Systematic Literature Review Search Query Refinement Pipeline: Incremental Enrichment and Adaptation

Maisie Badami[1], Boualem Benatallah[1], and Marcos Baez[2]

University of New South Wales (UNSW), Australia
{m.badami,b.benatallah}@unsw.edu.au
LIRIS – University of Claude Bernard Lyon 1, Villeurbanne, France
marcos.baez@liris.cnrs.fr

Abstract. Systematic literature reviews (SLRs) are at the heart of evidence-based research, collecting and integrating empirical evidence regarding specific research questions. A leading step in the search for relevant evidence is composing Boolean search queries, which are still at the core of how information retrieval systems work to perform an advanced literature search. Building these queries thus requires going from the general aims of the research questions into actionable search terms that are combined into potentially complex Boolean expressions. Researchers are thus tasked with the daunting and challenging task of building and refining search queries in their quest for sufficient coverage and proper representation of the literature. In this paper, we propose an adaptive Boolean query generation and refinement pipeline for SLR search. Our approach utilizes a reinforcement learning technique to learn the optimal modifications for a query based on the feedback collecting from the researchers about the query retrieval performance. Empirical evaluations with 10 SLR datasets showed our approach to achieve comparable performance to that of queries manually composed by SLR authors.

Keywords: Systematic Reviews · Query building · Query Enrichment · Query Adaptation · Reinforcement Learning

1 Introduction

Systematic literature reviews (SLRs) offer robust and transferable evidence for evaluating and interpreting relevant research on a topic of interest [13]. SLRs are used to set the foundation for future research, allowing researchers to formally plan and systematically collect and integrate available evidence in order to answer research questions [2]. Given their demonstrated value, SLRs are becoming a popular type of publication in empirical research domains such as evidence-based software engineering [13]. The identification of relevant studies is a fundamental step in the SLR process, as it can impact the overall *quality* and *workload* of the SLRs. This step is guided by the definition of research questions that set the tone and scope for the entire SLR [13,2]. Researchers are then tasked with capturing this scope in a search strategy that typically involves composing Boolean search queries for scientific digital libraries [13].

The ability of the search query to capture relevant works will determine whether the review has a proper representation of the literature for accurate synthesis of the literature downstream of the process [31]. It also greatly impact the workload as researchers need to screen the volume of returned results to filter out irrelevant work. It is not thus surprising that a large part of the SLR effort goes to the identification of relevant studies [37]. To make things more complicated for such a critical task, SLR authors might have limited knowledge of the review topic and the search terms at the beginning of the process, and the primary studies themselves may adopt different terminology to refer to similar concepts [17]. All these circumstances make building proper SLR search queries challenging task and a potential point of failure.

The challenges in SLR search have motivated research on how to automate query generation for SLR search. The existing solutions for building the SLR search query task mainly focused on automatically building search queries (e.g., [19,20]) or automatically refining existing search queries (e.g., [31,11]). These proposed methods, while valuable, are 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 to predict the performance and rank the generated queries. Moreover, these methods require authors to compose and provide an initial query, which poses challenges to researchers who have less knowledge about building search queries and especially who are new to a research topic [17].

This calls for solutions that can better adapt to different domains and SLRs while supporting SLR authors in the early the stage of review process where knowledge is still limited. We addressed these gaps by devising an adaptive query *building* and *refining* pipeline that relies on a reinforcement learning approach for incrementally refining a generated search query based on authors feedback. 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 that would impose on later screening steps. The rationale behind our approach is to leverage SLR authors' knowledge about the scope of the review and what is relevant, for automatically building and incrementally learning to refine the search queries. In sum, the contributions of this paper are as follows:

- We propose a method that exploits a high-level expression of the SLR scope to build initial search queries and semantically enrich them to deliver an applicable search query;
- 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.

The rest of this paper proceeds as follows. Related work is given in Section 2. Section 3 presents our proposed approach. The experiments and evaluations are presented in Section 4. And finally, Section 5 concludes the paper.

2 Related Work

While most of the literature on SLR automation focuses on study selection (see [6] for a review), interest in SLR search support has recently sparked. These efforts can be categorised into two main groups: i) automatic techniques for *generating* search queries, and ii) automatic *refining* of SLR search queries.

In the **search query generation task**, studies have leveraged information from the review protocol (e.g., review questions, inclusion and exclusion criteria) or a set of relevant abstracts to extract relevant keywords to build search queries [20,19]. These studies generally rely on text mining techniques (e.g., terms frequency-inverse document frequency (TF-IDF) [27]) to find the most relevant terms from a given corpus [19,20]. However, these approaches only focus on suggesting terms to help researchers building 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.

In the context of systematic reviews, **query refinement** is depicted as modifying a query to improve its recall or to reduce the number of studies retrieved [31]. Several studies explored techniques to achieve these goals [31,11]. By leveraging techniques in the form of query expansion (e.g., adding synonyms of the search terms), query reduction (e.g., removing unnecessary terms), or query transformation (e.g., rewriting query by replacing the Boolean operators).

Notably, Scells et al. in a series of studies [30,31] proposed a query refinement technique for medical SLR search queries comprised of query expansion (e.g., logical operator replacement (A AND B)$\rightarrow$(A OR B)) and semantic transformations (e.g., using medical embeddings)) and query reduction (e.g., removing unnecessary terms) and then automatically selecting the best query candidate.

While valuable, current query refinement techniques in SLRs have some limitations. These solutions require authors to compose an initial query which poses challenges to researchers who have less knowledge about building search queries and especially who are new to a research topic [17]. They also rely on machine learning techniques that require domain-specific training data, making them less reusable across domains. We note that the need for domain-specific data and training is a significant obstacle for the adoption of automation for SLRs [37].

The use of **human-machine approaches** has the potential to adapt to specific domains and incrementally improve outcomes by learning from human feedback. In SLRs, the active learning approach has shown significant success in reducing citation screening workload and cost [37,23]. Due to this success, some tools (e.g., Rayyan) have adopted this approach to support researchers in citation screening. However, the use of human-machine approaches for SLR query refinement remains unexplored.

In this paper we aim at filling the above gaps by designing an end-to-end pipeline that takes a seed expressing the scope of the review, to generate a Boolean search query and refine it following a human-machine approach. We take inspiration from previous research in automatic query expansion to generate a SLR search query from a seed (research questions or abstracts). We then propose a novel reinforcement learning method for refining SLR search queries, adopting reinforcement learning models to solve the query refinement as multi-armed bandit problem [34,38]. We contribute with empirical evidence characterising the performance of the query building and refinement components under various meaningful dimensions.

3 Incremental Query Building And Refining Pipeline

In our proposed query generation and refinement approach, the aim is to leverage minimum information available to the researchers on the scope of a review to build initial search queries. Then, through a refinement process, based on researcher feedback, incrementally refine and improve those initially generated queries. In the context of this work, the query quality associates with the performance of a query to i) retrieve relevant literature as defined by the scope of the review (recall), and ii) minimise the (unnecessary) screening effort that the number of studies in retrieved results imposes on the eligibility screening efforts. It is worth noting that this balance is important as very "open" search queries may be effective in retrieving the majority of relevant works but return a massive number of search results. Conversely, narrow search queries may be easily manageable but miss important relevant works.

To realise this goal, we devised the query building and refinement pipeline illustrated in Figure. 1. In summary, the pipeline receives a seed representing the scope of the SLR (e.g. research questions). Then, it uses the seed to extract candidate terms to build an initial query. The pipeline expands the initial query by enriching the terms in the initial query. The generated query is then automatically executed on a digital library (DL) search engine to retrieve the search results. The pipeline uses relevance feedback from researchers on the search results to measure the performance of the executed query. Finally, the pipeline uses these observations to refine the query using a reinforcement learning approach. In what follows, we elaborate on each component of our proposed pipeline.

3.1 Initial Query Builder

The first component of the pipeline is *Initial Query Builder* which leverages a high-level expression of the scope of an SLR to build an initial query. The input to this component is a *seed*, which can be partially defined research questions

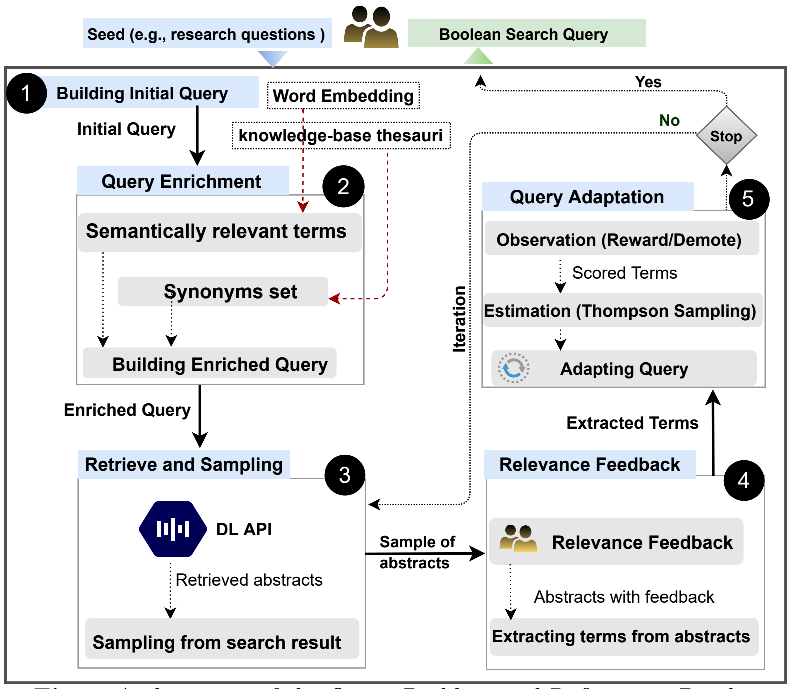


Fig. 1. Architecture of the Query Building and Refinement Pipeline

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 [18]. This component relies on the terms TF-IDF as a criterion to select terms to construct 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 terms frequency (TF) is used to select top-n relevant terms. The selected terms at this stage represent concepts that should be present in relevant literature (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 *'AND'* operator. For instance, for given research question *RQ: "Which techniques perform best when used to predict software fault?"* as the seed, *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 same concept [8]. To address this, we devise 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-base approach for finding the synonyms of main query terms [5], 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 [5].

Knowledge-base Enrichment. This component finds all the synonyms of a given query term using WordNet [22]. It builds a *synonym set* for each query term, containing the term and its synonyms. WordNet collects English words into groups of synonyms, called *synsets* [22]. One word in WordNet may have more than one synset. WordNet records the semantic relations between the synsets that describes the specific concept (hyponym) or generalized concept (hypernym) of a synset [22]. To enrich a given 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 that denotes the concept of "geography" and a synset that denotes "programming". The synset that denotes "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,...}.

The described enrichment approach has the potential to improve the query's retrieval performance [5]. However, the knowledge-based thesauri often do not hold all the semantically relevant terms that are not synonyms [21]. Therefore, we introduced another enrichment technique to further improve the query performance in retrieving papers that use alternative terms to express similar concept.

Embedding Enrichment. This enrichment component builds upon a word embedding approach [21]. In word embeddings, similar words have similar vectors in a vector space [21]. This component uses a word embeddings model to find the most relevant terms to 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 the terms within the synonym set ($\vec{set_s} = 1/|set_s| \sum_{s \in set_s} \vec{s}$). Herein, set_s denotes a synonym set. Next, the component uses the cosine similarity score of embedding terms and the mean vector of each synonym set ($\vec{set_s}$) to find 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, *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 instead of their synonym sets (set_s). This way, we ensured the terms that are more relevant to the query are ranked higher and selected for enrichment [15]. A query's mean vector is calculated by averaging the vectors of all the terms in the query ($\vec{q} = 1/|q| \sum_{t \in q} \vec{q}$). Herein, q denotes a query. Therefore, the score of term w from the candidate sets (W) is calculated as: $score(w, q) = cos(\vec{q}, \vec{w})$. Herein, $\vec{w}$ denotes vector of a term w in a candidate set (W). In our experiment, a minimum similarity threshold (α) is used to select 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, *Query Composer* component builds an enriched query by applying Boolean logic. It is assumed that the terms in the enriched sets are alternative to the *main terms* in the initial query. Therefore, *Query Composer* assembles the alternative terms in the enriched set using *OR* operator and forms a *query cluster* for each *enriched set*. For instance, the main term *predict* and its enriched set {predict, estimate, model} results in a cluster as: (predict OR detect OR model). The final query is then formed by concatenating all the query clusters, using an *'AND'* operator. Therefore, for initial query q, enriched query q^* will be generated: $q^* =$ (software OR program...) AND (fault OR defect OR...) AND (predict OR detect...). At this stage the queries are built using only OR and AND operation. Using other operations are left to the future work.

3.3 Query Adaptation

Query adaptation is a common practice in SLR search. Researchers modify search queries based on the knowledge gained by screening more candidate papers over time. In practice, this process can take many iterations and is prone to error. Furthermore, due to the large and expanding number of publications, researchers can spend significant time evaluating and appraising search results and accordingly adapting the queries [32]. To help researchers with 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 papers. We built our approach on a multi-armed bandit problems [14]. In multi-armed bandit problems, the algorithm continually allows a choice of which action to take and each action results in a reward based on an underlying probability distribution [33]. The algorithm learns to take the actions that maximise the accumulative rewards, by repeating the action and observing the results [33]. In what follows, we elaborate on the components of our proposed query adaptation approach.

Observation. The first step in the query adaptation is *observation*. It consists of three components: i) search and retrieve; ii) sampling; and iii) relevance feedback.

Search and Retrieve. This component facilitates the search on various digital libraries. It contains adapters for executing search queries by calling digital libraries API. It executes the queries and retrieves the results for processing.

Sampling. In practice, researchers examine the query performance by screening a subset of retrieved papers [16]. Building on this practice, we aim at improving the retrieval performance of the search query by learning about the relevance of a sample set from the query search results. For this, we introduced a *sampling* mechanism to the pipeline. The *sampling* component has the task of selecting S items from the search result based on a learning strategy introduced next. This component first extracts all the terms (nouns, verbs) from the retrieved abstracts using Stanford's CoreNLP library [18]. Next, it extracts the corresponding word vectors for these terms using an embedding model (e.g., Glove [24]). It then averages all these terms vectors to find 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 makes use of the cosine similarity metric to calculate the similarity between the vector of each abstract and the vector of the query: $Score(a, q*) = Sim_a = cos(\vec{q*}, \vec{a})$. Here, $\vec{a}$ represents the vector of the abstract and $\vec{q*}$ is the vector of the executed query. Then, it ranks the abstracts in descending order based on their similarity score. Finally, it 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 ($\eta < Sim_a < \zeta$). Instead, 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.

Relevance Feedback. This component presents the selected samples to researchers for receiving feedback. Leveraging a binary relevance feedback method [29], 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. 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.

Estimation. The next step in query adaptation process is *estimation* which includes two components: i) reward/demote schema and ii) terms performance estimator that computes the retrieval performance of the query terms.

Reward/Demote Scheme. This 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 (i.e., candidate terms). *Reward/Demote Scheme* complies with two main functions: i) Rewards/Demotes Calculator, and ii) Candidate Terms Finder.

Each time new relevant/irrelevant abstracts are received, the *Rewards/ Demotes calculator* extracts all terms (nouns, verbs) from each abstract, using Stanford's CoreNLP [18]. It then updates the rewards and demotes of the query terms, based on their presence in relevant or irrelevant abstracts. Each query term has default value of 1 for reward and demote in the first iteration. Each time a query term appears in a new irrelevant abstract, it will be demoted ($d_t = +1$). In turn, every time a query term appears in a new relevant abstract, it will be rewarded ($r_t = +1$). This component 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 $d_t = \sum_{i=1}^{n-1} d_{ti}$.

In addition to keeping the record of query terms' rewards and demotes, *Reward/Demote Scheme* has another function that finds 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 that meet these criteria are added to a *candidate terms* set. The candidate terms are 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 demote 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?". Instead of choosing an uncertain heuristic, we used Thompson Sampling [28], in which this question is answered by capturing this uncertainty in a probability distribution [28]. In our query adaptation problem, Thompson Sampling holds a *policy* for deciding which term should be modified in a query. It also provides an algorithm for finding new candidate terms that could update this policy [38]. We utilized Thompson Sampling for our multi-armed bandit problem because it has provided near-optimal regret[1] found in previous research [34].

For candidate term t, Thompson sampling estimates a probability distribution θ=$Beta(r_t, d_t)$, using the 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, which affect the action that is taken (removing or replacing a term) [38]. This distribution probability value is updated based on the algorithm observation (i.e., whether the term appears on relevant or irrelevant abstracts). Each time terms with rewards and demotes are received, the algorithm updates their θ value based on their rewards and demotes. More specifically, 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: $P(\theta|t) = \frac{P(t|\theta)P(\theta)}{P(t)}$. 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) satisfies: $argmax_t P(\theta|t) = E[reward|\theta_t]p(\theta_t|A)$ where A is the set of observed abstracts (relevant and irrelevant) 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 [4].

Query Modifier. The last component of query adaption is *Query Modifier* that modifies the query by adding or removing terms from the query based on their observed performance. To adapt a query, it first identifies query clauses[2] which their performance values are below a minimum threshold γ. The threshold outlines 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, this component finds the term in the clause that makes the clause inefficient. It compares the performance of the term with its sibling terms' performance[3]. *Query Modifier* would replace a term in the query clause if the number of relevant abstracts that the term appears on is

[1] The per-iteration regret is the mean of rewards of a choice with the best rewards and the action taken by the algorithm [28].

[2] We define a *query clause* as a conjunction of a set of terms such as (t_1 AND t_3)

[3] In a query cluster such as (t_1 OR t_2) , t_1 is considered as sibling term to t_2

below the average number of relevant abstracts its sibling terms appear on. This indicates that the term is not sufficient in retrieving relevant abstracts. Moreover, *Query Modifier* replaces this term with a *candidate term* which is estimated to yield the highest probability distribution (θ) by the Bayesian algorithm.

For instance, as illustrated in Figure 2, suppose that after i iterations *Query Modifier* identifies t_3 is to underperform. It would remove t_3 and replaces it with t_6, the candidate term that has the highest probability value θ in iteration i. When to stop the adaptation is a problem that could be addressed with different strategies. Currently we assume that the authors decide when to stop.

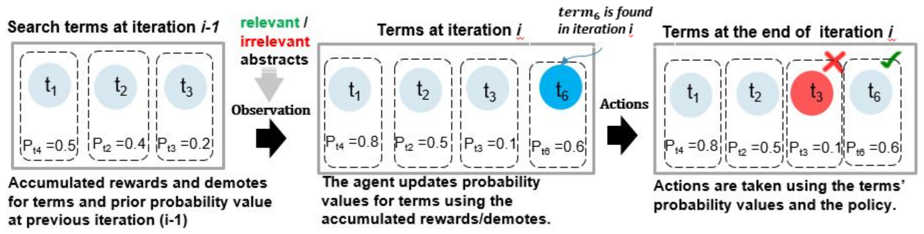

Search terms at iteration i-1
relevant / irrelevant abstracts
Observation
Terms at iteration i
term6 is found in iteration i
Actions
Terms at the end of iteration i
t1
t2
t3
t6
Pt4 =0.5
Pt2 =0.4
Pt3 =0.2
Pt4 =0.8
Pt2 =0.5
Pt3 =0.1
Pt6 =0.6
Accumulated rewards and demotes for terms and prior probability 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.

Fig. 2. Query modifier removes and adds terms in iteration i

4 Evaluation

The main goal of the evaluation was to assess the two main design decisions in our proposed pipeline: the (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 two experiments that evaluated the *initial* and *refined* queries according to relevant performance metrics.[4]

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 had published their search and screening data, by systematically searching datasets at Zenodo and Figshare.[5] As a result, we identified 10 SLRs that included research questions, the SLR search query, the search result dataset, and 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 label in our experiments.

Experiment 1- Initial query generation. In this experiment, we assessed our approach in generating the search query directly from an initial seed, by applying the query generation and enrichment components. To understand the impact of

[4] 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`

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

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).

Experiment 2- 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 author feedback by leveraging the relevance assessment already present in the SLR datasets, i.e., screening relevance feedback made strictly based on 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 a raw 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 papers 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 papers from each SLR dataset as the gold standard. However, since most SLR datasets include results from mul-

Table 1. Performance of generated queries based on different seed input, compared to manually formulated queries by SLR authors (#Ret= #Retrieved; #Rel=#Relevant).

Dataset	#Rel	BASE-OG			GEN-RQ			GEN-RQA			GEN-ABS-3		
		#Ret	Recall	Prec.	#Ret	Recall	Prec.	#Ret	Recall	Prec.	#Ret	Recall	Prec.
SLR_1[36]	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[9]	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[26]	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[10]	23	101	**0.95**	0.2178	260	0.83	0.0731	260	0.83	0.0731	146	0.87	0.1370
SLR_5[12]	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[3]	99	11,713	**0.93**	0.0079	350	0.80	0.2257	352	0.80	0.2244	220	0.59	0.2636
SLR_7[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[1]	19	1,652	**0.95**	0.0109	800	0.74	0.0175	800	0.74	0.0175	205	0.42	0.0390
SLR_9[25]	75	144	0.43	0.2222	730	**0.63**	0.0644	45	0.21	0.3556	79	0.16	0.1519
SLR_{10}[35]	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%	87.7%	121.9%	33.6%	72.8%	79.3%	24.1%	64.1%	66.3%

tiple 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

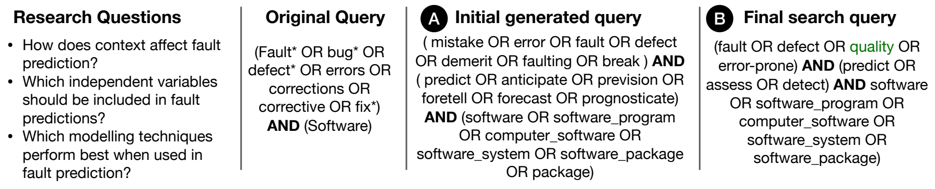
Experiment 1 The results of query generation are presented in Table 1[6], and the full details of the generated queries are available in the online Appendix.

In comparison to the baseline (BASE-OG), queries generated with the best variant of our approach (GEN-RQ), achieve 87.7% of the median recall, and 121.57% 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 identify 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 impact the quality of generated queries. For example, in our 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.

A closer look at the generated queries gives us hints into the characteristics of produced queries. As illustrated in Figure 3**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.

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



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)

Fig. 3. 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. 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 provides a solid base for composing and refining search queries.

Experiment 2 The results of the query adaptation and refinement step, for the uncertainty and certainty sampling, are shown in Table 2. 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, we can see that uncertainty sampling improves more slowly, or stalls in terms of 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 recall, and in particular avoiding missing out on relevant work, is paramount in SLRs, and our approach is able to build and refine queries towards this goal.

As illustrated in Figure 3, the refinement process is indeed able to reduce the non-relevant search terms (e.g., for "fault", removing "demerit") and add 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

Table 2. 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%

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 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 towards positive examples found to be the most effective. Overall, our empirical evaluation has shown the potential of the reinforcement learning approach to adapting search queries based on author feedback, without the need for domain-specific training.

While this approach and its evaluation showed some initial promise in generating search queries, there are limitations to the current pipeline and the evaluation methods. One limitation of our proposed enrichment approach is when the extracted terms from the seeds do not have many relevant terms in the embeddings. Therefore, selecting the most relevant terms becomes uncertain. Yet, replacing the general-embedding models with a domain-specific embeddings model could improve query enrichment process. The other limitation in our proposed approach is that pipeline only uses *AND* and *OR* operations for generating boolean queries, we leave exploring the impact of using NOT operator for generating and refining queries to future research. One limitation to our experiment is regarding taking the SLR datasets as gold-standard to measure the query performance. It is also required to measure the effectiveness of our proposed approach in building search queries that could retrieve relevant papers that are missing in the baseline SLR datasets (e.g., relevant papers that were excluded in published SLRs by mistake). This also requires further development in the pipeline and a scale user study. We leave this experiment for future research.

References

1. Adamo, G., Ghidini, C., Di Francescomarino, C.: What is a process model composed of? a systematic literature review of meta-models in bpm. arXiv preprint arXiv:2011.09177 (2020)
2. Badami, M., Baez, M., Zamanirad, S., et al.: On how cognitive computing will plan your next systematic review. arXiv preprint arXiv:2012.08178 (2020)
3. Barišić, A., Goulão, M., Amaral, V.: Domain-specific language domain analysis and evaluation: a systematic literature review. Faculdade de Ciencias e Technologia, Universidade Nova da Lisboa (2015)
4. Brochu, E., Cora, V.M., 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)
5. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. Acm Computing Surveys (CSUR) **44**(1), 1–50 (2012)
6. van Dinter, R., Tekinerdogan, B., Catal, C.: Automation of systematic literature reviews: A systematic literature review. Information & Soft. Tech. p. 106589 (2021)
7. Frank, M., Hilbrich, M., Lehrig, S., Becker, S.: 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). pp. 48–55. IEEE (2017)
8. Garousi, V., Felderer, M.: Experience-based guidelines for effective and efficient data extraction in systematic reviews in software engineering. In: Proc. of EASE'17. pp. 170–179 (2017)
9. Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S.: A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering **38**(6), 1276–1304 (2011)
10. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud migration research: a systematic review. IEEE transactions on cloud computing **1**(2), 142–157 (2013)
11. Kim, Y., Seo, J., Croft, W.B.: Automatic boolean query suggestion for professional search. In: Proceedings of SIGIR. pp. 825–834 (2011)
12. Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering–a systematic literature review. Information and software technology **51**(1), 7–15 (2009)
13. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering (2007)
14. Kohavi, R., Longbotham, R., Sommerfield, D., et al.: Controlled experiments on the web: survey and practical guide. DMKD **18**(1), 140–181 (2009)
15. Kuzi, S., Shtok, A., Kurland, O.: Query expansion using word embeddings. In: Proc. of CIKM. pp. 1929–1932 (2016)

16. Lee, G.E., Sun, A.: Seed-driven document ranking for systematic reviews in evidence-based medicine. In: SIGIR. pp. 455–464 (2018)
17. Li, H., Scells, H., Zuccon, G.: Systematic review automation tools for end-to-end query formulation. In: Proc. 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval. pp. 2141–2144 (2020)
18. Manning, C.D., Surdeanu, M., et al.: The stanford corenlp natural language processing toolkit. In: Proc. of ACL. pp. 55–60 (2014)
19. Marcos-Pablos, S., García-Peñalvo, F.J.: Decision support tools for slr search string construction. In: Proc. of TEEM'18. pp. 660–667 (2018)
20. Mergel, G.D., Silveira, M.S., da Silva, T.S.: 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. pp. 1594–1601 (2015)
21. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: NeurIPS. pp. 3111–3119 (2013)
22. Miller, G.A.: WordNet: An electronic lexical database. MIT press (1998)
23. Ouzzani, M., Hammady, H., Fedorowicz, Z., Elmagarmid, A.: Rayyan—a web and mobile app for systematic reviews. Systematic reviews **5**(1), 210 (2016)
24. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proc. of EMNLP. pp. 1532–1543 (2014)
25. Qin, C., Eichelberger, H., Schmid, K.: Enactment of adaptation in data stream processing with latency implications—a systematic literature review. Information and Software Technology **111**, 1–21 (2019)
26. Radjenović, D., Heričko, M., Torkar, R., Živkovič, A.: Software fault prediction metrics: A systematic literature review. Information and software technology **55**(8), 1397–1418 (2013)
27. Robertson, S.: Understanding inverse document frequency: on theoretical arguments for idf. Journal of documentation (2004)
28. Russo, D., Van Roy, B., Kazerouni, A., et al.: A tutorial on thompson sampling. arXiv preprint arXiv:1707.02038 (2017)
29. Salton, G., Buckley, C.: Improving retrieval performance by relevance feedback. Journal of the American society for information science **41**(4), 288–297 (1990)
30. Scells, H., Zuccon, G.: Generating better queries for systematic reviews. In: ACM SIGIR. pp. 475–484 (2018)
31. Scells, H., Zuccon, G., Koopman, B.: Automatic boolean query refinement for systematic review literature search. In: WWW. pp. 1646–1656 (2019)
32. Scells, H., Zuccon, G., Koopman, B.: A comparison of automatic boolean query formulation for systematic reviews. Information Retrieval Journal **24**(1), 3–28 (2021)
33. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT (2018)
34. Tabebordbar, A., Beheshti, A., Benatallah, B., et al.: Feature-based and adaptive rule adaptation in dynamic environments. DSE **5**(3), 207–223 (2020)
35. Teixeira, E.N., Aleixo, F.A., de Sousa Amâncio, F.D., OliveiraJr, E., Kulesza, U., Werner, C.: Software process line as an approach to support software process reuse: A systematic literature review. Information and Software Technology **116**, 106175 (2019)
36. Wahono, R.S.: A systematic literature review of software defect prediction. Journal of Software Engineering **1**(1), 1–16 (2015)
37. Wallace, B.C., Small, K., Brodley, C.E., et al.: Who should label what? instance allocation in multiple expert active learning. In: SDM. pp. 176–187. SIAM (2011)
38. Williams, J.J., Kim, J., Rafferty, A., et al.: Axis: Generating explanations at scale with learnersourcing and machine learning. In: L@Scale. pp. 379–388 (2016)