A Machine Learning Approach for Semi-Automated Search and Selection in Literature Studies

Rasmus Ros
rasmus.ros@cs.lth.se
Lund University
Department of Computer Science
Sweden

Elizabeth Bjarnason
elizabeth.bjarnason@cs.lth.se
Lund University
Department of Computer Science
Sweden

Per Runeson
per.runeson@cs.lth.se
Lund University
Department of Computer Science
Sweden

ABSTRACT

Background. Search and selection of primary studies in Systematic Literature Reviews (SLR) is labour intensive, and hard to replicate and update. **Aims.** We explore a machine learning approach to support semi-automated search and selection in SLRs to address these weaknesses. **Method.** We 1) train a classifier on an initial set of papers, 2) extend this set of papers by automated search and snowballing, 3) have the researcher validate the top paper, selected by the classifier, and 4) update the set of papers and iterate the process until a stopping criterion is met. **Results.** We demonstrate with a proof-of-concept tool that the proposed automated search and selection approach generates valid search strings and that the performance for subsets of primary studies can reduce the manual work by half. **Conclusions.** The approach is promising and the demonstrated advantages include cost savings and replicability. The next steps include further tool development and evaluate the approach on a complete SLR.

CCS CONCEPTS

•**General and reference** → **Surveys and overviews;** Empirical studies; •**Computing methodologies** → *Semi-supervised learning settings;*

KEYWORDS

Systematic literature review, Research identification, Study selection, Automation, Machine learning, Reinforcement learning

ACM Reference format:
Rasmus Ros, Elizabeth Bjarnason, and Per Runeson. 2017. A Machine Learning Approach for Semi-Automated Search and Selection in Literature Studies. In *Proceedings of EASE'17, Karlskrona, Sweden, June 15-16, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3084226.3084243

1 INTRODUCTION

Systematic literature reviews (SLR) have become very popular in software engineering (SE) as a means to synthesize existing knowledge in a growing field of research. The growing number of SLRs have even called for SLRs of SLRs [10, 15, 18], or tertiary studies. Kitchenham et al. conclude that the most common criticisms of SLRs is "that they take a long time, that SE digital libraries are not appropriate for broad literature searches and that assessing the quality of empirical studies of different types is difficult." [15] Improvement approaches are proposed to mitigate the problems, including the use of 'quasi-gold standard' to iteratively improve the search string [11, 36]. Another issue is the lack of replicability, and as a consequence, the problem of keeping SLRs up-to-date as new research is published [35].

One of the key underlying problems is that the terminology in software engineering is not well established. Being a systems science, elements from several fields of research are integrated into solutions to software engineering problems. Thus, terms may vary between studies due to the field the research is based on. For example, not even the definition of the term 'experiment' is consistently used [11]. Further, SE research outcomes may be published not just in SE venues, but also in venues related to other fields of research, for research where SE is applied, e.g. the mining software repositories community.

To address these issues, we propose an approach to search and selection in SLRs that is based on machine learning technology. While current processes and tool support for SLR consider search and selection as two distinct steps, we propose an integrated approach to search and selection, supported by machine learning. The first step is to train a classifier on a starting set of validated papers. Secondly, this set of papers is extended by snowballing or by broad automated searching, and the results are filtered through the classifier. Thirdly, the reviewer is asked for input on validation of the classified papers, which is iteratively integrated with the classifier. Thus the only manual work in search and selection is to interactively train the classifier by accepting or rejecting papers suggested by the classifier.

We investigate machine learning techniques that are suitable to use in an iterative setting, such as active learning [29] with support vector machines (SVM) [8] and reinforcement learning with logistic regression [6]. In addition we show how to infer search strings by learning a decision tree [26].

We argue that the proposed approach supports the SLR process by reducing the manual effort needed and enabling wider searches through machine classification. Further, the use of machine learning tools may—if they are made openly available together with their parameters—standardise the SLR process and thus enable replication, and reduce the space needed to report study methodology. As a consequence, updating SLR search may be done continuously, by rerunning the search and selection tool [34]. The saved effort on search and selection may instead be spent on systematically

EASE'17, Karlskrona, Sweden

978-1-4503-4804-1/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3084226.3084243

synthesising the knowledge in the SLR; a research effort that is much needed as the amount of SE research increases [9]. The price to pay is a less transparent search process (even though steps are taken to mitigate this threat), and the detailed evaluation of the approach remains for future research.

We present related work on tools and methodology for SLRs in Section 2. In Section 3 we present the proposed approach in detail, and Section 4 presents a proof-of-concept example and an empirical evaluation of the proof-of-concept tool. Sections 5 and 6 conclude the paper with a discussion and an outline of further work.

2 RELATED WORK

While there is much research on conducting SLRs and on tools to support this, we focus here on the work on which we base our contribution. The process that we are proposing builds on and combines ideas from several areas. We use standard machine learning methods, e.g. logistic regression, decision trees, and cross-validation [22]. In addition, we use standard natural language processing techniques for creating a training set for classification, such as stemming, n-grams, TF-IDF, and topic modelling [3].

2.1 General tool support for SLRs

There are several tools that support the whole SLR process, such as enabling multiple reviewers to work together, providing templates for writing protocols, and information extraction for synthesis. In contrast, our work is focused solely on the search and selection process. Medicine researchers Tsafnat et al. [33] performed a SLR on automation for SLRs, they describe in detail the potential for automation on the different parts of the SLR process. Our key take away is that an integration of the tools that solve the different parts of the SLR process may lead to a changed work flow, and our contribution is an attempt to move in that direction.

There are also tools that support federated search, where search from multiple search engines is combined in a single tool. Ghafari et al. [13] show that federated search reduces the workload and that it can facilitate integrated duplication removal. We believe federated search is an integral feature for a search tool.

2.2 Automated selection

In a mapping study and a feature analysis on SLR tools, Marshall et al. [19, 20] found that existing tools have poor support for automated selection. Automated study selection by machine learning tools was also addressed in an SLR by medicine researchers O'Mara-Eves et al. [25]. They find that tool support for study selection can be used for prioritising which papers to validate, and that automated study selection may be used with low loss of recall. They call for further work by other research disciplines to validate and improve the tool support for automated selection.

The SE researchers Olorisade et al. performed a re-review of O'Mara-Eves et al.'s SLR to further investigate the text mining techniques used. They claim that there is insufficient evidence of automatic primary study selection working in practice, and that researchers have insufficient insight into how the tools they are using works [24]. We deduce that it is important for tools to support the SLR process in a transparent way and agree that future empirical evidence is needed.

2.3 Refining search strings

The literature on SE SLR methodology suggests using a set of initial papers to refine the search strings, a quasi-gold standard (QGS) set [16, 36]. This is a known starting set of papers that are expected to be captured in a search string. The search string is refined to be as precise as possible while still capturing a sufficient number of papers from the QGS set. The method has been improved to use a method that splits the QGS into two parts, one for validation and one for refinement [17]. This prevents over specialising to the papers included in the QGS. This process is similar to the automatic version we present in Section 3.3. To the best of our knowledge there has been no previous research on generating search strings automatically with data-driven methods and using them in SLRs.

2.4 Snowballing

Related studies can be found through snowballing based on references between papers, either backwards or forwards. Backwards snowballing follows references from a targeted paper. Forwards snowballing uses an indexing service like Scopus or Google scholar to find papers that have cited the target paper. Based on our review, snowballing appears to be common in the SE community. Compared to term-based search, there is some evidence that it performs similarly regarding recall, and some researchers even report improved precision [2, 14, 31].

Badampudi et al. [2] advise caution in selecting a good start set, since it is required to reach sufficient recall. Medicine researchers have found that citation graphs of papers are often disconnected [27], and also advise use of a good start set when snowballing. We recommend combining snowballing with our automated term based search, which automatically extends the start set.

There is existing tool support for automated forwards snowballing by Choong et al. [7], which has the benefit of facilitating automatic updates of review studies as new research is published.

2.5 Active learning and reinforcement learning

Active learning is a type of semi-supervised machine learning that reduces the labour of manual classification needed to construct a training set [29]. The core idea behind active learning is that the classifier queries an external oracle for help, i.e. a human. Designing an active learning solution includes a number of options, such as choice of classification algorithm, a stopping criterion, and a selection scheme from which the learner selects papers. The selection criterion is based on which example will give the most information to the learner, regardless of whether the example is positive or negative.

Learning in iterations maps well to study selection and has been found to reduce workload [21, 34], because the selection process can be stopped before all papers are validated. Unbalanced training sets can pose a problem for machine learning in general, where the number of positive examples (included papers) are much smaller than the number of negative examples (excluded papers). In one of the studies the authors simply weighed positive examples higher [21].

Reinforcement learning is gaining traction in the machine learning community and we believe it may be an efficient approach for primary study selection in SLRs. It is the underlying algorithm for many state-of-the art content recommendation engines [30],

which is superficially similar to the problem at hand. For automated study selection it enables the same work flow as an active learning solution. The key difference to active learning is that reinforcement learning strikes an optimal balance between selecting the best examples and exploring the search space. To the best of our knowledge there is no published work that uses reinforcement learning for SLRs.

3 APPROACH

We propose a method for automating the search and selection phases of the SLR process, which correspond to the 'identify research' and 'select studies' in terms of the guidelines by Kitchenham et al. (see [16] Chapters 5–6). 'Identify research' refers to the method by which papers are collected and 'select studies' is the method by which relevant papers are chosen. The other phases of the SLR process remain unchanged, e.g. creating a protocol and synthesis of the accepted papers. As part of the proof-of-concept we implement a tool prototype to show that the approach is feasible.

The ideal tool for study search and selection would be one that provides paper recommendations with the only manual task required is for the researcher to perform the validations of papers suggested by the tool. This requires a fully automated research identification and a semi-automated selection. We argue that the separation of search and selection is artificial, and that this work aims to bridge the gap between the two where the search is transparently handled by the tool. The steps in the process will be broadly explained in Section 3.1, and further details follow in the succeeding sections.

Section 3 is focused on the *conceptual tool*, Section 3.1 gives an overview of the process, and the other sections discusses design choices for the steps in the process.

3.1 Process for semi-automated search and selection

The method we propose for identifying research and selecting studies is a five step process, see Figure 1. The steps are as follows: 1) perform automated search using automatically calculated search strings, 2) perform broad automated snowballing of papers previously accepted into the review, 3) remove duplicates of newly found papers, 4) rank all non-validated identified papers using a classifier with fine granularity, and finally 5) present the highest ranked paper to the reviewer for validation, this validation is added to the training set for the classifier. This process is the repeated iteratively until the stopping criterion is satisfied. The input to the process is a starting set of accepted and rejected papers that are added to the training set, and the output from the process is the set of papers in the training set that are accepted by the reviewer.

The key to automation is machine learning, using statistical inference to learn what is a relevant paper. The overall goal is to reduce the workload of the search and selection process and also expand the scope of search. The machine learning algorithms are expected to be used interactively and iteratively.

3.1.1 Step 1. Automated search. We propose to use a data-driven approach to generate search string using terms from title, abstract and keyword of the currently accepted papers. This data set is the same training set that is used for automated ranking. The search

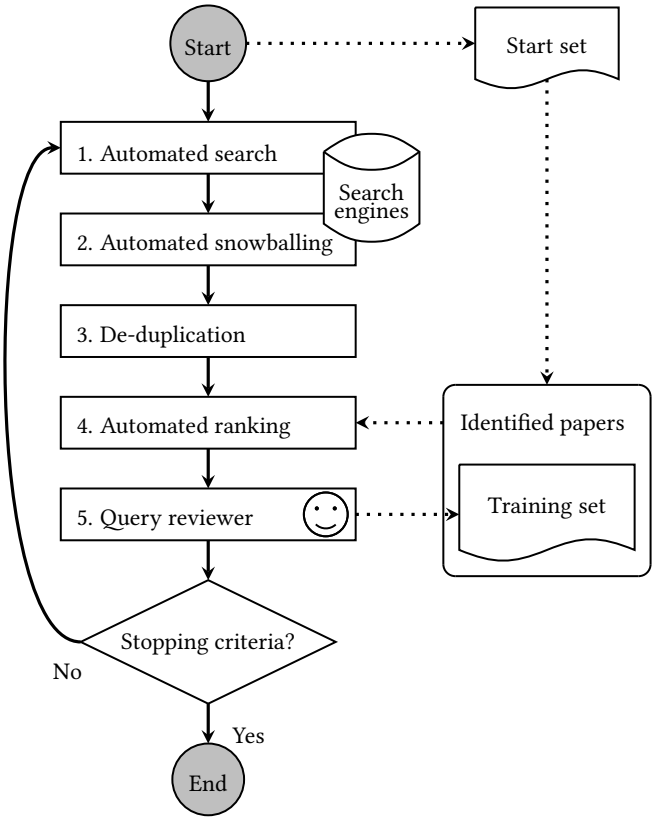


Figure 1: Overview of the process for semi-automated study selection and research identification. The smiley show where a human is directly involved.

strings are executed against the search engines' programmatic interface and the meta-data are automatically downloaded. Normally a strict order of search first and snowballing after is performed. This is not necessary for our computer-supported approach with an efficient performance of these tasks, including switching between them. Thus the conceptual tool can interleave snowballing and term based search. Automated search is in theory the same as automated selection, but there are big technical differences in that search operates towards a search engine and selection towards the downloaded paper meta-data.

A weakness of the traditional manual approach is that the search string is kept fixed once the selection phase is started. Any new knowledge obtained during selection that could have been applied to search thus cannot be used in an efficient way, because it would invalidate some of the previous selections. The method we are suggesting for automated search is data-driven and produces continuously refined search strings, the extra work is trivial for a computer to perform.

The tool implementation should use several search engines and combine the result, i.e. federated search [13]. Since the validation of this study is a proof-of-concept we are content to validate that it is possible using only the Scopus search engine.

3.1.2 Step 2. Automated snowballing. Performing automated snowballing seems like a good target for automation, but it requires using information extraction to extract references from a paper. Scopus also has tool support for backwards snowballing, where they provide a list of used references for a paper. This list of references is not currently delivered as structured data, so there is still a reliance on information extraction.

Forward snowballing is hard to perform automatically due to the reliance on an index service. There are two major indexing alternatives that can be used as a service. The first is Google Scholar which does not allow tool access to its service, and the other is Scopus, which does allow tool access, but does not have an ambition to cover all literature.

Due to these technical difficulties none of the snowballing alternatives have been implemented in the proof-of-concept tool in this study. There is however sufficient evidence that the approach works [7].

3.1.3 Step 3. De-duplication. During the research identification there will be many duplicated papers found, because term based search is performed multiple times and combined with snowballing. This means duplicated papers must be removed automatically. Since full meta-data is retrieved by the search engines this can be done easily with the digital object identifier (DOI), which is a globally unique string for each paper. Not all papers have a DOI though, in which case de-duplication is done by matching title and authors with heuristics, or other unique identifiers.

Study de-duplication is not included in the approach because there are no standard ways of reporting study meta data in SE, which means it is hard to automatically perform.

3.1.4 Step 4. Automated ranking. Study selection is performed on the papers found previously. We are proposing a semi-automated study selection where each paper is ranked according to the classifier. The papers recommended to be included in the review are then iteratively presented to the reviewer. The classifier is a supervised machine learning algorithm which ranks papers according to the probability of it being included. The ranking order of the papers depends on the details of the underlying algorithm. All papers currently validated by the reviewer are used as training data for the classifier.

The tool should be interactive and the trained model should be interpretable so that it can be manually inspected. Suitable algorithms for study selection are then, e.g. decision trees, logistic regression, and naïve Bayes. If the requirements on efficiency and interpretability are relaxed, then state-of-the art methods like random forest, SVM, or artificial neural networks could be used.

The machine learning algorithms require a processing step of the data into structured numeric form. Each paper is transferred into a feature vector, which is a numeric description of the attributes constituting the paper. The procedure for forming feature vectors is as follows. Each word (as n-grams) in the abstract, title, and keywords are joined together to form a single unordered set, which is converted to a vector by indexing. This is known as a n-gram bag-of-words model, the reason for choosing n-grams is that it can accurately encode technical terms, which are given additional meaning when written in order. For example "things" and "Internet" in isolation is very different from "Internet of Things". The procedure follows a standard pipeline for natural language processing with: 1) tokenising, 2) stemming, 3) removing stop words, 4) constructing n-grams, 5) removing rare terms. We suggest using term frequency-inverse document frequency (TF-IDF) [28], using IDF from the set of all currently downloaded papers, as is commonly used in information extraction. Thus, information from the non-validated papers can be used as well, in the form of term counts.

The training set can be extended with a topic model, where similar papers are clustered together with an unsupervised learning algorithm, such as latent Dirichlet allocation (LDA) [4]. Each feature vector in the training set is combined with the feature vector that describes the topic cluster. This usually improves the classification accuracy by providing additional information to the classifier.

Due to technical reasons inherent to the search engines available, the training set can not contain the full text of the papers, and is instead restricted to title, abstract, and keywords. This is not necessarily a problem: Dieste et al. [11] showed for manual search strings, that using only title, abstract, and keywords gave better precision while sacrificing little recall compared to using full text term searches. Whether this result is valid for automated generation of search strings is left for future research.

3.1.5 Step 5. Query reviewer. The tool should have an integrated user interface that supports validation by the reviewer. The bare minimal requirements on the interface is to present title, authors, abstract, and keywords. There should be a supporting motivation for why the paper was suggested, such as what the most relevant terms were that were used by the classifier, and information on how the paper was found, i.e. via snowballing or search. Additionally the tool can provide full-text links and other time saving features for the reviewer.

The machine learning used must be fast to train and evaluate so that the reviewer is presented with a new paper as soon as a validation is completed.

3.1.6 Stopping criteria. As always when obtaining data sets for machine learning there are diminishing returns for obtaining more data. For the purpose of being systematic, it is possible to devise a quantifiable stopping criterion which signals a discrete point where the cost of obtaining data is higher than the benefit from receiving it is. In a sense, a stopping criterion is always devised, where in a standard manual review the stopping criterion is to stop once all papers returned in the search are validated. The automated process should not be stopped if there are still papers that are classified as accepted by the classifier. Once the process is stopped the remaining non-validated papers are regarded as automatically filtered.

There are several suggestions for stopping criteria in active learning [29]. The general strategy is to measure the confidence of the classifier and stop the active learning once the uncertainty drops below a certain threshold. Similar constructions could be created for reinforcement learning. Since all of these measures are oversimplifications, we propose using simple and general stopping criteria. For example, when no more positive examples are found by the classifier, there are no accepted papers in the previous X classifications, or X classifications have been performed. These simple alternatives are preferable because they are easy to report and understand.

There should also be decision support in the form of statistics and graphs of the classifier. Learning curves are well suited to this, i.e. a plotted curve of accuracy to the number of classified examples, or the number of accepted paper to the number of validations, as seen in Figure 4 in Section 4.2.

3.2 Data provenance and transparency

Conducting a trustworthy review requires all parts of the literature review to be as transparent as possible. As there appears to be a potential clash with using a machine learning tool—which can be a black box—and transparency, we are taking four steps to mitigate this risk.

Firstly all resulting models should be interpretable, to enable manual inspection of the learned model. This way the reviewer using the tool can see that the model does not overfit to frivolous details, e.g., a specific author's commonly used words.

Secondly all actions taken by the tool and the reviewer should be logged in a human readable format and could therefore in theory be manually replicated. The log should be made publicly available by the reviewer. From the log a trace of the full data provenance of all papers can be obtained, data provenance is the lineage or history of the data. It should be clear how a paper was identified, by which search string and from which search engine, which queries were run by the tool to a search engine, what papers were presented to the reviewer, and what the classification was.

Thirdly the tool should report relevant statistics and metrics relating to both the optimization method (such as mean squared error) of the machine learning algorithm for classification and general metrics (such as accuracy) of the learned model.

Finally we recommend making the classified data set publicly available so that it is explicit not only which papers were accepted in the review, but also which were rejected. This has the additional benefit that anyone can continuously keep the review up to date by using the tool.

3.3 Inferring search strings for automated search

Here we describe a baseline method which infers search strings fully automated, using previously validated examples. This search string is continuously updated when new examples are obtained. A search string can be formalised as a predicate first order logic rule which evaluates to true for papers that should be returned in a query. First order logic can be learned from data, but it is a challenging combinatorial optimisation problem [1]. Since the search string will be fully recomputed iteratively, the algorithm to infer search strings from papers must be fast. Because the end-result must be a search string we can only use n-gram features during training.

A decision tree can be calculated efficiently with an entropy based greedy heuristic as in the ID3 algorithm [26]. The tree can be linearised into a set of first order logic rules by following every valid path from the root of the tree to a leaf with positive classification. Thus every path in the decision tree forms one search string. These search strings could be joined with an OR operator if desired, but they are guaranteed to be disjoint so they can be executed and downloaded in parallel. See Figure 2 for an example of the process; this example comes from a SLR on continuous experimentation,

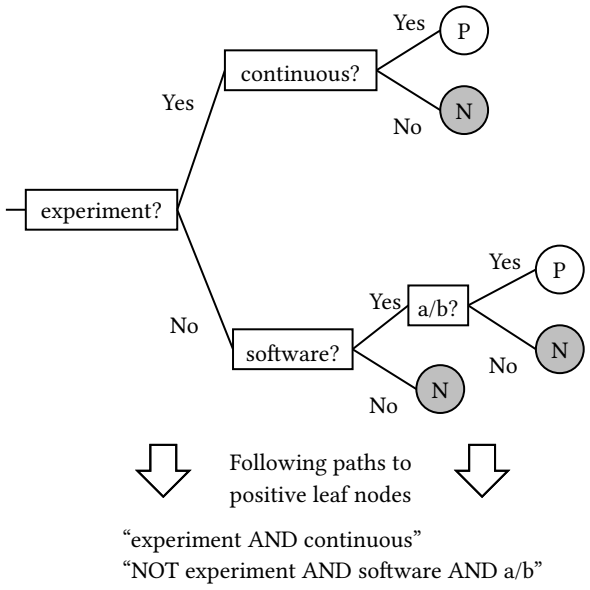


Figure 2: Converting a learned decision tree to search strings.

as will be discussed in Section 4. Since there are two leaf nodes with a positive classification, there are two disjoint search strings generated. The decision tree is a much simplified version of the final result.

Since the actual data set used for training is a small subset of the whole universe of papers we use as precise search strings as possible, so the decision tree is not pruned for increased recall. At the start of the process the decision tree will be simple, thus the search strings will be broad and will capture a lot of irrelevant papers. When the irrelevant papers are validated by the reviewer the search strings are refined for higher precision.

The queries that are generated can return excessive amounts of search hits, especially at the beginning of the SLR process. We use lazy fetching where the set of papers is expanded as needed, to reduce the traffic to the search engines. Scopus and other search engines expose a paging functionality that can be used iteratively to fetch a specific part of the search results. The tool should keep the set of non-validated papers to, e.g. at least 1000 papers, and there should be at least 10 non-validated papers that are classified as accepted. If any of these conditions are not met then additional results are fetched from the search engines using the sub-queries that have not been expanded fully so far. It is necessary to keep the set of non-validated papers high to support the more refined semi-automated selection algorithm.

3.4 Semi-automated selection with reinforcement learning

The classifier used for semi-automated selection should produce a ranking that is more granular than the binary ranking produced by the decision tree for automated searching, since the ranking is performed on the downloaded meta-data and not on the search

engine itself. The trained classifier will rank papers according to the probability that the paper is accepted.

It is necessary to explore papers outside of what the learned model currently believes is the best papers in order not to get stuck on a local optima. Current state-of-the-art solutions to semi-automated study selection use active learning, as described earlier, to focus on the papers that the classifier find difficult. Our suggestion is instead to use reinforcement learning, which optimally suggest the best papers while still exploring the search space, this is known as the explore/exploit trade off. Our hypothesis is that it will lead to a more efficient selection process.

One way to attack this problem is by defining a probability distribution which describes the uncertainty on all the variables that form a learned model. The probability distribution has an initial high variance which represents high uncertainty, as the evidence for the different variables increase the variance will decrease. This probability distribution is used for inference on which non-validated examples are fruitful to explore. Picking the best example involves integrating out all uncertainty of a joint probability distribution to arrive at a discrete decision, but this is intractable in an interactive setting, even for the modest data set sizes of the problem for semi-automated selection. Instead we use a simple heuristic, probability matching, which enjoys good formal motivations. The idea is to simply pick values at random from the probability distribution and the integration will occur over time instead [30].

We use a modified version of a simple algorithm with underlying ridge logistic regression, described in algorithm 3 by Chapelle and Li [6]. In their model the distribution is placed on the weights of the regression model and are approximated by a Gaussian distribution with a diagonal covariance matrix, using a Laplace approximation. The mean of this distribution is the actual weight value and the variance is inverse proportional to the number of observations of the weight in the feature vectors. There are two advantages from this model, it can be solved using off-the shelf logistic regression and it is trivial to take samples from this distribution which can be used for probability matching.

Applying the model directly on the feature weights of all terms in the training set would also result in too much exploration, since the training set is sparse and most n-grams have few observations. For this reason we split the classifier in a topic part and a term part, which are trained separately, and apply the reinforcement learning to the topic part only. The result is combined before applying the logit link function, by simply adding the regression equations. See the final results in algorithm 1 for details.

In summary the paper recommended to the reviewer is that which is predicted to be the most relevant by the classifier, while factoring in the uncertainty of the prediction.

Input: Regularization parameters λ and γ.
Terms x_i, $i = 1, \ldots, d$.
Papers with terms $\mathbf{x}_j$, $j = 1, \ldots, n$.
Initial classifications y_j.
Topic model over topics z_l, $l = 1, \ldots, k$.

for $t = 1, \ldots, T$ **do**
  for $j = 1, \ldots, n$ **do**
    Calculate $\mathbf{z}_j$ from the topic model.
  end
  Learn the logistic regression weights:

$$\mathbf{m} = \arg\min_{\mathbf{m}} \frac{1}{2} \sum_{l=1}^{k} q_l m_l^2 + \sum_{j=1}^{n} \log(1 + \exp(-y_j \mathbf{m}^{\mathrm{T}} \mathbf{z}_j)),$$

$$\text{where } q_l = \gamma + \sum_{j=1}^{n} z_{lj}^2 p_j (1 - p_j), pj = (1 + \exp(-\mathbf{m}^{\mathrm{T}} \mathbf{z}_j))^{-1}.$$

$$\mathbf{w} = \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{n} \lambda w_i^2 + \sum_{j=1}^{n} \log(1 + \exp(-y_j \mathbf{w}^{\mathrm{T}} \mathbf{x}_j)).$$

  Draw $\tilde{\mathbf{m}}$ from each independent $\mathcal{N}(m_l, q_l^{-1})$.
  Select the best paper according to:
  $\mathbf{x}_{jt} = \arg\max_{\mathbf{x}_j} \mathbf{w}^{\mathrm{T}} \mathbf{x}_j + \tilde{\mathbf{m}}^{\mathrm{T}} \mathbf{z}_j$.
  Receive y_{jt} from human validation and update $\mathbf{y}$.
end

Algorithm 1: Logistic regression with reinforcement learning based on the topics of papers.

3.5 Evaluating search performance

Good recall is crucial for the findings of the review to be correct. Recall is a commonly used metric for search completeness [16]; it is the fraction of relevant items retrieved. Precision can be used to evaluate how efficient a search method is and is defined as the fraction of retrieved items that are relevant. An estimate of the true recall can get an indication of whether there are any missing papers from the final review.

One way to get a recall estimate is by having a quasi-gold set separate from the starting set of papers and comparing the search result to the quasi-gold set [17]. This type of evaluation has high rigidity since it is external to the conceptual tool, but the sample size will most likely be low.

K-fold cross-validation is common in the machine learning literature to evaluate the performance of a learned model for both recall and precision. It would be workload expensive to use in a manual search performance evaluation, but since we are using automated inference to compute search strings and perform snowballing we can evaluate with k-fold cross-validation. The training of the decision tree is repeated k times, each time holding one different k-th part of the training set and using it as test set. The evaluation will see which parts of the test set is reached by the computed search strings and by automated snowballing. By evaluating in this way it is not necessary to have a separate test set, which could otherwise negatively impact the performance of the learned model, due to decreased training set size.

4 EVALUATION

The presented approach has been validated by implementing a proof-of-concept tool that supports the novel functionality, i.e. the automated search and the machine learning for semi-automated selection.

The data set for evaluation is from an ongoing SLR on continuous experimentation. The full data set contains 1939 papers, of which 108 are included in the review and 1831 are excluded, these papers

form the training set for the classifier. The data set was created with manual backwards and forwards snowballing. By having the data set be created with snowballing we can do an evaluation of the search performance of the automated search, without a bias that would be present if the data set was constructed with term based search. Only 419 of the papers in the data set have abstracts, which hurts the performance of the classifier.

A short explanation of the different learning algorithms and settings are given in Section 4.1, the evaluation of search and selection is described in Section 4.2 and Section 4.3 respectively.

4.1 Implementation of learning algorithms and settings

We have tried multiple supervised learning algorithms, and tested them in different semi-supervised learning settings, such as active learning. This section presents brief details on the implementations, and references to the standalone software packages we have used.

Several of the algorithms have hyper-parameters, meaning that they are free design parameters that are not automatically optimised by the algorithm. We find these hyper-parameters by using a semi-exhaustive grid search, and evaluation of the harmonic mean of precision and recall (F_1-score) found by 10 fold cross-validation. Grid search is a common technique and a reasonable first step for this proof-of-concept tool. During the simulation of different learning settings we search for hyper-parameters after every 100th example is added.

For the LDA topic model we use the implementation of jLDADMM [23]. It is necessary to chose the number of topics to use; we use a common heuristic of $\sqrt{n}$, where n is the number of papers available. The prior for the Dirichlet distribution is set to 1 (for a uniform probability of each term in each topic).

4.1.1 *Learning algorithms.*

SVM is support vector machines, using the LibSVM tool for implementation [5]. The best kernel was found to be linear, we also tried sigmoid, polynomial, and gaussian. Thus, for a linear kernel, mis-classification cost is the only hyper-parameter.

LR is logistic regression, using the Liblinear tool for implementation [12]. Regularization constant is the only hyper-parameter. We tried ridge and lasso regularization and found ridge (or L_2) regularization to work best.

LR-T learns models from terms and topics separately, this adds a regularization constant as another hyper-parameter.

DT infers a decision tree with information gain as the splitting decision [26], it does not have any pruning of the decision tree and it uses only term features.

4.1.2 *Learning settings.*

Greedy always selects the paper with the highest probability of being accepted.

AL is active learning using uncertainty sampling as heuristic, it picks the paper where the classifier is the most undecided. For SVM it selects the feature vector which is closest to the separating hyperplane, for logistic regression it selects $\mathbf{x}_j = \arg\min_{\mathbf{x}_j} |\mathbf{w}^T \mathbf{x}_j|$, i.e. the vector which has the regression equation closest to zero.

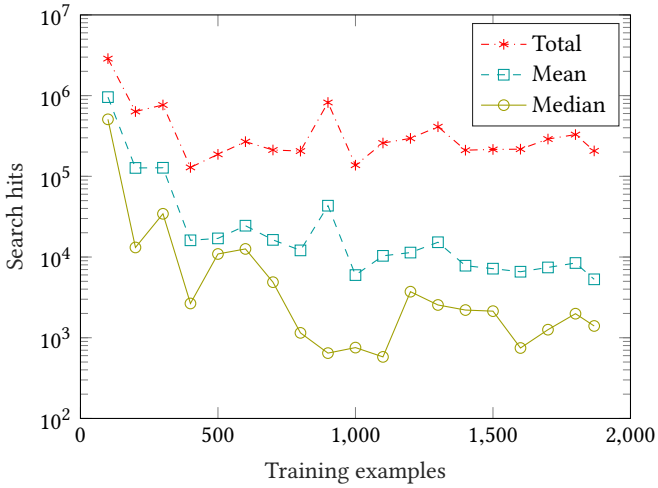


Figure 3: Evaluation of automated search, with the total number of search hits and the mean and median of the sub-queries. The differences in mean and median indicate outliers.

AL-W is a modified version of **AL**, which weighs examples classified as positive by the classifier higher. The weight is the ratio between negative and positive examples in the training set [21].

RL is the reinforcement learning algorithm with Thompson sampling and logistic regression, as described in algorithm 1 in Section 3.4.

4.2 Evaluation of automated search

The algorithm outlined in Section 3.3 has been evaluated with the training set of SLR papers on continuous experimentation. The decision tree is iteratively trained, and the search strings are generated and evaluated for how many search hits are returned. The training set is randomly extended in batches of size 100 containing both positive and negative examples. The number of search hits from each sub-query is recorded and then totalled. There are no duplicates since the sub-queries are disjoint. The cross-validation performance of the decision tree applied to automated selection instead can be seen in Table 2.

The results are displayed in Figure 3, which shows the number of validated training examples to the number of search hits. The mean and median are of each sub-query, which starts at 3 for 100 examples and ends in 31 for 1939 examples. The number of sub-queries increase linearly with a slope of 1/60, meaning every 60 validations the decision tree increases in complexity. The final number of search hits are 205, 013; it is possible to combine the final search string with a manual filter of, e.g. "software engineering", doing so brings the total down to 24, 699. The longest sub-query is only 10 terms long, which should present no problems for any search engine.

As can be seen from the large difference between mean and median, the distribution is heavily skewed by outliers that return many search hits. All iterations of the decision tree are pure during

Table 1: Evaluation of ranking performance. Comparing different versions of semi-automated selection to manual work, using an oracle stopping criterion; averages of 200 simulation runs.

Setting	Algorithm	Stop at	Reduction
Greedy	LR	1212	37.5%
Greedy	SVM	1227	36.7%
AL	SVM	1398	27.9%
AL-W	SVM	1215	37.3%
RL	LR	1625	16.2%
RL	LR-T	**843**	**56.5%**

training, meaning that precision and recall are both 100% when evaluated on the training set. This is normally an indication of overfitting, when compared to the values obtained during cross-validation in Table 2. This is working as intended, even though the mean and median have a clear trend of decreasing as the number of validated examples increase, unfortunately the total search hits do not decrease. When inspecting the sub-queries it is apparent that for some sub-queries the model have selected general broad terms that by random chance are not well represented in the training data; these sub-queries generate a large number of search hits. We believe this can be solved by adding a global language model of the n-gram probabilities of each term in scientific language and incorporating this in the decision tree splitting criteria. This language model can be pre-calculated and packaged with the tool.

4.3 Evaluation of semi-automated selection

The process for semi-automated selection is evaluated in terms of the amount of work saved compared to performing a fully manual selection. This is defined as the stopping point in the process based on an oracle stopping criterion (which knows when there are no more papers to accept). The ranking power of the algorithms is evaluated in a simulation where it is started with a randomly selected accepted and a rejected paper, and it uses a batched iterative learning setting where 10 additional papers are picked and validated for each iteration. This validation is performed automatically based on the previous human validation of the 1939 papers. The simulation is repeated 200 times and the results are averaged to get a less noisy result. All evaluated algorithms and learning settings are presented in Table 1, where they are compared to a fully manual selection that validates all papers. We have tested all possible combinations of the settings and algorithms explained here. In this paper, we include only viable results, or where the method has been used in a previous study. We see that the performance of active learning does not outperform the greedy baseline, and that the naïve version of reinforcement learning is exceptionally poor, as hypothesised in Section 3.4.

The top 3 versions (Greedy LR, AL-W SVM, and RL-T LR) from Table 1 are also evaluated in further detail. A graph showing the number of found positive examples to the number of training examples are shown together with a histogram of where the process was stopped during the 200 runs, see Figure 4. The variance on when the stopping criterion is met for Greedy LR, AL-W SVM, and RL LR-T is 125, 80, and 108 respectively. Low variance might be an advantage of the active learning approach, while the greedy version has high variance.

Table 2: Evaluation of classification performance. Results are obtained with 10-fold cross-validation performance of whole training set.

Algorithm	Accuracy	Precision	Recall	F_1-score
LR	**93.8**%	**45.7%**	63.9%	**0.533**
LR-T	92.8%	41.1%	**66.7%**	0.509
SVM	93.7%	44.8%	60.2%	0.514
DT	92.1%	36.7%	56.5%	0.445
Coin-flip	54.5%	5.6%	50.0%	0.100

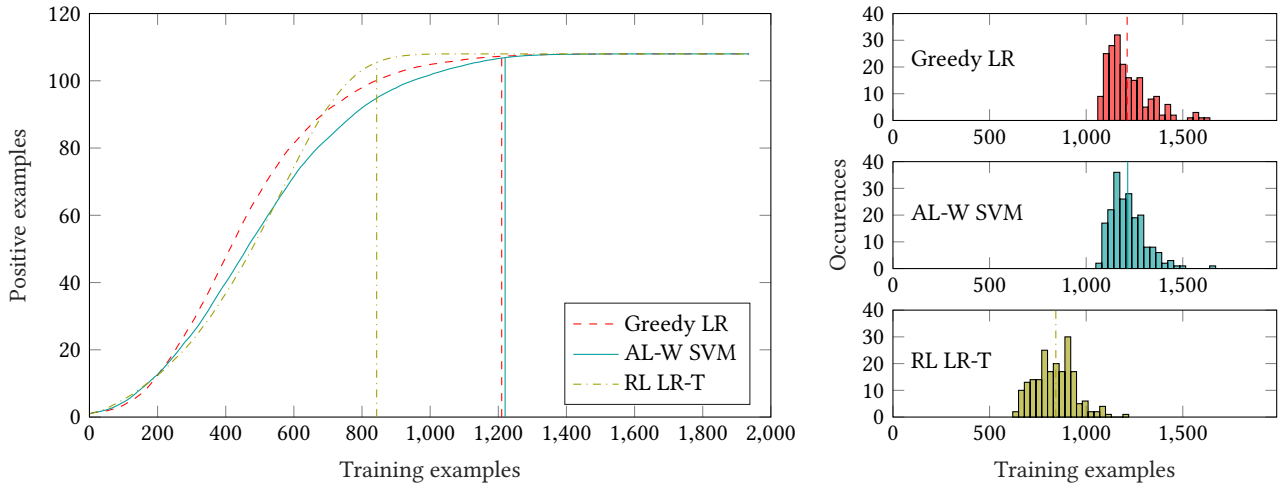
We also test the final classification performance of the algorithms using 10-fold cross-validation evaluation on the data set with all 1939 papers validated. This is done by setting a fixed 50% cut-off for all rankings, where every paper above this threshold is included. Thus the true recall of the process used in a ranking setting will be much higher, since the cut-off is continuously decreased until the process is stopped. In the data set there are several papers by some authors and we take care to put all articles of a first author in the same fold, otherwise there could be data leakage from one fold to another. The accuracy, precision, and recall of the different algorithms are presented in Table 2. The decision tree used for automated search strings inference is also included there.

The results in Table 2 are of secondary importance to the results in Table 1, since the purpose of the algorithmically enhanced selection is to reduce work load and not to obtain a good classifier. The cross-validation performance of the algorithms have low precision, when taken out of context this is alarming. But compared to a coin-flip it is clear that precision is good, since there is a high imbalance in the training set, the rejected examples outnumber the accepted by 17 to 1. As mentioned earlier the recall of the algorithms is not necessarily crucial compared to precision.

RL with LR-T clearly outperforms the other alternatives in Table 1, and LR is the marginal winner in Table 2. The curve in Figure 4 shows that the method outperforms the other in terms of reaching the goal of all 108 accepted papers first, even if it starts out slow. It is also clear that active learning does not improve on the baseline of greedy selection, even with the modification of selection to favour higher ranked examples. The numbers are produced with an oracle stopping criterion, meaning that the process stops as soon as all accepted papers are found. For a more realistic number, we can change the stopping criterion to: the process stops after 50 papers are rejected in a row, which brings the reduction of manual work down from 57% to 54%.

5 DISCUSSION

We have shown by a proof-of-concept that the approach we present is sound and that it can reduce the labour needed to produce a SLR. In our initial evaluation the reinforcement learning algorithm we propose reduces the manual work necessary for selection by half. It is clear from the evaluation that active learning is not well suited to automated selection, as it does not outperform the baseline of



(a) Number of positive examples found to the number of training examples validated, the lines are averaged over 200 simulations. A fully manual process would have a linear progression from origo to the upper right corner.

(b) Histograms of number of manual validations needed until the stopping criterion is met. This gives an indication of the variance of the methods.

Figure 4: Comparing different versions of semi-automated selection to manual work, using an oracle stopping criterion. The results were obtained by bootstrapping a simulation 200 times, based on one fully manual validation of all 1939 papers. The vertical lines represent the average number of validations until the stopping criterion is met.

greedy selection. The results from automated search are inconclusive, and while the total search hits seem excessive, downloading and classifying 24, 699 or even 205, 013 abstracts is not a challenging task for a computer. There are many suggestions for improving those results, but even without these improvements the proposed automated process is an improvement on the current manual process. Whether the approach leads to an overall improved quality of SLRs remains for further evaluation. We argue that by reducing the work load through a semi-automated approach, the quality of the research will likely increase since a more efficient search and selection can lead to more time spent on the SLR synthesis phase instead, and we hypothesize that it will lead to better recall.

Some improvements that might improve automated search and selection respectively follows. The search could be extended with a pre-calculated n-gram language model to reduce the irrelevant search hits. The machine learning model for selection could be improved further by: testing and evaluating more advanced supervised learning algorithms, adding more features from meta-data, testing different feature selection options, and by using a topic model that does not have the need to select the number of topics, such as hierarchical Dirichlet processes [32]. The reinforcement learning can be improved by using a hierarchical model that combines the information from the topic clusters with terms.

There is always a danger of not finding all relevant parts of the literature, e.g. by creating search strings that are too narrow or with snowballing by having a starting set that is too small. This risk is present in all sorts of literature reviews. Just as there might be different research communities that might not cite each other which creates difficulties for snowballing, there might be different terminology that creates difficulties for term based searching. The risk is mitigated in our approach by performing multiple types of research identification that complement each other, and by having search strings that are iteratively refined. We demonstrate that the proposed approach works even on a new research topic, where the terms are not well established and where there are conflicts with other terms, e.g. in the case of continuous experimentation the conflict is with experimentation as a research method.

The evaluation performed is an internal threat to validity in this study, since the data set is not based on a completed SLR, and the evaluation is performed on search and selection separately. There are two reasons for not comparing to other completed SLRs, firstly both accepted and rejected papers are needed, secondly search based SLRs cannot be used since the terms used to generate the papers are normally strong predictors, but they would be present in all papers and so would invalidate the evaluation. However, by using the completed tool we could quickly redo previously published SLRs and compare recall.

There is a potential objection on the systematic properties of the approach, there are several steps in the proposed model that include randomness, such as the algorithms used for reinforcement learning and the LDA topic model. We believe this is a small price to pay for the advantages offered by an efficient automated process.

6 CONCLUSION

We propose an integrated process for semi-automated search and selection in SLRs based on an iterative approach to machine learning, using reinforcement learning. The approach has been validated by implementing a proof-of-concept tool and its performance measured on a data set from a partially completed mapping study. In our initial evaluation the semi-automated primary studies selection reduces the necessary work by half and the automated search is demonstrated to be feasible. We demonstrate that the method is

useful for mapping studies, and derive that it can also be applied for systematic reviews, which usually have a narrower focus in a mature field of study.

The five main benefits of an integrated solution of semi-automated search and selection are that, 1) reviewers do not need to construct search strings, 2) as a consequence reviews can have better recall when it is not dependent on the recall of an initial search string, 3) the set of included papers can be updated automatically by the tool once the classifier is sufficiently trained, 4) the approach can be implemented with an efficient interface that the reviewer can use until the search and selection is done, reducing cognitive load, and finally 5) the process is well suited for complex reviews where search strings are hard to create. Further tool development is necessary to fully support the proposed automated process.

We believe that the approach can lead to reviews of higher quality if the saved time is spent on other phases of the review. However future research is required to evaluate this work by performing complete SLRs using the proposed approach.

ACKNOWLEDGMENT

This work was conducted within the Wallenberg Autonomous Systems and Software Program (WASP) (http://wasp-sweden.se).

REFERENCES

[1] S. Augier, G. Venturini, and Y. Kodratoff. 1995. Learning first order logic rules with a genetic algorithm. In *Proc. of The 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)*.

[2] D. Badampudi, C. Wohlin, and K. Petersen. 2015. Experiences from Using Snowballing and Database Searches in Systematic Literature Studies. In *Proc. of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE '15)*. ACM, New York, NY, USA, Article 17, 10 pages. DOI: http://dx.doi.org/10.1145/2745802.2745818

[3] S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Inc.

[4] D. M. Blei, A. Y. Ng, and J. I. Michael. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.

[5] C.-C. Chang and C.-J. Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011), 27:1–27:27. Issue 3.

[6] O. Chapelle and L. Li. 2011. An Empirical Evaluation of Thompson Sampling. In *Proc. of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*. 2249–2257.

[7] M. K. Choong, F. Galgani, A. G. Dunn, and G. Tsafnat. 2014. Automatic Evidence Retrieval for Systematic Reviews. *Journal of Medical Internet Research* 10, e223 (Oct 2014). DOI: http://dx.doi.org/10.2196/jmir.3369

[8] C. Cortes and V. Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297. DOI: http://dx.doi.org/10.1007/BF00994018

[9] D. S. Cruzes and T. Dybå. 2011. Research Synthesis in Software Engineering: A Tertiary Study. *Information and Software Technology* 53, 5 (2011), 440–455. DOI: http://dx.doi.org/10.1016/j.infsof.2011.01.004

[10] F. Q. B. da Silva, A. L. M. Santos, S. Soares, A. C França, C. V. F. Monteiro, and F. F. Maciel. 2011. Six Years of Systematic Literature Reviews in Software Engineering: An Updated Tertiary Study. *Information and Software Technology* 53, 9 (2011), 899–913. DOI: http://dx.doi.org/10.1016/j.infsof.2011.04.004

[11] O. Dieste, A. Grimán, and N. Juristo. 2009. Developing Search Strategies for Detecting Relevant Experiments. *Empirical Software Engineering* 14, 5 (2009), 513–539. DOI: http://dx.doi.org/10.1007/s10664-008-9091-7

[12] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.

[13] M. Ghafari, M. Saleh, and T. Ebrahimi. 2012. A Federated Search Approach to Facilitate Systematic Literature Review in Software Engineering. *International Journal of Software Engineering & Applications* 3, 2 (2012), 13–24. DOI: http://dx.doi.org/10.5121/ijsea.2012.3202

[14] S. Jalali and C. Wohlin. 2012. Systematic Literature Studies: Database Searches vs. Backward Snowballing. In *Proc. of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. 29–38. DOI: http://dx.doi.org/10.1145/2372251.2372257

[15] B. A. Kitchenham and P. Brereton. 2013. A Systematic Review of Systematic Review Process Research in Software Engineering. *Information and Software Technology* 55, 12 (2013), 2049–2075. DOI: http://dx.doi.org/10.1016/j.infsof.2013.07.010

[16] B. A. Kitchenham, D. Budgen, and P. Brereton. 2015. *Evidence-Based Software Engineering and Systematic Reviews.* CRC Press.

[17] B. A. Kitchenham, Z. Li, and A. Burn. 2011. Validating Search Processes in Systematic Literature Reviews. In *Proc. of the 1st International Workshop on Evidential Assessment of Software Technologies*.

[18] B. A. Kitchenham, R. Pretorius, D. Budgen, P. Brereton, M. Turner, M. Niazi, and S. Linkman. 2010. Systematic Literature Reviews in Software Engineering – A Tertiary Study. *Information and Software Technology* 52, 8 (2010), 792–805. DOI: http://dx.doi.org/10.1016/j.infsof.2010.03.006

[19] C. Marshall and P. Brereton. 2013. Tools to Support Systematic Literature Reviews in Software Engineering: A Mapping Study. In *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 296–299. DOI: http://dx.doi.org/10.1109/ESEM.2013.32

[20] C. Marshall, P. Brereton, and B. A. Kitchenham. 2014. Tools to Support Systematic Reviews in Software Engineering: A Feature Analysis. In *Proc. of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 13. DOI: http://dx.doi.org/10.1145/2601248.2601270

[21] M. Miwa, J. Thomas, A. OfiMara-Eves, and S. Ananiadou. 2014. Reducing Systematic Review Workload Through Certainty-based Screening. *Journal of Biomedical Informatics* 51 (2014), 242–253. DOI: http://dx.doi.org/10.1016/j.jbi.2014.06.005

[22] K. P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective.* The MIT Press.

[23] D. Q. Nguyen. 2015. jLDADMM: A Java Package for the LDA and DMM Topic Models. http://jldadmm.sourceforge.net/. (2015).

[24] B. K. Olorisade, E. de Quincey, P. Brereton, and P. Andras. 2016. A Critical Analysis of Studies That Address the Use of Text Mining for Citation Screening in Systematic Reviews. In *Proc. of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16)*. ACM, 14:1–14:11. DOI: http://dx.doi.org/10.1145/2915970.2915982

[25] A. O'Mara-Eves, , J. Thomas, J. McNaught, M. Miwa, and S. Ananiadou. 2015. Using Text Mining for Study Identification in Systematic Reviews: A Systematic Review of Current Approaches. *Systematic Reviews* 4, 1 (2015), 5. DOI: http://dx.doi.org/10.1186/2046-4053-4-5

[26] J. R. Quinlan. 1986. Induction of Decision Trees. *Machine Learning* 1, 1 (1986), 81–106.

[27] K. A. Robinson, A. G. Dunn, G. Tsafnat, and P. Glasziou. 2014. Citation Networks of Related Trials are Often Disconnected: Implications for Bidirectional Citation Searches. *Journal of Clinical Epidemiology* 67, 7 (2014), 793 – 799. DOI: http://dx.doi.org/10.1016/j.jclinepi.2013.11.015

[28] G. Salton, E. A. Fox, and H. Wu. 1983. Extended Boolean Information Retrieval. *Communication of the ACM* 26, 11 (1983), 1022–1036. DOI: http://dx.doi.org/10.1145/182.358466

[29] B. Settles. 2009. *Active Learning Literature Survey.* Computer Sciences Technical Report 1648. University of Wisconsin–Madison. 11 pages. https://minds.wisconsin.edu/handle/1793/60660.

[30] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. of the IEEE* 104, 1 (2016), 148–175. DOI: http://dx.doi.org/10.1109/CC.2014.7085614

[31] M. Skoglund and P. Runeson. 2009. Reference-based Search Strategies in Systematic Reviews. In *Proc. of the 13th international conference on Evaluation and Assessment in Software Engineering (EASE'09)*. 31–40.

[32] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet Processes. *J. Amer. Statist. Assoc.* 101, 476 (2006), 1566–1581. DOI: http://dx.doi.org/10.1198/016214506000000302

[33] G. Tsafnat, P. Glasziou, M. K. Choong, A. Dunn, F. Galgani, and E. Coiera. 2014. Systematic Review Automation Technologies. *Systematic Reviews* 3, 1 (2014), 74. DOI: http://dx.doi.org/10.1186/2046-4053-3-74

[34] B. C. Wallace, K. Small, C. E. Brodley, J. Lau, C. H. Schmid, L. Bertram, C. M. Lill, J. T. Cohen, and T. A. Trikalinos. 2012. Toward Modernizing the Systematic Review Pipeline in Genetics: Efficient Updating via Data Mining. *Genetics in Medicine* 14, 7 (2012), 663–669. DOI: http://dx.doi.org/10.1038/gim.2012.7

[35] C. Wohlin, P. Runeson, P. A. da Mota Silveira, E. Engström, I. do Carmo Machado, and E. S. de Almeida. 2013. On the Reliability of Mapping Studies in Software Engineering. *Journal of Systems and Software* 86, 10 (2013), 2594–2610. DOI: http://dx.doi.org/10.1016/j.jss.2013.04.076

[36] H. Zhang, M. A. Babar, and P. Tell. 2011. Identifying Relevant Studies in Software Engineering. *Information and Software Technology* 53, 6 (2011), 625–637. DOI: http://dx.doi.org/10.1016/j.infsof.2010.12.010