohl, F. Scholer, et al., Boolean versus ranked querying for biomedical systematic reviews, BMC 10 (1) (2010) 1–20.

[17] D. Martinez, S. Karimi, L. Cavedon, T. Baldwin, Facilitating biomedical systematic reviews using ranked text retrieval and classification, in: Australasian Document Computing Symposium (ADCS), 2008, pp. 53–60.

[18] T. Russell-Rose, P. Gooch, 2dsearch: A visual approach to search strategy formulation (2018).

[19] H. Scells, G. Zuccon, searchrefiner: A query visualisation and understanding tool for systematic reviews, in: Proc. of CIKM, 2018, pp. 1939–1942.

[20] H. Scells, G. Zuccon, , et al., Integrating the framing of clinical questions via pico into the retrieval of medical literature for systematic reviews, in: Proc. of CIKM, 2017, pp. 2291–2294.

[21] H. Scells, G. Zuccon, Generating better queries for systematic reviews, in: ACM SIGIR, 2018, pp. 475–484.

[22] H. Scells, L. Azzopardi, G. Zuccon, et al., Query variation performance prediction for systematic reviews, in: SIGIR, 2018, pp. 1089–1092.

[23] M. Riaz, M. Sulayman, N. Salleh, et al., Experiences conducting systematic reviews from novices' perspective, in: EASE, 2010, pp. 1–10.

[24] J. C. Carver, E. Hassler, E. Hernandes, N. A. Kraft, Identifying barriers to the systematic literature review process, in: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, IEEE, 2013, pp. 203–212.

[25] M. Badami, Automated and improved search query effectiveness design for systematic literature reviews, Ph.D. thesis, UNSW Sydney (2021).

[26] M. Ouzzani, H. Hammady, Z. Fedorowicz, A. Elmagarmid, Rayyan—a web and mobile app for systematic reviews, Systematic reviews 5 (1) (2016) 210.

[27] A. M. Cohen, N. R. Smalheiser, et al., Automated confidence ranked classification of randomized controlled trial articles: an aid to evidence-based medicine, Journal of the American Medical Informatics Association 22 (3) (2015) 707–717.

[28] B. C. Wallace, K. Small, C. E. Brodley, J. Lau, T. A. Trikalinos, Modeling annotation time to reduce workload in comparative effectiveness reviews, in: Proc. of the 1st ACM International Health Informatics Symposium, 2010, pp. 28–35.

[29] B. C. Wallace, T. A. Trikalinos, J. Lau, C. Brodley, C. H. Schmid, Semi-automated screening of biomedical citations for systematic reviews, BMC bioinformatics 11 (1) (2010) 1–11.

[30] P. Przybyła, A. J. Brockmeier, G. Kontonatsios, et al., Prioritising references for systematic reviews with robotanalyst: a user study, Research synthesis methods 9 (3) (2018) 470–488.

[31] A. M. Cohen, W. R. Hersh, et al., Reducing workload in systematic review preparation using automated citation classification, Journal of the American Medical Informatics Association 13 (2) (2006) 206–219.

[32] C. R. Norman, M. M. Leeflang, R. Porcher, A. Neveol, Measuring the impact of screening automation on meta-analyses of diagnostic test accuracy, Systematic reviews 8 (1) (2019) 1–18.

[33] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. De Freitas, Taking the human out of the loop: A review of bayesian optimization, Proceedings of the IEEE 104 (1) (2015) 148–175.

[34] Y.-E. Liu, T. Mandel, E. Brunskill, Z. Popovic, Trading off scientific knowledge and user learning with multi-armed bandits., in: EDM, 2014, pp. 161–168.

[35] J. J. Williams, J. Kim, A. Rafferty, et al., Axis: Generating explanations at scale with learnersourcing and machine learning, in: L@Scale, 2016, pp. 379–388.

[36] A. Tabebordbar, A. Beheshti, B. Benatallah, et al., Feature-based and adaptive rule adaptation in dynamic environments, DSE 5 (3) (2020) 207–223.

[37] T. V. Ribeiro, J. Massollar, G. H. Travassos, Challenges and pitfalls on surveying evidence in the software engineering technical literature: an exploratory study with novices, Empirical Software Engineering 23 (3) (2018) 1594–1663.

[38] M.-A. Yaghoub-Zadeh-Fard, B. Benatallah, F. Casati, et al., Dynamic word recommendation to obtain diverse crowdsourced paraphrases of user utterances, in: Proc. of IUI, 2020, pp. 55–66.

[39] S. Kuzi, A. Shtok, O. Kurland, Query expansion using word embeddings, in: Proc. of CIKM, 2016, pp. 1929–1932.

[40] S. Wang, H. Scells, B. Koopman, G. Zuccon, Can chatgpt write a good boolean query for systematic review literature search?, arXiv preprint arXiv:2302.03495 (2023).

[41] C. D. Manning, M. Surdeanu, et al., The stanford corenlp natural language processing toolkit, in: Proc. of ACL, 2014, pp. 55–60.

[42] V. Garousi, M. Felderer, Experience-based guidelines for effective and efficient data extraction in systematic reviews in software engineering, in: Proc. of EASE'17, 2017, pp. 170–179.

[43] C. Carpineto, G. Romano, A survey of automatic query expansion in information retrieval, Acm Computing Surveys (CSUR) 44 (1) (2012) 1–50.

[44] G. A. Miller, WordNet: An electronic lexical database, MIT press, 1998.

[45] T. Mikolov, I. Sutskever, K. Chen, et al., Distributed representations of words and phrases and their compositionality, in: NeurIPS, 2013, pp. 3111–3119.

[46] S. Imtiaz, M. Bano, N. Ikram, et al., A tertiary study: experiences of conducting systematic literature reviews in software engineering, in: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, 2013, pp. 177–182.

[47] R. Kohavi, R. Longbotham, D. Sommerfield, et al., Controlled experiments on the web: survey and practical guide, DMKD 18 (1) (2009) 140–181.

[48] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT, 2018.

[49] G. E. Lee, A. Sun, Seed-driven document ranking for systematic reviews in evidence-based medicine, in: SIGIR, 2018, pp. 455–464.

[50] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proc. of EMNLP, 2014, pp. 1532–1543.

[51] G. Salton, C. Buckley, Improving retrieval performance by relevance feedback, Journal of the American society for information science 41 (4) (1990) 288–297.

[52] V. Lavrenko, W. B. Croft, Relevance-based language models, in: ACM SIGIR Forum, Vol. 51, ACM New York, NY, USA, 2017, pp. 260–267.

[53] D. Russo, B. Van Roy, A. Kazerouni, et al., A tutorial on thompson sampling, arXiv preprint arXiv:1707.02038 (2017).

[54] S. Agrawal, N. Goyal, Analysis of thompson sampling for the multi-armed bandit problem, in: Conference on learning theory, 2012, pp. 39–1.

[55] D. N. Hill, H. Nassif, Y. Liu, et al., An efficient bandit algorithm for realtime multivariate optimization, in: KDD'17, 2017, pp. 1813–1821.

[56] J. Kawale, H. H. Bui, et al., Efficient thompson sampling for online matrix-factorization recommendation, Advances in neural information processing systems 28 (2015) 1297–1305.

[57] E. Brochu, V. M. Cora, et al., A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv preprint arXiv:1012.2599 (2010).

[58] R. Xiao, J. Ji, B. Cui, H. Tang, W. Ou, Y. Xiao, J. Tan, X. Ju, Weakly supervised co-training of query rewriting andsemantic matching for e-commerce, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 402–410.

[59] C.-C. K. Chang, H. Garcia-Molina, A. Paepcke, Predicate rewriting for translating boolean queries in a heterogeneous information system, ACM Transactions on Information Systems (TOIS) 17 (1) (1999) 1–39.

[60] E. Even-Dar, S. Mannor, Y. Mansour, S. Mahadevan, Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems., Journal of machine learning research 7 (6) (2006).

[61] A. Fathan, E. Delage, Deep reinforcement learning for optimal stopping with application in financial engineering, arXiv preprint arXiv:2105.08877 (2021).

[62] R. S. Wahono, A systematic literature review of software defect prediction, Journal of Software Engineering 1 (1) (2015) 1–16.

[63] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Transactions on Software Engineering 38 (6) (2011) 1276–1304.

[64] D. Radjenović, M. Heričko, R. Torkar, A. Živkovič, Software fault prediction metrics: A systematic literature review, Information and software technology 55 (8) (2013) 1397–1418.

[65] P. Jamshidi, A. Ahmad, C. Pahl, Cloud migration research: a systematic review, IEEE transactions on cloud computing 1 (2) (2013) 142–157.

[66] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Information and software technology 51 (1) (2009) 7–15.

[67] A. Barišić, M. Goulão, V. Amaral, Domain-specific language domain analysis and evaluation: a systematic literature review, Faculdade de Ciencias e Technologia, Universidade Nova da Lisboa (2015).

[68] M. Frank, M. Hilbrich, S. Lehrig, S. Becker, Parallelization, modeling, and performance prediction in the multi-/many core area: A systematic literature review, in: 2017 IEEE 7th International Symposium on Cloud and Service Computing (SC2), IEEE, 2017, pp. 48–55.

[69] G. Adamo, C. Ghidini, C. Di Francescomarino, What is a process model composed of? a systematic literature review of meta-models in bpm, arXiv preprint arXiv:2011.09177 (2020).

[70] C. Qin, H. Eichelberger, K. Schmid, Enactment of adaptation in data stream processing with latency implications—a systematic literature review, Information and Software Technology 111 (2019) 1–21.

[71] E. N. Teixeira, F. A. Aleixo, F. D. de Sousa Amâncio, E. OliveiraJr, U. Kulesza, C. Werner, Software process line as an approach to support software process reuse: A systematic literature review, Information and Software Technology 116 (2019) 106175.

[72] V. Efstathiou, C. Chatzilenas, D. Spinellis, Word embeddings for the software engineering domain, in: Proc. of the international Conf. on mining software repositories, 2018, pp. 38–41.

[73] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[74] M. C. Barukh, S. Zamanirad, M. Baez, A. Beheshti, B. Benatallah, F. Casati, L. Yao, Q. Z. Sheng, F. Schiliro, Cognitive augmentation in processes, in: Next-Generation Digital Services - Essays Dedicated to Mike Papazoglou, Vol. 12, Springer, 2020.

[75] G. Kontonatsios, A. J. Brockmeier, P. Przybyła, J. McNaught, T. Mu, J. Y. Goulermas, S. Ananiadou, A semi-supervised approach using label propagation to support citation screening, Journal of biomedical informatics 72 (2017) 67–76.

[76] G. Rizzo, F. Tomassetti, A. Vetro, L. Ardito, M. Torchiano, M. Morisio, R. Troncy, Semantic enrichment for recommendation of primary studies in a systematic literature review, Digital Scholarship in the Humanities 32 (1) (2017) 195–208.

[77] B. K. Olorisade, P. Brereton, P. Andras, The use of bibliography enriched features for automatic citation screening, Journal of biomedical informatics 94 (2019) 103202.

[78] I. J. Marshall, A. Noel-Storr, J. Kuiper, J. Thomas, B. C. Wallace, Machine learning for identifying randomized controlled trials: an evaluation and practitioner's guide, Research synthesis methods 9 (4) (2018) 602–614.