Empir Software Eng (2018) 23:3161–3186
https://doi.org/10.1007/s10664-017-9587-0

Finding better active learners for faster literature reviews

Zhe Yu[1] · **Nicholas A. Kraft**[2] · **Tim Menzies**[1]

Published online: 7 March 2018

Abstract Literature reviews can be time-consuming and tedious to complete. By cataloging and refactoring three state-of-the-art active learning techniques from evidence-based medicine and legal electronic discovery, this paper finds and implements FASTREAD, a faster technique for studying a large corpus of documents, combining and parametrizing the most efficient active learning algorithms. This paper assesses FASTREAD using datasets generated from existing SE literature reviews (Hall, Wahono, Radjenović, Kitchenham et al.). Compared to manual methods, FASTREAD lets researchers find 95% relevant studies after reviewing an order of magnitude fewer papers. Compared to other state-of-the-art automatic methods, FASTREAD reviews 20–50% fewer studies while finding same number of relevant primary studies in a systematic literature review.

Keywords Active learning · Systematic literature review · Software engineering · Primary study selection

Communicated by: Per Runeson

✉ Tim Menzies
tim.menzies@gmail.com

Zhe Yu
zyu9@ncsu.edu

Nicholas A. Kraft
nicholas.a.kraft@us.abb.com

[1] Department of Computer Science, North Carolina State University, Raleigh, NC, USA

[2] ABB Corporate Research, Raleigh, NC, USA

1 Introduction

When conducting a literature review in software engineering, it is common practice (Kitchenham and Brereton 2013) to conduct a *primary study selection* where a large number of potentially relevant papers, collected via some initial query (e.g. keyword search in Google Scholar), are manually reviewed and assessed for relevance. To reduce the effort associated with conducting such tedious and time-consuming *linear manual reviews*, researchers in the fields of evidence-based medicine (Paynter et al. 2016; Wallace et al. 2010a, b) and electronic discovery (Cormack and Grossman 2014, 2015) have developed *active learning* methods that can build an automatic classifier that prunes away irrelevant papers using feedback from users.

In this paper we investigate whether there are any insights from that related work that can reduce the effort associated with literature reviews in software engineering (SE). Specifically, in this paper we:

- Review those active learning methods (Paynter et al. 2016; Wallace et al. 2010a, b; Cormack and Grossman 2014, 2015) and find that in evidence-based medicine and legal electronic discovery, there are three widely recognized state-of-the-art active learning methods (Cormack and Grossman 2014; Wallace et al. 2010b; Miwa et al. 2014).
- Analyze those three active learning methods and find that they are each assembled from lower-level techniques that address four questions: (1) when to start training, (2) which study to query next, (3) whether to stop training, and (4) how to balance the training data.
- Investigate 32 possible active learning approaches that represent different combinations of lower-level techniques to address the four questions.
- Evaluate the 32 active learning approaches using SE data and the evaluation criteria "work saved over sampling at 95% recall" (WSS@95) (Cohen 2011).
- Discover that one of those 32 active learning approaches, which we call FASTREAD, reduces the effort required to find relevant papers the most.

Based on that work, the contributions and outcomes of this paper are:

1. A cautionary tale that verbatim reuse of data mining methods from other fields may not produce the best results for SE. Specifically, we show that supposed state-of-the-art methods from other fields do not work best on SE data.
2. A case study showing the value of refactoring and recombining data mining methods. The FASTREAD tool recommended by this paper was constructed via such refactoring.
3. A demonstration that FASTREAD is a new highwater mark in reducing the effort associated with primary study selection in SE literature reviews.
4. A open source workbench that allows for the fast evaluation of FASTREAD, or any other technology assisted reading method. See https://github.com/fastread/src.
5. Four new data sets that enable extensive evaluation of FASTREAD or other methods. The creation and distribution of these data sets is an important contribution, because prior to this study, it was very difficult to obtain even one such data set.

The rest of this paper offers background notes on the problem of reading technical documents and on how that problem has been solved in other fields. We then refactor those solution into 32 candidate solutions, which we asses using prominent published SE literature reviews. Using that data, we ask and answer the following three research questions:

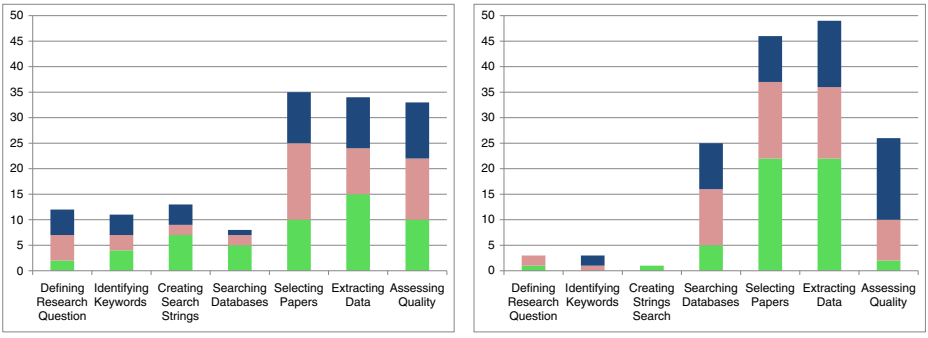
- **RQ1: Can active learning techniques reduce effort in primary study selection?** We find that using FASTREAD, after reviewing a few hundred papers, it is possible to find 95% of the relevant papers found by a linear manual review of thousands of papers.
- **RQ2: Should we just adopt the state-of-the-art treatments from other fields?** Our results show that better active learners for SE can be build by mixing and matching methods from the state-of-the-art in other fields.
- **RQ3: How much effort can FASTREAD, our new state-of-the-art method for primary study selection, save in an SLR?** We show that FASTREAD can reduce more than 40% of the effort associated with the primary selection study phase of a literature review while retrieving 95% of the relevant studies.

2 Background

Systematic Literature Reviews (SLRs) are a well established and widely applied review method in Software Engineering since Kitchenham, Dybå, and Jørgensen first adopted it to support evidence-based software engineering in 2004 and 2005 (Kitchenham et al. 2004; Dyba et al. 2005). Researchers can get a general idea of current activity in their field of interests by reading the SLR studies. Furthermore, a deeper understanding of the topic may be gained by conducting an SLR.

An increasing number of SLRs has been conducted since the proposal and revision of the SLR guidelines in 2007 (Keele 2007). For example, there were 26 SLRs on IEEE Xplore during the year of 2005 and that number has increased to 137, 199 for the years 2010, 2015 (respectively). Various scholars suggest that an SLR is required before any research in Software Engineering is conducted (Keele 2007). While this is certainly a good advice, currently an SLR is a large, time consuming and complex task (Hassler et al. 2016, 2014; Carver et al. 2013; Bowes et al. 2012).

Cost reduction in SLRs is therefore an important topic and will benefit researchers in software engineering community. Previously we have analyzed the costs of SLRs (Hassler et al. 2014; Carver et al. 2013). As shown in Fig. 1, primary study selection, which is noted as "selecting papers" in Fig. 1, is among the top three most difficult as well as time-consuming



(a) Most Difficult Aspects of SLR Process (b) Most Time Consuming Aspects of SLR Process

Fig. 1 Data collected from surveys to SLR authors (Carver et al. 2013). Green, red, and blue show the most, second most, and third most, respectively voted item

aspects in an SLR. Usually, reviewers need to evaluate thousands of studies trying to find dozens of them that are relevant to the research questions based on their title, abstract, or full text (Bowes et al. 2012). An extreme example of this is where reviewers sourced over 3000 studies, and only used 7 of them in their final review (Bezerra et al. 2009). In terms of actual cost, Malheiros has documented that it requires 3 h for one reviewer to review 100 studies (Malheiros et al. 2007). This implies that it is a month's work for one graduate student to review 3000 studies or three months' work to review 9000 studies. The cost associated with primary study selection has become a serious problem and will continue to grow in the near future as the population of candidates for primary studies increases dramatically. In this paper, we focus on reducing cost in primary study selection only. Our prioritization on the cost reductions of primary study selection is not to discount the effort associated with other parts of the SLR process. Indeed, one of our considerations is that there are already tools to support other parts of the SLR process, as well as different techniques to facilitate primary study selection. All these techniques (such as Quasi-Gold Standard based search (Zhang et al. 2011a, b), visual text mining (Felizardo et al. 2010, 2012, 2014; Malheiros et al. 2007), and snowballing (Wohlin 2014; Jalali and Wohlin 2012)) are compatible with works in this paper and a better performance is expected when applied together. This leads to a direction of future work in which the best setting to integrate different techniques will be explored.

There are three main aspects in primary study selection: **(1)** retrieving initial list of primary studies, **(2)** excluding irrelevant studies, **(3)** including missing studies. We focus on excluding irrelevant studies because **a)** there already exists techniques and tools to facilitate **(1)** and **(3)** such as Snowballing (Jalali and Wohlin 2012) and StArt (Hernandes et al. 2012); **b)** the performance of excluding irrelevant studies can be evaluated using existing SLR publications.

2.1 Related Work

2.1.1 Software Engineering Tools

In recent years, various tools have been developed to facilitate SLRs in the software engineering community, as summarized in the associated SLRs (Marshall et al. 2014, 2015; Marshall and Brereton 2013). These tools aim at providing support for protocol development (Molléri and Benitti 2015; Fernández-Sáez et al. 2010; Hernandes et al. 2012), automated search (Molléri and Benitti 2015; Hernandes et al. 2012), primary study selection (Molléri and Benitti 2015; Hernandes et al. 2012; Fernández-Sáez et al. 2010; Bowes et al. 2012), quality assessment (Fernández-Sáez et al. 2010; Bowes et al. 2012; Molléri and Benitti 2015), data extraction and validation (Molléri and Benitti 2015; Hernandes et al. 2012; Fernández-Sáez et al. 2010; Bowes et al. 2012), data synthesis (Molléri and Benitti 2015; Hernandes et al. 2012; Fernández-Sáez et al. 2010; Bowes et al. 2012), and report write up (Molléri and Benitti 2015; Hernandes et al. 2012; Fernández-Sáez et al. 2010; Bowes et al. 2012). It is extremely helpful to have a tool for managing the whole SLR process. However, the support for primary study selection using these tools is limited (e.g., to tasks such as assigning review jobs to multiple reviewers or to resolving disagreements). Hence, we planned to introduce machine learning to assist primary study selection in SE SLRs but before this paper is published, Ros et al. (2017) has achieved this in June 2017. While Ros'17 (Ros et al. 2017) provided a wide range of techniques to support both search

and selection, it has several limitations such as (a) not comparing against state-of-the-art techniques from other domains (which are approaches discussed later in Sections 2.1.2 and 2.1.3); (b) not considering any data balancing; (c) testing only on a single unpublished dataset.

Visual text mining (VTM) is a technique especially explored in Software Engineering community to support SLR. It is an unsupervised learning method which visualizes the relationship between candidate studies and helps the reviewer to make quick decisions. Malheiros et al. (2007) first applied VTM to support primary study selection in SLR. In their small-scale experiment (100 candidate studies, 31 of which are "relevant"), VTM retrieves around 90% of the "relevant" studies by spending about 30% as much time as manual review. However, VTM requires some prior experience and knowledge of text mining and visualization techniques to use (Bowes et al. 2012), and more case studies with large scale are needed to validate their results.

Snowballing is another technique attracting much attention in SE SLR research. Given the inherent relevance relationship between a study and its citations, it is of high probability for the citations of (used in backward snowballing) and the studies cite (used in forward snowballing) a known "relevant" study to also be "relevant" (Kitchenham et al. 2004). Jalali and Wohlin (Jalali and Wohlin 2012; Wohlin 2014) applied backward snowballing to search for primary studies in SE SLRs and found comparably good result as database search. Felizardo et al. (2016) and Wohlin (2016) applied forward snowballing to update SE SLRs and greatly reduced the number studies need to be reviewed comparing to a database search. This paper does not use snowballing since, as mentioned by Wohlin (2014), snowballing starts with an initial set of relevant papers. FASTREAD's task is very different: we start with zero relevant papers.

2.1.2 Legal Electronic Discovery Tools

Electronic Discovery (e-discovery) is a part of civil litigation where one party (the producing party), offers up materials which are pertinent to a legal case (Krishna et al. 2016). This involves a review task where the producing party need to retrieve every "relevant" document in their possession and turn them over to the requesting party. It is extremely important to reduce the review cost in e-discovery since in a common case, the producing party will need to retrieve thousands of "relevant" documents among millions of candidates. Technology-assisted review (TAR) is the technique to facilitate the review process. The objective of TAR is to find as many of the "relevant" documents in a collection as possible, with reasonable cost (Grossman and Cormack 2013). Various machine learning algorithms have been studied in TAR. So far, in every controlled studies, continuous active learning (Cormack'14) has outperformed others (Cormack and Grossman 2014, 2015), which makes it the state-of-the-art method in legal electronic discovery. It has also been selected as a baseline method in the total recall track of TREC 2015 (Roegiest et al. 2015). Details on continuous active learning are provided in Section 3.

2.1.3 Evidence-Based Medicine Tools

Systematic literature reviews were first adopted from evidence-based medicine in 2004 (Kitchenham et al. 2004). To facilitate citation screening (primary study selection) in

systematic review, many groups of researchers have investigated different types of machine learning algorithms and evaluation mechanisms (O'Mara-Eves et al. 2015; Paynter et al. 2016).

Cohen et al. first applied text mining techniques to support citation screening and developed several performance metrics (including WSS@95) for assessing the performance of different techniques in 2006 (Cohen et al. 2006). While the great contribution of introducing machine learning and text mining into citation screening as well as the proposed performance metrics of Cohen has been widely acknowledged (O'Mara-Eves et al. 2015), most of Cohen's work focused on supervised learning which does not utilize unlabeled data and relies on random sampling to obtain the sufficiently large training set (Cohen et al. 2006, 2010; Cohen 2011, 2006).

Wallace et al. conducted a series of studies with machine learning techniques, especially active learning (Wallace et al. 2010a, b, 2011, 2012, 2013a, b; Nguyen et al. 2015). Wallace first set up a baseline approach called "patient active learning" (Wallace'10) for machine learning assisted citation screening (Wallace et al. 2010b). The performance of patient active learning is good enough (nearly 100% of the "relevant" citations can be retrieved at half of the conventional review cost) to convince systematic review conductors to adopt machine learning assisted citation screening. Instead of improving this baseline method, Wallace then focused on other aspects of machine learning assisted citation screening such as introducing external expert knowledge (Wallace et al. 2010a), allocating review tasks to multiple experts (Wallace et al. 2011) or to crowdsourcing workers (Nguyen et al. 2015), and building a tool called abstrackr to provide overall support (Wallace et al. 2012). Wallace's work on this topic is of exemplary high-impact and his core algorithm (on simple expert screening), is one of the most popular active learning techniques we have found in the evidence-based medical literature. That said, this technique has not been updated since 2010 (Wallace et al. 2010b). In this paper we are focused on the core active learning algorithm for cost minimization. Hence, we do not explore techniques such as Wallace's use of multiple experts (but in future work, we will explore this approach).

More recent work of Miwa et al. explored alternative data balancing and query strategy in 2014 (Miwa et al. 2014) and proposed a new treatment of Certainty plus Weighting (Miwa'14). Instead of uncertainty sampling in patient active learning (Wallace'10), Miwa found that certainty sampling provides better results in clinical citation screening tasks. Similar conclusion for data balancing method as weighting relevant examples was found to be more effective than aggressive undersampling. Although not stated explicitly, Certainty plus Weighting keeps training until all "relevant" studies have been discovered, which differs from the stopping criteria of Wallace'10. Aside from the core algorithm, additional views from latent Dirichlet allocation (LDA) has been found to be potentially useful.

Other work related to machine learning assisted citation screening do not utilize active learning. Pure supervised learning requires a sufficiently large training set, which leads to a huge review cost (Cohen et al. 2006; Adeva et al. 2014). Semi-supervised learning (Liu et al. 2016) does not utilize the human reviewers' feedback for updating the model, which leads to a depreciated performance in a long run. As a result, the patient active learning proposed by Wallace et al. (2010b) and the Certainty plus Weighting approach by Miwa et al. (2014) are still considered to be the state-of-the-art method for citation screening in the scenario with no external knowledge and equally expensive reviewers. Details on these two approaches are provided in Section 3.

There are also existing tools to support study selection in systematic reviews, e.g. Abstrakr[1] (Wallace et al. 2012), EPPI-Reviewer[2] (Thomas et al. 2010), Rayaan[3] (Ouzzani et al. 2016). Useful features can be found in these tools such as a) Rayaan and EPPI-Reviewer: incorporated keyword search in screening; b) Rayaan and EPPI-Reviewer: deduplication; c) Rayaan and EPPI-Reviewer: define inclusion/exclusion criteria by terms; d) Abstrakr: user defined tags; e) all three: assign review tasks to multiple reviewers; f) all three: automatically extract data from PubMed. However, the active learning parts alone in these tools are depreciated. Under the condition that no additional feature (search, tags, define inclusion/exclusion terms) is used, we tried all three tools with one of our dataset–Hall set (104 relevant in 8911 studies) and after reviewing 1000 studies, only 10 to 15 relevant ones were found, which was very close to a random sampling result without any learning. Since none of these tools are open-source, we cannot tell whether active learning is applied or how/when it is applied in each tool. This motivates us to develop an open source tool which focuses on active learning to support the primary study selection process. Details about our tool are presented in Section 6.

3 Technical Details

As mentioned in Section 2.1, the existing state-of-the-art methods are Wallace'10 (Wallace et al. 2010b) (patient active learning), Miwa'14 (Miwa et al. 2014) (Certainty plus Weighting), and Cormack'14 (Cormack and Grossman 2014) (continuous active learning). All three state-of-the-art methods share the following common techniques.

Support vector machines (SVM) are a well-known and widely used classification technique. The idea behind is to map input data to a high-dimension feature space and then construct a linear decision plane in that feature space (Cortes and Vapnik 1995). Linear SVM (Joachims 2006) has been proved to be a useful model in SE text mining (Krishna et al. 2016) and is applied in the state-of-the-art active learning methods of both evidence-based medicine and electronic discovery (Miwa et al. 2014; Wallace et al. 2010b; Cormack and Grossman 2014). One drawback of SVM is its poor interpretability as compared to classifiers like decision trees. However, SVM still fits here since the model itself is not important as long as it could provide a relative ranking of literature.

Active learning is a cost-aware machine learning algorithm where labels of training data can be acquired with certain costs. The key idea behind active learning is that a machine learning algorithm can perform better with less training if it is allowed to choose the data from which it learns (Settles 2012). There are several scenarios active learning is applied to, such as membership query synthesis, stream-based selective sampling, and pool-based sampling (Settles 2010). There are also different query strategies of active learning, such as uncertainty sampling, query-by-committee, expected model change, expected error reduction, variance reduction, and density-weighted methods (Settles 2010). Here, we briefly introduce one scenario and two query strategies, which are used in our later experiments and discussions.

[1] http://abstrackr.cebm.brown.edu

[2] http://eppi.ioe.ac.uk/cms/er4/

[3] http://rayyan.qcri.org/

Figure 2 shows a simple demonstration of an SVM active-learner. For the sake of simplicity, this demonstration assumes that the data has two features (shown in that figure as the horizontal and vertical axis). In that figure, "*O*" is the minority class, "relevant" studies in SLR. "*O*"s in blue are studies already identified as "relevant" (included) by human reviewers. "*X*" is the majority class, "irrelevant" studies in SLR. "*X*"s in red are studies already identified as "irrelevant" (excluded) by human reviewers (note that in (c), some red "*X*"s are removed from the training set by aggressive undersampling). Markers in gray are the unlabeled studies (studies have not been reviewed yet), and black line is SVM decision plane. In (b) Weighting balances the training data by putting more weight on the minority class examples. In (c), aggressive undersampling balances the training data by throwing away majority class examples closest to the old decision plane in (a). When deciding which studies to be reviewed next, uncertainty sampling returns the unlabeled examples closest to the decision plane (U) while certainty sampling returns the unlabeled examples furthest to the decision plane from the bottom-left side (C).

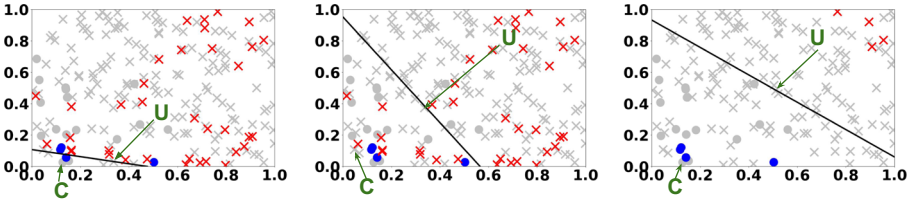
By analyzing the differences between the state-of-the-art methods, we identified the following key components in solving the problem with active learning and linear SVM.

When to Start Training

- **P** stands for "patient". As suggested by Wallace et al. (2010b), "hasty generation", which means start training with too few relevant examples, may lead to poor performance. The algorithm keeps random sampling until a sufficient number of "relevant" studies are retrieved. In our experiments, the sufficient number of "relevant" studies retrieved is set to 5, which means when at least 5 "relevant" studies have been retrieved by random sampling, the algorithm goes into next stage. Wallace'10 (Wallace et al. 2010b) and Miwa'14 (Miwa et al. 2014) use **P** for when to start training.
- **H** stands for "hasty", which is the opposite of **P**. The algorithm starts training as soon as *ONE* "relevant" study is retrieved, as suggested in Cormack'14 (Cormack and Grossman 2014, 2015).

Which Document to Query Next

- **U** stands for "uncertainty sampling". The algorithm utilizes uncertainty sampling to build the classifier, where unlabeled examples closest to the SVM decision plane are



(a) No data balancing (b) With weighting (c) With aggressive undersampling

Fig. 2 Active learning with SVM and different data balancing techniques

sampled for query (U in Fig. 2). Wallace'10 (Wallace et al. 2010b) uses **U** for Query strategy.
– **C** stands for "certainty sampling". The algorithm utilizes certainty sampling to build the classifier, where unlabeled examples furthest to the SVM decision plane and lie in the "relevant" side are sampled for query (C in Fig. 2). Miwa'14 (Miwa et al. 2014) and Cormack'14 (Cormack and Grossman 2014, 2015) use **C** for query strategy.

Whether to Stop Training (or not)

– **S** stands for "stop training". The algorithm stops training once the classifier is stable. In our experiments, the classifier is treated as stable once more than 30 "relevant" studies have been retrieved as training examples. Wallace'10 (Wallace et al. 2010b) uses **S** for whether to stop training.
– **T** stands for "continue training". The algorithm never stops training as suggested in Cormack'14 (Cormack and Grossman 2014) and Miwa'14 (Miwa et al. 2014). If query strategy is **U**, algorithm switches to certainty sampling after classifier is stable but training never stops.

How to Balance the Training Data

– **N** stands for "no data balancing". The algorithm does not balance the training data (demonstrated in Fig. 2a) as suggested by Cormack'14 (Cormack and Grossman 2014).
– **A** stands for "aggressive undersampling". The algorithm utilizes aggressive undersampling[4] after classifier is stable, as suggested by Wallace'10 (Wallace et al. 2010b).
– **W** stands for "Weighting". The algorithm utilizes Weighting[5] for data balancing (before and after the classifier is stable), as suggested by Miwa'14 (Miwa et al. 2014).
– **M** stands for "mixing of Weighting and aggressive undersampling". Weighting is applied before the classifier is stable while aggressive undersampling is applied after the classifier is stable. This treatment comes from the observation that "Weighting" performs better in early stages while "aggressive undersampling" performs better in later stages.

By combining different approaches, we ended up with 32 possible treatments including the state-of-the-art methods:

– The **PUSA** approach advocated by Wallace'10 (Wallace et al. 2010b).
– The **PCTW** approach advocated by Miwa'14 (Miwa et al. 2014).
– The **HCTN** approach advocated by Cormack'14 (Cormack and Grossman 2014).

[4]Aggressive undersampling throws away majority (irrelevant) training examples closest to SVM decision plane until reaching the same number of minority (relevant) training examples. A demonstration is shown in Fig. 2c.

[5]Weighting assigns different weight to each class, $W_R = 1/|L_R|$, $W_I = 1/|L_I|$, when training SVM. A demonstration is shown in Fig. 2b. L_R is defined in Fig. 3.

The algorithm on this page use
the following notation.

- E: the set of all candidate studies (returned from search).
- $R \subset E$: the set of ground truth relevant studies.
- $I = E \setminus R$: the set of ground truth irrelevant studies.
- $L \subset E$: the set of labeled/reviewed studies, each review reveals whether a study $x \in R$, but incurs a cost.
- $\neg L = E \setminus L$: the set of unlabeled/unreviewed studies.
- $L_R = L \cap R$: the identified relevant (included) studies.
- $L_I = L \cap I$: the identified irrelevant (excluded) studies.

The general form of the excluding irrelevant studies problem can be described as following: start with $L = \emptyset$, prioritize which studies to be reviewed so as to maximize $|L_R|$ while minimizing $|L|$ (identify more relevant studies with less cost).

Fig. 3 Notations and problem description

Pseudo code for the 32 machine learning treatments are shown in Algorithm 1. Along with the current standard procedure as a baseline approach:

– **Linear Review**: no machine learning, query studies in a random order.

All 32 machine learning treatments are tested and compared in Section 4.

Algorithm 1 Psuedo code

Input : E, set of all candidate studies
R, set of ground truth relevant studies
AL, algorithm code, e.g. **PUSA** for Wallace'10 treatment
Output : L_R, set of included studies

$L \leftarrow \emptyset$;
$L_R \leftarrow \emptyset$;
$\neg L \leftarrow E$;
// Keep reviewing until stopping rule satisfied
while $|L_R| < 0.95|R|$ **do**
  // Start training or not
  if $|L_R| \geq Enough(AL)$ **then**
    if $T \in AL$ ***or*** $NotStable(L_R)$ **then**
      $CL \leftarrow Train(AL, L)$;
    // Query next
    $x \leftarrow Query(AL, CL, \neg L, L_R)$;
  else
    // Random Sampling
    $x \leftarrow Random(\neg L)$;
  // Simulate review
  $L \leftarrow L \cup x$;
  $\neg L \leftarrow \neg L \setminus x$;
  if $x \in R$ **then**
    $L_R \leftarrow L_R \cup x$;
return L_R;

Support functions for Algorithm 1

```
t1 ← 5;
t2 ← 30;
Function Enough(AL)
    if P ∈ AL then
        return t1;
    else if H ∈ AL then
        return 1;
    else
        return ∞;
Function NotStable(L_R)
    if |L_R| ≥ t2 then return False ;
    else return True ;
Function Train(AL, L)
    if W ∈ AL or M ∈ AL then
        // Train linear SVM with Weighting
        CL ← SVM(L, kernel = linear, class_weight = balanced);
    else
        // Train linear SVM
        CL ← SVM(L, kernel = linear);
    if A ∈ AL or M ∈ AL then
        if ¬ NotStable(L_R) then
            // Aggressive undersampling
            L_I ← L \ L_R;
            tmp ← argsort(CL.decision_function(L_I))[: |L_R|];
            CL ← SVM(L_R ∪ tmp, kernel = linear);
    return CL;
Function Query(AL, CL, ¬L, L_R)
    if U ∈ AL and NotStable(L_R) then
        // Uncertainty Sampling
        x ← argsort(abs(CL.decision_function(¬L)))[0];
    else
        // Certainty Sampling
        x ← argsort(CL.decision_function(¬L))[−1];
    return x;
```

4 Experiments

This section describes the experimental procedures that we used to evaluate the treatments described in Section 3.

4.1 Performance Metrics

As shown in Fig. 3, the problem to be solved is multi-objective. The performance of each algorithm is thus usually evaluated by its **recall** ($|L_R|/|R|$) vs. **studies reviewed** ($|L|$) curve.

This performance metrics is suggested by Cormack et al. (Cormack and Grossman 2014, 2015) and best fits the objectives of excluding irrelevant studies problem. To enable a statistical analysis of the performances, the **recall** vs. **studies reviewed** curve is cut by a 0.95 **recall** line where **studies reviewed** ($|L|$) when reaching 0.95 **recall** ($|L_R| \geq 0.95|R|$) is

used to assess performances. The reason behind 0.95 **recall** is that a) 1.00 **recall** can never be guaranteed by any text mining method unless all the candidate studies are reviewed; b) 0.95 **recall** is usually considered acceptable in evidence-based medicine (Cohen 2011; Cohen et al. 2006; O'Mara-Eves et al. 2015) despite the fact that there might still be "relevant" studies missing (Shemilt et al. 2016). As a result, two metrics are used for evaluation:

– $X95 = \min\{|L| \mid |L_R| \geq 0.95|R|\}$.
– $WSS@95 = 0.95 - X95/|P|$.

Note that one algorithm is better than the other if its X95 is smaller or WSS@95 (Cohen 2011) is larger.

4.2 Datasets

Although a large number of SLRs are published every year, there is no dataset clearly documenting the details in primary study selection. As a result, three datasets are created reverse-engineering existing SLRs and being used in this study to simulate the process of excluding irrelevant studies. The three datasets are named after the authors of their original publication source– Wahono dataset from Wahono (2015), Hall dataset from Hall et al. (2012), and Radjenović dataset from Radjenović et al. (2013).

For each of the datasets, the search string **S** and the final inclusion list **F** from the original publication are used for the data collection. We retrieve the initial candidate collection **E** from IEEE Xplore with the search string (slightly modified to meet the requirement of IEEE Xplore). Then make a final list of inclusion **R** as $\mathbf{R} = \mathbf{F} \cap \mathbf{E}$. Here, for simplicity we only extract candidate studies from IEEE Xplore. We will explore possibilities for efficiently utilizing multiple data sources in the future work but in this paper, without loss of generality, we only extract initial candidate list from single data source. In this way, we created three datasets that reasonably resemble real SLR selection results assuming that any study outside the final inclusion list **F** is irrelevant to the original SLRs. A summary of the created datasets is presented in Table 1.

Apart from the three created datasets, one dataset (Kitchenham) is provided directly by the author of Kitchenham et al. (2010) and includes two levels of relevance information.

Table 1 Descriptive statistics for experimental datasets

Datasets	Generated		Original	
	#Candidate $\|\mathbf{E}\|$	#Relevant $\|\mathbf{R}\|$	#Candidate	#Relevant $\|\mathbf{F}\|$
Wahono	7002	62	2117	72
Hall	8911	104	2073	136
Radjenović	6000	48	13126	106
Kitchenham	1704	44 (132)	1704	44 (132)

Our datasets are generated using information in the original SLR literature. Our candidate studies are retrieved by applying similar if not the same the search string from original SLR literature and search in IEEE Xplore. The set of our relevant studies is the intersection of the set of our candidate studies and the set of final included studies in the original SLR literature. Kitchenham dataset is different as it is provided directly by Kitchenham and it has two level of relevance labels– 132 relevant studies by title and abstract review and within which, 44 relevant studies by content review

In general, only the "content relevant" labels are used in experiments for a fair comparison with other datasets. Additionally, the "abstract relevant" labels are used for detailed review cost analysis in RQ3. Summary of Kitchenham dataset is also presented in Table 1.

All the above datasets are available on-line at Seacraft, Zenodo.[6]

4.3 Simulation Studies

In the following, each experiment is a simulation of one specific treatment on one dataset. More specifically, there is no human activity involved in these experiments, when asked for a label, the true label in the dataset is queried instead of a human reviewer. As a result, each experiment can be repeated with different random seed to capture variances and also makes reproducing the experiments possible.

4.4 Controlled Variables

For the sake of a fair comparison, different treatments in Section 3 share an identical set of controlled variables including preprocessing, featurization and classifier.

Each candidate study in the initial list is first tokenized by stop words removal after concatenating its title and abstract. After tokenization, the bag of words are featurized into a term frequency vector. Then, reduce the dimensionality of the term frequency vector with to keep only $M = 4000$ of the terms with highest tf-idf[7] score and normalize the hashed matrix by its L2 norm each row at last. TfidfVectorizer in scikit-learn is utilized for the above preprocessing and featurization steps. Alternatives such as stemming, LDA (Blei et al. 2003), paragraph vectors (Le and Mikolov 2014) require further exploration and are scheduled in our future works. All 32 treatments use the same classifier– linear SVM from scikit-learn.

5 Results

All the following results were generated from 30 repeats simulations, using different random number seeds from each simulation. As shown below, all our results are reported in terms of medians (50th percentile) and iqrs ((75-25)th percentile).

RQ1: Can Active Learning Techniques Reduce Effort in Primary Study Selection? In Table 2, we tested 32 active learning treatments and linear review. According to the results, most active learning treatments perform consistently better than linear review (colored in blue) on all four datasets while four treatments (**HCS***) can be even worse than linear review. Interestingly these four treatments share same codes of **HCS**, which hastily start training (**H**) with greedy query strategy (**C**) and give up the attempt to correct the model short after (**S**). The problem of "hasty generation" is maximized in the setting of **HCS** and thus leads to an even worse performance than linear review. In general, other active learning treatments can reduce review costs by allowing the reviewer to read fewer studies while

[6] https://doi.org/10.5281/zenodo.1162952

[7] For term t in document d, $Tfidf(t,d) = w_d^t \times \left(\log \frac{|D|}{\sum_{d\in D} sgn(w_d^t)} + 1\right)$ where w_i^t is the term frequency of term t in document d. For term t, $Tfidf(t) = \sum_{d\in D} Tfidf(t,d) = \sum_{d\in D} w_d^t \times \left(\log \frac{|D|}{\sum_{d\in D} sgn(w_d^t)} + 1\right)$ and is used for feature selection.

Table 2 Scott-Knott analysis for number of studies reviewed/ work saved over sampling to reach 95% recall

		X95		WSS@95				X95		WSS@95	
Rank	Treatment	Median	IQR	Median	IQR	Rank	Treatment	Median	IQR	Median	IQR
Wahono						Hall					
1	HUTM	670	230	0.85	0.04	1	HUTW	340	90	0.91	0.01
1	HCTM	740	220	0.84	0.03	1	HUTA	340	130	0.91	0.02
2	HUTA	780	140	0.84	0.02	1	HUTM	350	120	0.91	0.01
2	HCTW	790	90	0.84	0.02	1	HCTW	370	60	0.91	0.01
2	HUTW	800	110	0.84	0.02	2	HUTN	370	90	0.91	0.01
2	HCTA	800	140	0.83	0.02	2	HCTM	380	100	0.91	0.01
3	PCTM	1150	450	0.78	0.07	2	HCTA	390	150	0.91	0.02
3	PUTM	1180	420	0.78	0.07	2	HCTN	410	80	0.90	0.01
3	PCTA	1190	340	0.78	0.05	3	HUSM	530	120	0.89	0.01
3	PUTA	1190	340	0.78	0.05	3	HUSW	560	250	0.89	0.03
3	PCTW	1210	350	0.78	0.06	3	PCTW	610	210	0.88	0.02
3	PUTW	1220	370	0.77	0.06	3	PUTW	610	220	0.88	0.03
4	HUSM	1410	400	0.75	0.06	4	HUSA	630	170	0.88	0.02
5	HUSA	1610	370	0.72	0.07	4	PCTN	650	220	0.88	0.03
6	PUSM	1810	370	0.69	0.06	4	PUTN	650	220	0.88	0.03
6	PUSA	1910	700	0.67	0.10	4	PUTM	670	220	0.87	0.03
7	HUSW	2220	400	0.63	0.06	4	PCTM	680	230	0.87	0.03
7	PUSW	2240	360	0.63	0.06	4	PCTA	700	210	0.87	0.03
8	HUTN	2700	40	0.56	0.01	4	PUTA	700	220	0.87	0.03
8	HCTN	2720	40	0.56	0.01	4	PUSW	740	230	0.87	0.03
8	PCSW	2860	1320	0.54	0.20	5	PUSM	770	240	0.86	0.03
8	PCSM	2860	1320	0.54	0.20	5	PUSA	880	270	0.85	0.04
8	PCTN	2850	1130	0.54	0.17	6	PCSW	1150	570	0.82	0.07
8	PUTN	2850	1130	0.54	0.17	6	PCSM	1150	570	0.82	0.07
9	PCSN	3020	1810	0.51	0.26	7	PCSN	1530	1050	0.78	0.13
9	PCSA	3020	1810	0.51	0.26	7	PCSA	1530	1050	0.78	0.13
10	HUSN	4320	110	0.33	0.03	7	PUSN	1550	1120	0.77	0.13
10	PUSN	4370	1290	0.32	0.19	7	HUSN	1800	1020	0.74	0.11
11	Linear	6650	0	0	0	8	HCSA	7470	5980	0.03	0.67
11	HCSA	6490	2760	−0.01	0.39	8	HCSN	7470	5980	0.03	0.67
11	HCSN	6490	2760	−0.01	0.39	8	linear	8464	0	0	0
11	HCSM	6490	3110	−0.01	0.44	8	HCSM	8840	6060	−0.04	0.68
11	HCSW	6490	3110	−0.01	0.44	8	HCSW	8840	6060	−0.04	0.68

Table 2 (continued)

Radjenović						Kitchenham					
1	HUTM	680	180	0.83	0.03	1	HUSA	590	170	0.60	0.19
1	HCTM	780	130	0.82	0.02	1	HUTA	590	80	0.60	0.06
1	HCTA	790	180	0.82	0.03	1	HUSM	620	70	0.58	0.04
1	HUTA	800	180	0.82	0.03	1	HUTM	630	110	0.58	0.07
2	HUSA	890	310	0.80	0.06	1	PUSA	640	130	0.57	0.08
2	HUSM	890	270	0.80	0.05	1	HUSW	640	140	0.57	0.09
3	HUTW	960	80	0.79	0.02	2	HUTN	680	30	0.55	0.02
3	HCTW	980	60	0.79	0.01	2	HCTA	680	100	0.55	0.08
3	HUSW	1080	410	0.77	0.07	2	PUSM	680	90	0.55	0.06
4	PCTM	1150	270	0.76	0.05	2	HCTM	680	110	0.55	0.07
4	PUTM	1150	270	0.76	0.05	2	PCTM	690	90	0.54	0.06
5	HUTN	1250	100	0.74	0.02	2	PUTM	690	70	0.54	0.05
5	PCTA	1260	210	0.74	0.05	2	PUTA	710	110	0.53	0.08
5	PUTA	1260	210	0.74	0.05	2	HUTW	710	20	0.53	0.02
5	HCTN	1270	70	0.74	0.02	3	PUSW	720	110	0.52	0.08
5	PUSM	1250	400	0.74	0.07	3	PCTA	720	100	0.52	0.08
5	PUSW	1250	450	0.73	0.08	3	HCTN	730	60	0.52	0.04
5	PUTW	1350	310	0.72	0.06	3	HCTW	750	60	0.51	0.04
5	PCTW	1370	310	0.72	0.06	3	PUTN	750	80	0.51	0.05
5	PUSA	1400	490	0.71	0.09	4	PCTN	750	80	0.51	0.05
6	HUSN	1570	300	0.69	0.05	4	PUTW	780	70	0.49	0.04
6	PCTN	1600	360	0.68	0.06	4	PCTW	780	150	0.49	0.09
6	PUTN	1600	360	0.68	0.06	5	PUSN	800	140	0.47	0.09
7	PUSN	1890	320	0.64	0.06	5	HUSN	870	280	0.43	0.16
8	PCSW	2250	940	0.57	0.20	6	PCSW	990	330	0.35	0.19
8	PCSM	2250	940	0.57	0.20	6	PCSM	990	330	0.35	0.19
9	PCSN	2840	1680	0.47	0.31	6	PCSN	1050	370	0.32	0.24
9	PCSA	2840	1680	0.47	0.31	6	PCSA	1050	370	0.32	0.24
10	HCSA	5310	2140	0.07	0.36	7	linear	1615	0	0	0
10	HCSN	5310	2140	0.07	0.36	7	HCSA	1670	60	−0.04	0.04
10	HCSM	5320	2200	0.02	0.37	7	HCSN	1670	60	−0.04	0.04
10	HCSW	5320	2200	0.02	0.37	7	HCSM	1680	60	−0.04	0.04
10	Linear	5700	0	0	0	7	HCSW	1680	60	−0.04	0.04

Simulations are repeated for 30 times, medians (50th percentile) and iqrs ((75–25)th percentile) are presented. Smaller/larger median value for X95/WSS@95 represents better performance while smaller iqr means better stability. Treatments with same rank have no significant difference in performance while treatments of smaller number in rank are significantly better than those of larger number in rank. The recommended treatment FASTREAD is colored in green while the state-of-the-art treatments are colored in red and linear review is colored in blue

still find 95% of the relevant ones. As for how much effort can be saved, **RQ3** will answer the question in details.

Based on the above, we say:

Finding 1
In general, active learning techniques can reduce cost in primary study selections with a sacrifice of (say 5%) recall.

RQ2: Should we Just Adopt the State-of-the-Art Treatments from Other Fields? Is it Possible to Build a Better One by Mixing and Matching from Those? In Table 2, performance of the three state-of-the-art treatments are colored in red. On Wahono datasets, Miwa'14 (**PCTW**) outperforms the other two treatments; while on Hall dataset, Cormack'14 (**HCTN**) has the best performance; on Radjenović dataset, all three treatments perform similarly; and on Kitchenham dataset, Wallace'10 (**PUSA**) outperforms the others. Neither of the three state-of-the-art treatments consistently performs the best. This means that adopting the state-of-the-art treatments will not produce best results. According to Scott-Knott analysis, the performance of one treatment, **HUTM** (colored in green), consistently stays in the top rank across all four datasets. Further, this treatment dramatically out-performs all three state-of-the-art treatments by requiring 20-50% fewer studies to be reviewed to reach 95% recall. We call this treatment FASTREAD. It executes as follows:

1. Randomly sample from unlabeled candidate studies until 1 "relevant" example retrieved.
2. Then start training with weighting and query with uncertainty sampling, until 30 "relevant" examples retrieved.
3. Then train with aggressive undersampling and query with certainty sampling until finished.

Hence, our answer to this research question is:

Finding 2
No, we should not just adopt the state-of-the-art methods from other fields. A better method called FASTREAD is generated by mixing and matching from the state-of-the-art methods.

RQ3: How much Effort can FASTREAD Save in an SLR? In terms of the number of studies reviewed, WSS@95 scores in Table 2 reflects how much FASTREAD can save. Number of "relevant" studies ($|R|$) and the total number of candidate studies ($|C|$) affect WSS@95 a lot, e.g. WSS@95=0.50 in Kitchenham dataset with $|R| = 44$, $|C| = 1704$ and WSS@95=0.91 in Hall dataset with $|R| = 104$, $|C| = 8911$. Even the smallest number of WSS@95=0.50 in Kitchenham dataset is a success in the reduction of number of studies need to be reviewed comparing to the 5% recall lost.

The above performance metrics can be used for comparing the performance of different algorithms. However, for a more realistic cost analysis, labeling/reviewing each study has different costs. For each studies in L, its abstract and title has been reviewed, thus costs C_A. In addition, there exists a set $L_D \subset L$, $L_R \subset L_D$ where studies in L_D have been reviewed by their contents, thus cost an additional C_D for each study. Table 3 shows how much FASTREAD save over reviewing all candidate studies. Suppose $C_D = 9C_A$, following the estimation that Shemilt made: 1 minute to screen a title-abstract record, 4 minutes to retrieve a full-text study report, and 5 minutes to screen a full-text study report (Shemilt et al. 2016).

Table 3 How much can FASTREAD save?

Datasets	# Studies reviewed	Review cost	# Missing relevant
Wahono	$7002 - 670 = 6332$	$\geq 6332C_A + 4C_D$	4
Hall	$8991 - 350 = 8641$	$\geq 8641C_A + 6C_D$	6
Radjenović	$6000 - 680 = 5320$	$\geq 5320C_A + 3C_D$	3
Kitchenham	$1704 - 630 = 1074$	$32C_D + 1074C_A$	3

Numbers of reviewing every candidate study minus numbers of reviewing with FASTREAD. For example, on Kitchenham dataset, FASTREAD reviews 944 fewer studies, which costs $32C_D + 944C_A$ less review effort, while misses 3 "relevant" ones. Here C_D is the cost to review a study by its content and C_A is the cost to review a study by its title and abstract

Then the reduction in review cost is $(32C_D + 1074C_A)/(132C_D + 1704C_A) = 47.1\%$.[8] On other datasets, although we do not have the exact number of "abstract relevant" studies, we can estimate the worst case review cost reduction[9] with the numbers in Tables 1 and 3: a) Wahono dataset: $1 - 670(C_A + C_D)/((670 + 4)C_D + 7002C_A) = 48.7\%$; b) Hall dataset: $1-360(C_A+C_D)/((360+6)C_D+8991C_A) = 70.7\%$; c) Radjenović dataset: $1-680(C_A+C_D)/((680+3)C_D+6000C_A) = 44.0\%$. Note that training time costs are negligibly small (1 second for each round in average) compared to the review time C_A because of the small training size (less than 1000 examples before reaching 95% recall).

Finding 3
Our results and estimations suggest that FASTREAD can save more than 40% of the effort (associated with the primary selection study phase of a literature review) while retrieving 95% of the "relevant" studies.

6 Tool Support

In order to implement FASTREAD, we developed a simple tool as shown in Fig. 4. This software is freely available from SeaCraft Zenodo at https://doi.org/10.5281/zenodo.837861 and its Github repository at https://github.com/fastread/src.

Using FASTREAD, a review starts with **A**: selecting the input candidate study list from *workspace/data/* directory. The input CSV file must have the *Document Title*, *Abstract*, *Year*, and *PDF Link* columns. The *label* column, which is the true label of the candidate studies, is optional and is only used for testing. The output CSV file generated by the FASTREAD tool has an additional *code* column, which is the reviewer-decided label for the candidate study. The final inclusion list can be retrieved by extracting all the studies with "yes" in the *code* column.

As shown by the annotations in Fig. 4, reviews using FASTREAD proceeds as follows:

[8] According to Table 1, reviewing all studies costs $132C_D + 1704C_A$. In our simulations, in average FASTREAD did 630 abstract reviews and 100 content reviews.

[9] In the worst case we assume that every study reviewed is "abstract relevant" and thus costs $C_D + C_A$ to review and there is no "abstract relevant" study left except for the 5% missing "content relevant" ones. E.g. in Wahono dataset, FASTREAD reviews 670 studies among the 7002 candidate ones, it costs $670(C_A + C_D)$ while reviewing all studies costs $(670 + 4)C_D + 7002C_A$.

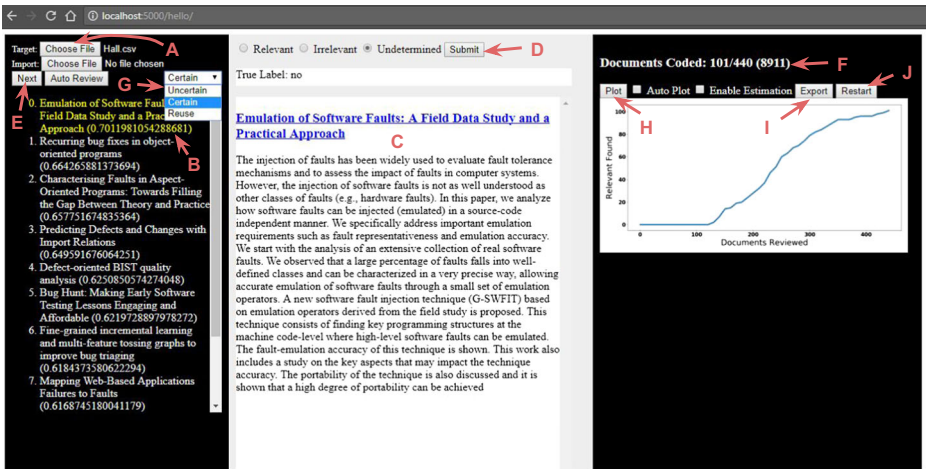


Fig. 4 Basic interface of the FASTREAD tool

- **B** Randomly select 10 candidate studies for review.
- **C** Read through the title and abstract (and click on the title and read the full text if needed) of the candidate study.
- **D** Decide whether this study should be coded as *Relevant* or *Irrelevant* then click *Submit*.
- **E** Click the *Next* button to save codes. 10 more candidates are then selected.
- **F** The review status will change every time new studies are coded by reviewer and the *Next* button is hit. The status is shown in the format "Documents Coded: *Number of relevant studies found* / *Number of studies reviewed* (*Total number of candidate studies*)."
- **G1** Once **1** "relevant" study is coded, *Random sampling* moves to *Uncertainty sampling*.
- **G2** Once **30** "relevant" study is coded, *Uncertainty sampling* can change *Certainty sampling*.
- **H** Fig. H can be plotted by clicking the *Plot* button or checking *Auto Plot* (figure cached in *src/static/image/* directory).
- **I** Once finished, coded studies can be exported into a CSV file in the *workspace/coded/* directory.

Note that the *Restart* button (**J**) is only for testing and discards all codes.

7 Discussion

7.1 What is Missed?

Our results will show, with FASTREAD, 95% of the "relevant" studies can be retrieved by reviewing a small portion (usually hundreds of studies) of long candidate study list. Given that, it is wise to reflect on the 5% of papers *not* found by such an analysis. To this end, we took one of our case studies and reflected on:

– The set of papers R that a human analyst declared to be "relevant" (as listed in their reference list at the end of their paper).

- The *tangentially relevant* subset of those papers $R_1 \subseteq R$ that a human analyst explicitly mentions, however briefly, in the body of their paper.
- The yet smaller subset of those papers $R_2 \subseteq R_1$ that a human analyst discusses, at length, in the body of their report (and for our purposes "at length" is "more that two lines"). We call these *insightful papers*. Clearly, FASTREAD should not be recommended if our method always misses the insightful papers.

For our case studies, on 30 repeats of our methods, we found that $R_2 \setminus L_R = \emptyset$; i.e. FASTREAD never missed an insightful paper. As for the tangentially relevant papers, FASTREAD found all of those in 95% of the 30 repeats. Based on this analysis, we infer that missing 5% of the papers is not a major impediment to using FASTREAD. Similar conclusion was derived by Shemilt et al. in 2016 (Shemilt et al. 2016). More interestingly, we found that more than 90% of the missing studies come from a same set of size of $0.1|R|$. Which means some studies are more likely to be missed while most studies have never been missed in 30 repeats. This may suggest that there are outliers in relevant studies, which are not very important according to our case studies.

That said, if the SLR conductor does not want to miss any potential relevant study, they need to review all the candidate studies with full cost. We are actively exploring possibilities to mitigate or compensate the missing studies issue. For example, one technique is "diversity sampling"; i.e. to explore unknown regions by sampling the least similar studies from what have been reviewed before. Exploration and exploitation can be balanced by selection different weight between diversity sampling and certainty/uncertainty sampling Note that more exploration means fewer missing studies but higher review cost.

7.2 What About Domain Knowledge?

In our simulations, we assume that no initial seed training set is available thus a random sampling is performed to collect the minimum training set. This assumption represents the worst case while no external knowledge is available. We show in this work that the absence of that domain knowledge is not a critical failing of the approach. On the other hand, such domain knowledge usually exists in real world SLRs and will boost the performance of FASTREAD if wisely used. For example, if one relevant example and one irrelevant example are known in the very beginning, the random sampling step of FASTREAD is no longer needed and thus leads to additional cost reduction. More details about how to wisely use domain knowledge to boost FASTREAD will be explored further after this work. While we have some preliminary results in that area, we have nothing definitive to report at this time.

7.3 What About Real Human Reviewers?

In our simulations, we assume that there is only one reviewer who never make mistakes. In real world SLRs, there will be multiple reviewers who make some mistakes.

To handle this, FASTREAD could be changed to one central learner with multiple review agents. Every agent reviews different studies and feedback his or her decisions to the central learner. The central learner then trains on the feedback of every agent and assigns studies to each agent for review. Such schema will keep all the property of single reviewer FASTREAD and performs similarly. In addition, there might be more intelligent way to allocate review tasks based on the different performance of review agents (Wallace et al. 2011).

Second, consider those multiple reviewers now make mistakes. Candidate studies need to be reviewed by multiple reviewers in case any of them makes mistakes. To explore this issue,

appropriate data need to be collected on how human reviewers make mistakes. Wallace et al. addressed this issue in 2015 (Nguyen et al. 2015) by analyzing the best policy for allocating review tasks to reviewers with different experience levels as well as difference costs. We also plan to to address this issue in our future work.

7.4 What About Multiple Categories of Studies?

In our simulations, we assume that the target is binary classification. However, primary study selection in real world SLRs might be a multi-label classification problem. For example, an SLR with two research questions might go through a primary study selection while each candidate is labeled as "relevant to RQ1", "relevant to RQ2", or "irrelevant" while the first two labels can co-exist. The simplest solution for this is to run multiple FASTREAD learners each learns on one label vs. others and each reviewer classify on one label only. In this case, the multi-label classification problem can be divided into multiple FASTREAD problems. Additional work such as ensemble learners can be explored in future works.

8 Threats to Validity

There are several validity threats to the design of this study (Feldt and Magazinius 2010). Any conclusions made from this work must be considered with the following issues in mind:

Conclusion validity focuses on the significance of the treatment. To enhance the conclusion validity of this work, we employed several statistical tests (Scott-Knott) to reduce the changes of making spurious conclusions.

Internal validity measures how sure we can be that the treatment actually caused the outcome. To enhance internal validity, we heavily constrained our experiments (see our simulated results in strictly controlled environments as discussed in Section 4.4).

Construct validity focuses on the relation between the theory behind the experiment and the observation. In this work, we evaluated our results via different treatments with WSS@95 as stated in Section 4.1– note that those measures took us as close as we can to computing cost reduction without "abstract relevant" information. That is, it fits the objective of human-in-the-loop primary study selection as defined in the current literature (Cormack and Grossman 2014, 2015). Increasing the number of different measures may increase construct validity so, in future work, we will further explore more metrics.

External validity concerns how well the conclusion can be applied outside. All the conclusions in this study are drawn from the experiments running on three software engineering SLR datasets created with information from Hall, Wahono, Radjenović et al. studies (Hall et al. 2012; Wahono 2015; Radjenović et al. 2013) and one dataset provided by Kitchenham et al. (2010). Therefore, such conclusions may not be applicable to datasets of different scenarios, e.g., citation screening from evidence based medicine or TAR from e-discovery. Such bias threatens any classification experiment. The best any researcher can do is to document that bias then make available to the general research community all the materials used in a study (with the hope that other researchers will explore similar work on different datasets). Existing active learning techniques in citation screening have been criticized by Olorisade et al. for being not replicable (Olorisade et al. 2016, 2017). To this end, we have published all our code at https://github.com/fastread/src and all our data at https://doi.org/10.5281/zenodo.1162952.

In the experiments, we assume that the human reviewer is always correct. In practice, this assumption cannot hold and problems such as disagreement between reviewers or concept drift (in which reviewers disagree with themselves as time passes) may occur. As discussed below when we discuss *Future Work*, we intend to explore this matter in the near future.

The comparisons in our experiment are based on the controlled variables listed in Section 4.4. If those settings change, then the conclusion in Section 5 may become unreliable.

9 Conclusions

Systematic literature reviews are the primary method for aggregating evidence in evidence-based software engineering. It is suggested for every researcher in software engineering to frequently conduct SLRs (Keele 2007). One drawback with such SLRs is the time required to complete such a study: an SLR would can weeks to months to finish and the conclusion drawn can be out of date in a few years.

To tackle this barrier to understanding the literature, this study focuses on primary study selection, one of the most difficult and time consuming steps in an SLR. Machine learning methods, especially active learning, are explored in our attempts to reduce the effort required to exclude primary studies. In this paper:

- We explored 32 different active learners. To the best of our knowledge, this is largest such study yet completed in the software engineering domain.
- We have collected data from four large literature reviews. This data is publically available (https://doi.org/10.5281/zenodo.1162952). Note that the creation and distribution of these data sets is an important contribution, because prior to this study, it was difficult to obtain even one such data set.
- We have offered a baseline result that can serve as a challenge problem for SE researchers: how to find more relevant papers after reviewing fewer papers. We have placed in the public domain (github.com/fastread/src) software tools that let others compare our approach with alternative methods.
- We created a new reading-assistant tool called FASTREAD. To the best of our knowledge, FASTREAD's combination of methods has not been previously explored.
- Using FASTREAD, we decreased the number of studies to be reviewed by 20-50% (comparing to the prior state-of-the-art).

As a result of the above we can:

- Offer much assistance to any future SLR.
- Offer a cautionary tale to SE researchers who use data miners. Specifically: do not be content with off-the-shelf solutions developed by other communities. SE has nuanced differences to other domains so our methods need to be tuned to our data. Even within the SE community there may be variations, so the framework provided by this paper is a good example to find the best method for a specific task on specific data.

10 Future Work

This study has several limitations as described in Section 7. We consider the limitations as open challenges and plan to address those in future work. Specific problems and plans for the future are listed below.

- *Conclusions are drawn from three synthetic SLR datasets and one Kitchenham dataset.* Validate the generalizability of the results on different datasets, including datasets from evidence-based medicine and e-discovery.
- *Experiment results are evaluated by WSS@95, which assumes a stop rule of reaching 95% recall.* How to stop at 95% recall without first knowing the number "relevant" studies in the pool is an interesting topic. We are exploring this topic actively.
- *The size and prevalence of data can affect performance of FASTREAD.* With the capability of cost reduction from FASTREAD, it is reasonable to ask whether we need the narrow initial search. An interesting future research would be to use every paper on, say Scopus, database as candidates and allow user to just using some simple search to initiate and guide the selection. As a result, the recall is no longer restricted by the initial search string thus may yield higher recall with reasonable cost.
- *About 10 to 20% efforts are spent on random selection step and most of the variances are also introduced in this step.* To speed up the random selection step, external expert knowledge will be introduced while unsupervised learning methods such as VTM, LDA, word2vec, or t-SNE will also be considered in future work.
- *Some magic parameters are arbitrarily chosen, which may affect the performance.* However, parameter tuning is not a good fit for human-in-the-loop primary study selection because a) parameters should be tuned for the data working on; b) but the effect of applying different parameters can not be tested since querying extra label incurs extra cost. Therefore, novel methods should be explored for parameter selection; e.g. better criterion for when to switch from uncertainty sampling to certainty sampling (instead of the "30" relevant examples rule applied now). Works from Borg (2016) and Fu et al. (2016) will be considered as candidate solutions to this problem.
- *Current scenario is restricted to having only one reviewer, which is impractical in practice.* Problems including how to assign review tasks to multiple reviewers and how to utilize reviewers with different cost and different capability will be explored in the future.
- *Currently, we assume that reviewers never make mistakes.* In future work, we will explore concept drift (reviewers disagree with themselves, at some later time) and how to settle disagreements (reviewers disagree with each other).
- *This study focuses only on primary study selection.* Assistance on other steps of SLR such as searching, data extraction, and protocol development can also help reduce total effort of SLRs. The potential of combining VTM, snowballing, and other tools with FASTREAD needs to be explored as well.

We invite other researchers to join us in the exploring the above.

Acknowledgements The authors thank Barbara Kitchenham for her attention to this work and for sharing with us the "Kitchenham" dataset used in our experiments.

References

Adeva JG, Atxa JP, Carrillo MU, Zengotitabengoa EA (2014) Automatic text classification to support systematic reviews in medicine. Expert Syst Appl 41(4):1498–1508

Bezerra YM, Pereira TAB, da Silveira GE (2009) A systematic review of software product lines applied to mobile middleware. In: Sixth international conference on information technology: new generations, 2009. ITNG'09. IEEE, pp 1024–1029

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3(Jan):993–1022

Borg M (2016) Tuner: a framework for tuning software engineering tools with hands-on instructions in r. Journal of Software Evolution and Process 28(6):427–459
Bowes D, Hall T, Beecham S (2012) Slurp: a tool to help large complex systematic literature reviews deliver valid and rigorous results. In: Proceedings of the 2nd international workshop on evidential assessment of software technologies. ACM, pp 33–36
Carver JC, Hassler E, Hernandes E, Kraft NA (2013) Identifying barriers to the systematic literature review process. In: 2013 ACM/IEEE international symposium on empirical software engineering and measurement. IEEE, pp 203–212
Cohen AM (2006) An effective general purpose approach for automated biomedical document classification. In: AMIA annual symposium proceedings, vol 2006. American Medical Informatics Association, p 161
Cohen AM (2011) Performance of support-vector-machine-based classification on 15 systematic review topics evaluated with the wss@ 95 measure. J Am Med Inform Assoc 18(1):104–104
Cohen AM, Hersh WR, Peterson K, Yen PY (2006) Reducing workload in systematic review preparation using automated citation classification. J Am Med Inform Assoc 13(2):206–219
Cohen AM, Ambert K, McDonagh M (2010) A prospective evaluation of an automated classification system to support evidence-based medicine and systematic review. In: AMIA annual symposium proceedings, vol 2010. American Medical Informatics Association, p 121
Cormack GV, Grossman MR (2014) Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In: Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval. ACM, pp 153–162
Cormack GV, Grossman MR (2015) Autonomy and reliability of continuous active learning for technology-assisted review. arXiv:1504.06868
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
Dyba T, Kitchenham BA, Jorgensen M (2005) Evidence-based software engineering for practitioners. IEEE Softw 22(1):58–65. https://doi.org/10.1109/MS.2005.6
Feldt R, Magazinius A (2010) Validity threats in empirical software engineering research-an initial survey. In: SEKE, pp 374–379
Felizardo KR, Nakagawa EY, Feitosa D, Minghim R, Maldonado JC (2010) An approach based on visual text mining to support categorization and classification in the systematic mapping. In: Proc. of EASE, vol 10. pp 1–10
Felizardo KR, Andery GF, Paulovich FV, Minghim R, Maldonado JC (2012) A visual analysis approach to validate the selection review of primary studies in systematic reviews. Inf Softw Technol 54(10):1079–1091
Felizardo KR, Nakagawa EY, MacDonell SG, Maldonado JC (2014) A visual analysis approach to update systematic reviews. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, EASE '14. ACM, New York, pp 4:1–4:10. https://doi.org/10.1145/2601248.2601252
Felizardo KR, Mendes E, Kalinowski M, Souza ÉF, Vijaykumar NL (2016) Using forward snowballing to update systematic reviews in software engineering. In: Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement. ACM, p 53
Fernández-Sáez AM, Bocco MG, Romero FP (2010) SLR-Tool: a tool for performing systematic literature reviews. In: ICSOFT (2), pp 157–166
Fu W, Menzies T, Shen X (2016) Tuning for software analytics: is it really necessary? Inf Softw Technol 76:135–146
Grossman MR, Cormack GV (2013) The grossman-cormack glossary of technology-assisted review with foreword by john m. facciola, u.s. magistrate judge. Federal Courts Law Review 7(1):1–34
Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304
Hassler E, Carver JC, Kraft NA, Hale D (2014) Outcomes of a community workshop to identify and rank barriers to the systematic literature review process. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. ACM, p 31
Hassler E, Carver JC, Hale D, Al-Zubidy A (2016) Identification of SLR tool needs—results of a community workshop. Inf Softw Technol 70:122–129
Hernandes E, Zamboni A, Fabbri S, Thommazo AD (2012) Using gqm and tam to evaluate start-a tool that supports systematic review. CLEI Electronic Journal 15(1):3–3
Jalali S, Wohlin C (2012) Systematic literature studies: database searches vs. backward snowballing. In: Proceedings of the ACM-IEEE international symposium on empirical software engineering and measurement. ACM, pp 29–38
Joachims T (2006) Training linear svms in linear time. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 217–226

Keele S (2007) Guidelines for performing systematic literature reviews in software engineering. In: Technical report, Ver. 2.3 EBSE Technical Report. EBSE

Kitchenham B, Brereton P (2013) A systematic review of systematic review process research in software engineering. Inf Softw Technol 55(12):2049–2075

Kitchenham BA, Dyba T, Jorgensen M (2004) Evidence-based software engineering. In: Proceedings of the 26th international conference on software engineering. IEEE Computer Society, pp 273–281

Kitchenham B, Pretorius R, Budgen D, Brereton OP, Turner M, Niazi M, Linkman S (2010) Systematic literature reviews in software engineering–a tertiary study. Inf Softw Technol 52(8):792–805

Krishna R, Yu Z, Agrawal A, Dominguez M, Wolf D (2016) The bigse project: lessons learned from validating industrial text mining. In: Proceedings of the 2nd international workshop on BIG data software engineering. ACM, pp 65–71

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31st international conference on machine learning (ICML-14), pp 1188–1196

Liu J, Timsina P, El-Gayar O (2016) A comparative analysis of semi-supervised learning: the case of article selection for medical systematic reviews. Inf Syst Front:1–13 https://doi.org/10.1007/s10796-016-9724-0

Malheiros V, Hohn E, Pinho R, Mendonca M, Maldonado JC (2007) A visual text mining approach for systematic reviews. In: First international symposium on empirical software engineering and measurement (ESEM 2007). IEEE, pp 245–254

Marshall C, Brereton P (2013) Tools to support systematic literature reviews in software engineering: a mapping study. In: 2013 ACM/IEEE international symposium on empirical software engineering and measurement. IEEE, pp 296–299

Marshall C, Brereton P, Kitchenham B (2014) Tools to support systematic reviews in software engineering: a feature analysis. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, EASE '14. ACM, pp 13:1–13:10

Marshall C, Brereton P, Kitchenham B (2015) Tools to support systematic reviews in software engineering: a cross-domain survey using semi-structured interviews. In: Proceedings of the 19th international conference on evaluation and assessment in software engineering. ACM, p 26

Miwa M, Thomas J, O'Mara-Eves A, Ananiadou S (2014) Reducing systematic review workload through certainty-based screening. J Biomed Inform 51:242–253

Molléri JS, Benitti FBV (2015) Sesra: a web-based automated tool to support the systematic literature review process. In: Proceedings of the 19th international conference on evaluation and assessment in software engineering, EASE '15. ACM, New York, pp 24:1–24:6. https://doi.org/10.1145/2745802.2745825

Nguyen AT, Wallace BC, Lease M (2015) Combining crowd and expert labels using decision theoretic active learning. In: Third AAAI conference on human computation and crowdsourcing

Olorisade BK, de Quincey E, Brereton P, Andras P (2016) A critical analysis of studies that address the use of text mining for citation screening in systematic reviews. In: Proceedings of the 20th international conference on evaluation and assessment in software engineering. ACM, p 14

Olorisade BK, Brereton P, Andras P (2017) Reproducibility of studies on text mining for citation screening in systematic reviews: evaluation and checklist. J Biomed Inform 73:1

O'Mara-Eves A, Thomas J, McNaught J, Miwa M, Ananiadou S (2015) Using text mining for study identification in systematic reviews: a systematic review of current approaches. Systematic Reviews 4(1):5

Ouzzani M, Hammady H, Fedorowicz Z, Elmagarmid A (2016) Rayyan—a web and mobile app for systematic reviews. Systematic Reviews 5(1):210. https://doi.org/10.1186/s13643-016-0384-4

Paynter R, Bañez LL, Berliner E, Erinoff E, Lege-Matsuura J, Potter S, Uhl S (2016) Epc methods: an exploration of the use of text-mining software in systematic reviews. Research white paper (prepared by the Scientific Resource Center and the Vanderbilt and ECRI Evidence-based Practice Centers under contract nos. HHSA290201200004C (SRC), HHSA290201200009I (Vanderbilt), and HHSA290201200011I (ECRI). Agency for Healthcare Research and Quality (US). http://www.effectivehealthcare.ahrq.gov/reports/final/cfm

Radjenović D, Heričko M, Torkar R, Živkovič A (2013) Software fault prediction metrics: a systematic literature review. Inf Softw Technol 55(8):1397–1418

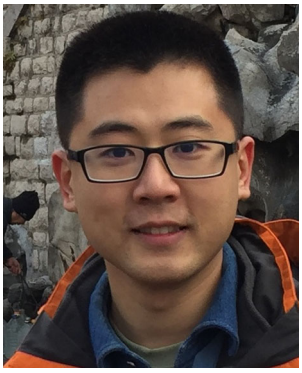
Roegiest A, Cormack GV, Grossman M, Clarke C (2015) Trec 2015 total recall track overview. Proc TREC-2015

Ros R, Bjarnason E, Runeson P (2017) A machine learning approach for semi-automated search and selection in literature studies. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering. ACM, pp 118–127

Settles B (2010) Active learning literature survey. University of Wisconsin, Madison 52(55-66):11

Settles B (2012) Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 6(1):1–114

Shemilt I, Khan N, Park S, Thomas J (2016) Use of cost-effectiveness analysis to compare the efficiency of study identification methods in systematic reviews. Systematic Reviews 5(1):140
Thomas J, Brunton J, Graziosi S (2010) Eppi-reviewer 4.0: software for research synthesis
Wahono RS (2015) A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks. J Softw Eng 1(1):1–16
Wallace BC, Small K, Brodley CE, Trikalinos TA (2010a) Active learning for biomedical citation screening. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 173–182
Wallace BC, Trikalinos TA, Lau J, Brodley C, Schmid CH (2010b) Semi-automated screening of biomedical citations for systematic reviews. BMC Bioinf 11(1):1
Wallace BC, Small K, Brodley CE, Trikalinos TA (2011) Who should label what? Instance allocation in multiple expert active learning. In: SDM. SIAM, pp 176–187
Wallace BC, Small K, Brodley CE, Lau J, Trikalinos TA (2012) Deploying an interactive machine learning system in an evidence-based practice center: abstrackr. In: Proceedings of the 2nd ACM SIGHIT international health informatics symposium. ACM, pp 819–824
Wallace BC, Dahabreh IJ, Moran KH, Brodley CE, Trikalinos TA (2013a) Active literature discovery for scoping evidence reviews: how many needles are there. In: KDD workshop on data mining for healthcare (KDD-DMH)
Wallace BC, Dahabreh IJ, Schmid CH, Lau J, Trikalinos TA (2013b) Modernizing the systematic review process to inform comparative effectiveness: tools and methods. Journal of Comparative Effectiveness Research 2(3):273–282
Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering. ACM, p 38
Wohlin C (2016) Second-generation systematic literature studies using snowballing. In: Proceedings of the 20th international conference on evaluation and assessment in software engineering. ACM, p 15
Zhang H, Babar MA, Bai X, Li J, Huang L (2011a) An empirical assessment of a systematic search process for systematic reviews. In: 15th annual conference on evaluation & assessment in software engineering (EASE 2011). IET, pp 56–65
Zhang H, Babar MA, Tell P (2011b) Identifying relevant studies in software engineering. Inf Softw Technol 53(6):625–637



Zhe Yu is a third year Ph.D. student in the department of Computer Science at North Carolina State University. He received his bachelor and master degree in Shanghai Jiao Tong University, China. His primary interest lies in the collaboration of human and machine learning algorithms that leads to better performance and higher efficiency than pure human or machine learning. He currently works on developing active learning algorithms and tools that help researchers conduct literature reviews. For more information, visit https://azhe825.github.io.

Nicholas A. Kraft is a software researcher at ABB Corporate Research in Raleigh, North Carolina. Previously, he was an associate professor in the Department of Computer Science at The University of Alabama. He received the Ph.D. degree in computer science from Clemson University in 2007. His research interests are in software evolution, with an emphasis on techniques and tools to support developers in understanding evolving software and to support managers in understanding software evolution processes. Dr. Kraft's research has been funded by grants from the NSF, DARPA, and ED. He currently serves on the editorial board of IEEE Software and on the steering committee of the IEEE International Conference on Software Maintenance and Evolution (ICSME). He is a senior member of the ACM and the IEEE.

Tim Menzies (Ph.D., UNSW, 1995) is a full Professor in CS at North Carolina State University. He researchers SE, data mining, AI, search-based SE, and open access science. A former SE research chair at NASA, he is the author of over 250 referred publications and he has been a lead researcher on projects for NSF, NIJ, DoD, NASA, USDA. Prof. Menzies is the cofounder of the PROMISE conference series devoted to reproducible experiments in software engineering (http://tiny.cc/seacraft). He has served as associate editor of IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering Methodologies, Empirical Software Engineering, the Automated Software Engineering Journal the Big Data Journal, Information Software Technology, IEEE Software, and the Software Quality Journal. He has also served as PC cochair for IEEE ASE (2012); ICSE NIER track (2015); Search-based SE (2017); and co-general chair of ICMSE (2016). For more, see his publications https://goo.gl/qNQAIq) or home page http://menzies.us.