Received 30 May 2023, accepted 26 June 2023, date of publication 4 July 2023, date of current version 18 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3292056

RESEARCH ARTICLE

Helping Pull Request Reviewer Recommendation Systems to Focus

NIKOLA PEJIĆ[1,2], ZAHARIJE RADIVOJEVIĆ[1], AND MILOŠ CVETANOVIĆ[1]
[1]School of Electrical Engineering, University of Belgrade, 11120 Belgrade, Serbia
[2]Microsoft Development Center Serbia, 11000 Belgrade, Serbia

Corresponding author: Miloš Cvetanović (cmilos@etf.bg.ac.rs)

This work was supported in part by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under Contract 451-03-47/2023-01/200103; and in part by the Science Fund of the Republic of Serbia under Contract AVANTES.

ABSTRACT The selection of code reviewers for a pull request can impact the quality as well as the speed of the review. In order to have the best experience both for the developer and the reviewers, there is a need for automatic reviewer recommendation systems for pull requests. Although there has been previous research in this area, it was mainly focused on smaller and medium repositories of up to around 200 developers, while larger repositories were rarely targeted. In this paper we evaluate several existing approaches on a set of 8 Microsoft repositories of different sizes, noticing that the average performance of the approaches seemed to decrease with the number of reviewers the repository has. In order to focus the existing approaches only on relevant reviewers, we propose a technique for improving their performance by scoping down the set of candidate reviewers based on multiple filters. We defined several basic filters and determined that 5 out of the 7 tested existing approaches experienced performance improvements of up to 16.24% better precision and 19.66% better recall averaged over all datasets, with the per dataset improvements peaking at 36.63% better precision and 28.63% better recall. Additionally, by combining different basic filters we were able to achieve additional improvements for 6 out of the 7 existing approaches (with a minor improvement for the remaining approach), which on average over all datasets had up to 17.60% better precision and 21.23% better recall, while the per dataset improvements peaked at 43.28% better precision and 30.94% better recall.

INDEX TERMS Modern code review, pull request, reviewer recommendation.

I. INTRODUCTION

The number of software repositories has been growing over the years, and so has the number of their contributors [1]. This is true both for open-source and closed-source software. Developers are starting to feel overloaded [2], so in order to help them to keep up with the growing load and be able to focus on important tasks, there is a need to automate repetitive work as much as possible.

An area that has received attention over the past years is the reviewer recommendation problem for pull requests in the pull-based development model. The pull-based model functions by having the developers design and implement their changes on their copy of the codebase (either a fork of the repository or a developer branch), and once they wish for the change to be integrated they send a pull request (PR). The maintainers of the codebase (reviewers) perform a code review by assessing the quality of the code, looking for potential bugs and regressions, and raising any concerns by leaving comments on the PR. The selection of reviewers assigned to a PR can influence the time to completion of the PR [3], [4], as well as the quality of the code review [5], [6], [7]. This is even more important for large repositories with many developers, as developers (and reviewers) tend to specialize in smaller parts of the codebase [8]. While in smaller teams it might be acceptable to always choose the best matches for a given PR, bigger teams might prefer ‘‘local’’ experts (from the same subteam as the PR developer), who are more familiar with the changes made in the PR. Rather than assigning global experts to be reviewers every time there are code changes in a certain area (and thus overloading them with incoming PRs), subteams can decide to favor local experts who, although

The associate editor coordinating the review of this manuscript and approving it for publication was Hailong Sun.

© 2023 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

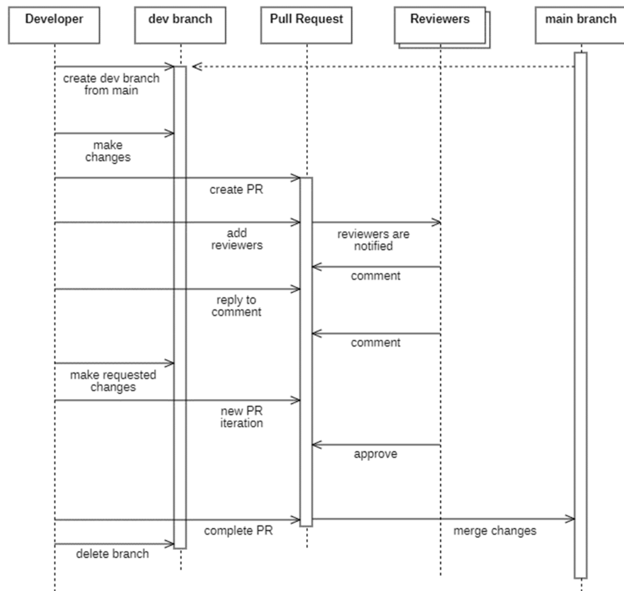

FIGURE 1. A sequence diagram illustrating the full lifecycle of a PR. After the PR is created, the developer interacts with the reviewers via comments, the PR is improved based on the reviewers' suggestions, and ultimately it is completed and the changes are merged.

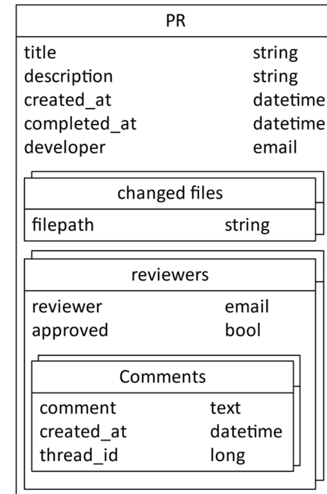

FIGURE 2. A simplified overview of the data content of a PR.

not as proficient as the global ones, can promptly review minor changes, while the global experts are included only in the cases of major/critical changes. Out of the several reviewer recommendation approaches we investigated, the majority of them were focused only on small to medium sized repositories with 100-200 developers [3], [8], [9], [10], and only some of them have been tested on larger repositories with more than 200 developers [11], [12], [13].

We propose a technique for improving the performance of PR reviewer recommendation systems by helping them to focus. The technique takes the reviewer recommendations from existing systems and filters out any reviewers which are irrelevant based on certain locality criteriums (basic filter) or a combination of criteriums (composite filter). Apart from using criteriums based on the PR history of the repository, we also utilize the organizational structure of teams which, to our knowledge, hasn't been used for the reviewer recommendation problem up to this point, as most previous papers were using open-source repositories which didn't include that information. In this paper we evaluated 6 basic filters (of which 2 are from the organizational structure of the teams - managers, and colleagues reporting to the same manager), as well as all of their combinations as composite filters.

In this paper, we first present an overview of existing approaches (Section II). We then evaluate their baseline performance and propose our technique for improving it (Section III). We define research questions and metrics that will help us evaluate the benefits of our technique (Section IV), and then proceed with the evaluation itself and the interpretation of the results (Section V). Finally, we give a brief conclusion of our work and provide some ideas for additional efforts that could be done in the future (Section VI).

II. RELATED WORK

This section first briefly goes over the relevant background related to PRs by introducing the PR lifecycle and giving an overview of the data contained within a PR. Afterwards, it summarizes 7 existing reviewer recommendation approaches that are used throughout the rest of the paper.

A. BACKGROUND

The general flow of code review, which is common across tools such as Gerrit and CodeFlow [14], as well as in systems like GitHub and Azure DevOps, is illustrated in **Figure 1**. First, the developer creates a development branch from the current code in the main branch. He implements some changes and commits them to the development branch. Once he believes the changes contain the improvements he wants to make and that they are of sufficient quality, he creates a PR containing the changes, with a title and description explaining what they represent. He can also manually add reviewers to the PR (if he believes they might be interested in the PR and would review it), and/or the reviewers could be automatically added based on some simple pre-defined rules of that repository (i.e., static file to reviewer mappings). The reviewers then assess the quality of the code, leave comments suggesting some additional improvements or requesting clarifications from the PR's developer. The developer then replies to the comments and makes the required code improvements. Since usually the PR cannot be completed until at least one reviewer marks it as "approved", the process of reviewers leaving comments and the developer replying to them and making additional changes continues until the reviewers are satisfied, and the PR gets approved. Once that happens, the developer can "complete" the PR, meaning that the proposed changes are merged into the main branch, and the PR itself is closed.

When considering what data and metadata a PR contains (**Figure 2**), apart from the already mentioned title and description, it also contains timestamps of its creation and completion, as well as information on which developer

created it. Then, there is a list of files the PR changes, as well as list of reviewers which should review the PR and ultimately give their approval. Each reviewer can leave textual comments on the PR, which also hold a timestamp of their creation time, as well as an ID to the comment thread (so that the developer and other reviewers can reply to it).

Since it was shown that the assignment of reviewers to a PR can impact the review process, it is in everyone's interest to select relevant reviewers with enough expertise – for the author of the PR it means that his change would be less probable to contain bugs [5], [7] and possibly that the review process would be faster [3], [4], while the reviewers would be able to focus only on PRs that are relevant to their expertise and would not be overwhelmed with unrelated PRs. That is why dozens of papers have been investigating PRs, the ways developers approach code reviews and how they interact with each other during the review process, in order to (among other things) determine what would be the best approach for recommending reviewers.

B. EXISTING APPROACHES

In this paper we evaluated the performance of 7 existing approaches (**TABLE 1**), out of which only 3 approaches ([11], [12], and [13]) were tested on larger repositories with more than 200 develops ([3] and [10] were tested on up to 202 reviewers, which we didn't consider significantly more than 200). Of those 3, only 2 ([11] and [12]) were tested on extremely large repositors with 1000 or more developers.

Yu et al. [9] proposed a reviewer recommendation method which combines two scoring functions. The first represents each PR as a term vector, where the terms are extracted from the title, description and list of reviewers who commented on the PR, and the weight of each term is calculated via TF-IDF (term frequency-inverse document frequency). Then, for a given PR, using cosine similarity the top k most similar PRs are selected from the training corpus. The score of each reviewer is the sum of the cosine similarity of each PR and the number of comments the reviewer left on that PR. The second scoring function calculates the sum of all the comments reviewer R left on developer D's PRs, where each comment has an additional time decaying factor (so that newer comments score higher than older ones) and a locality decaying factor (so that n comments on 1 PR will score less than 1 comment on n PRs). After those two scores are calculated for each candidate reviewer, they are normalized and added together, and the candidate reviewers are selected in decreasing order of the final score.

Thongtanunam et al. [3] proposed an approach that takes each previous PRs and calculates the similarity score to the new (incoming) PR, based on 4 different file path similarities. The reviewers are scored by summing up the similarity scores for each PR assigned to them, and then ranked. The same process is repeated for each of the 4 file path similarity techniques, resulting in 4 lists of ranked reviewers. Then, using the Borda count [15], the ranks are combined, creating the final reviewer ranking which is used to recommend reviewers.

TABLE 1. Overview of existing approaches, with statistics regarding the datasets they used – the number of repositories and the distribution of repositories by the number of developers that worked on them.

Approach	repositories in dataset	
	num.	distribution by number of developers
Yu2014	10	
Th2015	4	
Ha2016	4	
Fe2017	4	
Ji2017	8	
As2019	5	
Ye2019	12	

Hannebauer et al. [12] proposed an approach which precalculates a metric that denotes the expertise level of a reviewer for a given file up to a certain point in time, and then calculates the final score for each candidate reviewer using the precalculated metric in combination to the file path similarity of the files in the target PR and every other file previously reviewed by the candidate reviewer.

Fejzer et al. [10] proposed an approach that recommends relevant reviewers based on a function of the file paths they had previously reviewed. For each PR, a multiset is constructed using the path segments from the individual file paths of the changed files. A given reviewer is described by the union of the PR multisets which were assigned to him. For a new PR, its path segment multiset is extracted and compared to the multisets of all the reviewers using the Tversky index. The authors considered a couple of modifications by adding

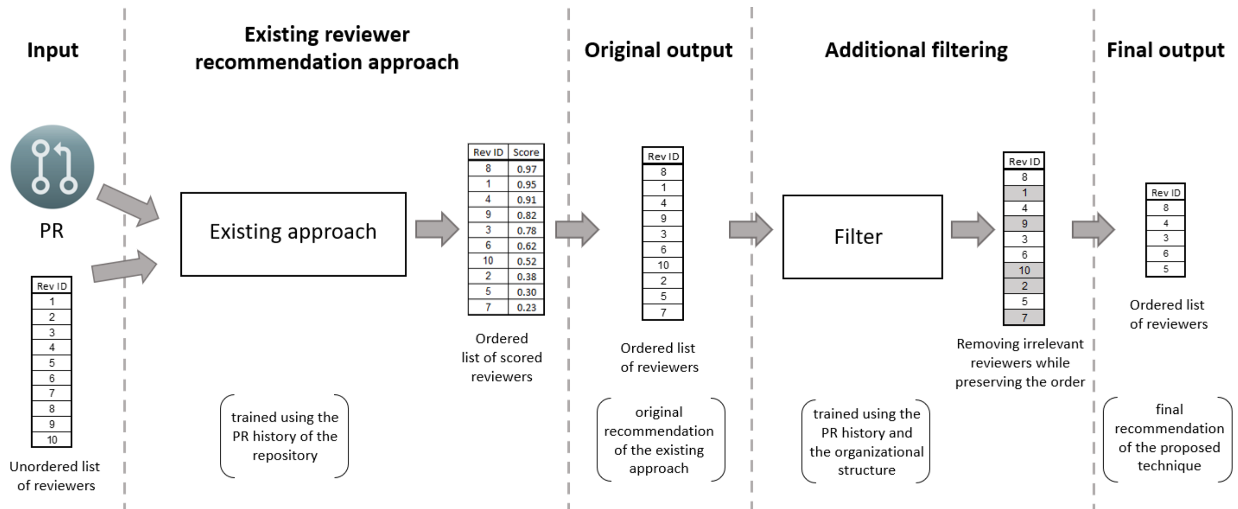

FIGURE 3. Diagram showing the overview of the proposed technique. The existing approach determines how relevant each reviewer (represented here by a reviewer ID) is for a given PR, assigns them appropriate scores, and outputs a list of reviewers ordered by their scores. Then, the filter removes any reviewers from the list that do not match it's filtering criterium, producing the final recommendation of reviewers.

different extinguishing factors (by date or PR id) but concluded that the best performance is achieved without them.

Jiang et al. [11] investigated several approaches for recommending reviewers, but the one that showed the best performance was reviewer activeness. For each candidate reviewer and for a given target PR, the activeness metric is calculated by looking at the PRs that the candidate left a comment on and which were no later than γ days before the target PR (best performance was achieved for $+\propto$) and summing up the values of a time-decaying function which gives higher scores to more recent PRs. Then, the candidate reviewers are sorted based on the activeness metric and recommended in that order.

Asthana et al. [8] proposed a scoring function that combines four time-decaying metrics. For a given PR and a candidate reviewer, the sum of products of the number of times the candidate reviewer has changed that file and an inverse time difference between the latest change and the given PR is calculated. The same is repeated for the case when the candidate reviewer reviewed the file (approved the PR or left a comment on the PR), as well as for the last-level directories for both cases (when the candidate reviewer changed or reviewed the last-level directory). Finally, the four metrics are added together using different user-defined weights, with the default value of the weights being 1.0.

Ye [13] used a different approach, extracting 14 metrics depicting the interactions between each reviewer and the PR author. The metrics ranged from simple counts of PRs where there were interactions (where the reviewer accepted the PR or left comments), to file path similarities between the files changed in the PR and the change history of each developer, etc. Some of the metrics also have time filtering - i.e., one metric is the number of PRs each developer reviewed, and another one is the number of PRs each developer reviewed in the last 30 days. After scaling each of the metrics, a learning-to-rank [16] approach was taken in order to train an SVM model to predict the rank of each candidate reviewer [17], and then select the top ranked candidates as the recommended ones.

III. PROPOSED TECHNIQUE

In order to improve the performance of an existing approach for larger repositories, we propose a technique for reducing the reviewer candidate set from every reviewer in the repository to only those reviewers who share some form of locality with either the author of the PR, or the changed files, as both were shown to be important factors for reviewers [3], [18], [19]. What we propose is relatively simple and is illustrated in **FIGURE 3**. First, an existing approach is used as-is, and for a given PR it outputs an ordered list of all the reviewers it considers relevant, from most to least relevant. Then, using a filter we filter out reviewers based on a locality criterium while preserving the ordering, so that only reviewers that have a shared locality with the PR remain in the ordered list. Finally, the top N (N denoting the number of reviewers the system is asked to recommend) reviewers are taken from that ordered list and presented as the final recommendation.

In this paper, we consider two types of filters – basic filters, which use a single criterium for filtering, and composite filters, which use multiple criteriums and represent unions of different basic filters. We propose the following basic filters:

- *File Reviewers (FR)* – this filter looks at the developers who previously reviewed the files changed in the PR, and scopes down the candidates only to those reviewer candidates. It uses the filepath and reviewer fields from **Figure 2.**
- *File Authors (FA)* – this filter looks at the developers who previously modified the files changed in the PR, and scopes down the candidates only to those reviewer candidates. It uses the filepath and developer fields from **Figure 2.**

TABLE 2. Overview of the repository datasets which were used in the evaluation.

dataset	num of PRs	num of devs	dev coverage (%)	num of revs	rev coverage (%)	num of files
A	15816	950	0.23	1532	1.13	327640
B	7663	399	0.52	585	1.45	40298
C	1930	168	0.83	264	1.94	57813
D	5011	238	0.79	385	2.01	22775
E	2465	326	0.51	679	2.94	7289
F	2000	418	0.31	682	0.49	16012
G	2251	102	1.40	174	11.69	8275
H	1318	124	1.30	157	3.55	1984

- *Previous Reviewers (PR)* – this filter scopes down the reviewer candidates only to the previous reviewers of the PR developer's code reviews. It uses the developer and reviewer fields from **Figure 2.**
- *Previous Voters (PV)* – this filter scopes down the reviewer candidates only to the previous reviewers who approved at least one of the developer's PRs. It uses the developer, reviewer and approved fields from **Figure 2**, and is always a subset of the *Previous Reviewers* filter.
- *Colleagues (CO)* – this filter scopes down the reviewer candidates only to the developer's colleagues which report to the same manager. It utilizes the organizational structure (org chart) of the teams working on the repository.
- *Manager (MN)* – this filter scopes down the reviewer candidates only to the developer's manager. It utilizes the organizational structure (org chart) of the teams working on the repository.

IV. VALIDATION SETUP

In order to assess the proposed technique, in this section we first define four research questions which will guide us in the evaluation. Then, we define the metrics which we use to measure the performance, as well as give an overview of the datasets that we use in this paper. Finally, we go over the setup of the experiments we conducted.

A. RESEARCH QUESTIONS

In order to evaluate our proposed technique, we defined the following research questions:

RQ1: Does scoping down the set of candidate reviewers using a basic filter improve the performance of existing reviewer recommendation approaches?

RQ2: Does using a union of different basic filters (a composite filter) improve the performance of existing reviewer recommendation approaches?

RQ3: What basic filters are the most important for each existing approach?

RQ4: What basic filters are the most important for each dataset?

B. METRICS

When measuring the performance of the approaches, we used the following metrics:

- *Precision@k* represents the number of correctly recommended reviewers in the top k recommendations, divided by k.
- *Recall@k* represents the number of correctly recommended reviewers in the top k recommendations, divided by the actual number of reviewers.

For each of the defined filters, we used the following null hypotheses to evaluate the improvements for each of the metrics:

- H_{0_1} : There is no statistically significant difference between the *Precision@k* of the baseline approach with and without a filter.
- H_{0_2} : There is no statistically significant difference between the *Recall@k* of the baseline approach with and without a filter.

C. DATASETS

Using CloudMine [20] we gathered the PR history from 8 internal Microsoft repositories for the calendar year 2021, whose details can be seen in **Table 2**. They are repositories of different sizes, from relatively small ones with around 100-200 active developers, over some larger ones with hundreds of developers, up to the biggest one with around 1000 developers and around 1500 reviewers. They are also different in regard to the number of files they contain, ranging from a few thousand files, up to more than 300 000 files.

What's interesting to note is that the developer coverage metric [8] (the percentage of files a single developer modified on average compared to all modified files in the dataset) seems to inversely correlate with the size of the repository in terms of number of reviewers. This seems to match the findings that Asthana et al. had, where they noticed that larger repositories tended to have more specialized developers.

D. EXPERIMENTS

In order to evaluate them, we implemented each of the existing approaches based on the descriptions from their respective papers. For each experiment we ran, we split each dataset into two equal parts, where only the first one is used for training the approaches and filters, while the second one is used for validation. After training the existing approaches as well as the filters on the training dataset, for each PR we used the existing approaches to generate a sorted list of reviewers from most to least relevant to the PR. Then, different filters generate sets of reviewers that matched their filtering criteria for the given PR, and remove any reviewers that are not present in their generated sets from the sorted reviewer list, and output those modified reviewer lists.

Finally, the precision and recall metrics are calculated by comparing the first k reviewers from the modified reviewer lists and checking if they were actually present on the given PR. In order to measure the improvements of the proposed filters, their metrics are compared to the metrics obtained from the original (unmodified) reviewer output by the existing approaches. To validate that the

TABLE 3. Overview of all the experimental results for RQ1 ((a) precision and (b) recall) and only the best performing filter techniques by existing approach for RQ2 ((c) precision and (d) recall).

(a) RQ1 - precision

Technique	Yu2014	Th2015	Ha2016	Fe2017	Ji2017	As2019	Ye2019
Manager	-7.81	-21.26	-18.44	7.49	-5.44	-46.54	-27.60
Colleagues	-3.58	-12.02	-9.24	12.79	3.02	-34.56	-17.27
PreviousVoters	5.83	-3.48	-1.08	**<u>16.24</u>**	11.12	-24.01	-8.33
PreviousReviewers	4.42	1.17	**<u>2.89</u>**	10.29	7.31	**<u>-12.19</u>**	**<u>-1.10</u>**
FileAuthors	0.60	-9.56	-7.22	11.07	4.93	-31.19	-14.98
FileReviewers	**<u>7.45</u>**	**<u>1.34</u>**	1.96	11.03	**<u>11.12</u>**	-12.34	-2.74

(b) RQ1 - recall

Technique	Yu2014	Th2015	Ha2016	Fe2017	Ji2017	As2019	Ye2019
Manager	-10.55	-21.48	-19.86	5.68	-8.29	-55.44	-33.17
Colleagues	-6.91	-12.80	-11.18	12.09	-0.31	-44.43	-23.21
PreviousVoters	3.24	-0.43	0.91	**<u>19.66</u>**	**<u>11.35</u>**	-30.00	-10.43
PreviousReviewers	**<u>4.77</u>**	**<u>5.86</u>**	**<u>6.81</u>**	15.75	11.12	**<u>-14.90</u>**	**<u>-0.13</u>**
FileAuthors	-1.90	-8.46	-7.23	12.44	3.75	-38.82	-18.89
FileReviewers	4.52	3.11	3.34	14.11	9.84	-18.70	-4.87

(c) RQ2 – best precision

Approach	Techniques						Results		
	FR	FA	PR	PV	CO	MA	Datasets w/ imp	Prec Imp	Rec Imp
Yu 2014	1	0	0	1	1	1	A, B, C, D, E, G	9.49	7.32
Th2015	1	1	0	1	1	1	A, B, C, D, E, F, G	5.34	8.87
Ha2016	0	1	1	b	1	1	A, B, C, D, E, F	6.34	9.98
Fe2017	0	0	0	1	0	1	A, B, C, D, E, F, G, H	17.60	21.17
Ji2017	0	1	0	1	1	1	A, B, C, D, E, F, G	15.39	16.57
As2019	1	1	1	b	1	1		-2.57	-3.83
Ye2019	0	1	1	b	1	1	A, B, C, D, E, F	3.05	4.48

(d) RQ2 – best recall

Approach	Techniques						Results		
	FR	FA	PR	PV	CO	MA	Datasets w/ imp	Prec Imp	Rec Imp
Yu 2014	1	0	0	1	1	1	A, B, C, D, E, G	9.49	7.32
Th2015	0	1	1	b	1	1	A, B, C, D, E, F, G	4.84	9.21
Ha2016	0	1	1	b	1	1	A, B, C, D, E, F	6.34	9.98
Fe2017	0	0	0	1	1	1	A, B, C, D, E, F, G, H	17.10	21.23
Ji2017	0	1	0	1	1	1	A, B, C, D, E, F, G	15.39	16.57
As2019	1	1	1	b	1	1		-2.57	-3.83
Ye2019	0	1	1	b	1	1	A, B, C, D, E, F	3.05	4.48

performance improvements are statistically significant, we used a one-sided Student's t-test and rejected the appropriate null hypothesis if the p-value $p < 0.05$ and t-value $t < t_{crit}$.

V. RESULTS

This section contains the results that were obtained from the conducted experiments per each research question, a short discussion regarding the overall results and conclusions, and an overview of threats to their validity.

A. RQ1

The results summarized by each approach for the RQ1 validation are listed in **Table 3**, while a more detailed overview of results is shown in Table 5. As we can see in **Table 3**, some basic filters like *Manager* do not result in statistically significant improvements in the majority of cases. Although [8] and [13] did not experience any improvements and instead only experienced a decrease in performance, other approaches experienced some benefits.

When looking at the precision improvements, *FileReviewers* achieved the best average improvements for 3 out of the 7 existing approaches, while *PreviousVoters* and *PreviousReviewers* showed the best improvements on one approach each. On the other hand, the greatest recall improvements were achieved by *PreviousReviewers* on 3 out of 7 approaches, and *PreviousVoters* on 2 out of 7 approaches. Overall, the best improvements over all datasets that we measured were a 16.24% improvement in precision and 19.66% improvements in recall for [10] with *PreviousVoters*.

One thing worth noting is that, as suspected, some filters might be too restrictive – for example, *Managers* will only allow 1 reviewer candidate to pass, so since the performance we measured is for the top 1-10 suggestions, the overall performance of *Managers* will not be very good.

B. RQ2

For RQ2 we checked all different unions of the filters, and the complete results we obtained are provided in **Table 6**, while a summarized version showing only the best results per approach is shown in **Table 3**. Again, [8] did not experience any improvements, however, we can see that for all other existing approaches the union of filters resulted in additional improvements, even in the case of [13] which initially didn't benefit from basic filters in RQ1. Overall the best improvements were achieved again for [10], with 17.60% better precision for the union of *PreviousVoters* and *Manager* filters, and 21.23% better recall for the union of *PreviousVoters, Colleagues* and *Manager* filters.

Looking at each existing approach, the best performance of the unions of filters outperforms the basic filters, even in the case of [8], were the regression was much smaller than previously recorded. This means that our assumption that certain filters did contain relevant reviewers but were too restrictive by themselves was valid, and that additional performance benefits can be achieved by combining different basic filters.

It's interesting to note that there are some ties for the best performing composite filers for some approaches. This happens in cases when the *PreviousReviewers* filter is included in the composite filter. Since *PreviousVoters* is always a subset of *PreviousReviewers*, if the *PreviousReviewers* filter is already a part of the composite filter, then the *PreviousVoters* filter will not contribute anything to the composite filter, and as such will not affect the performance.

C. RQ3

In order to determine the influence a basic filter has on the total improvements of a composite filter, we first look at a composite filter that contains the given basic filter and at least one other basic filter. Then, we find the composite filter which consists of the same set of basic filters minus

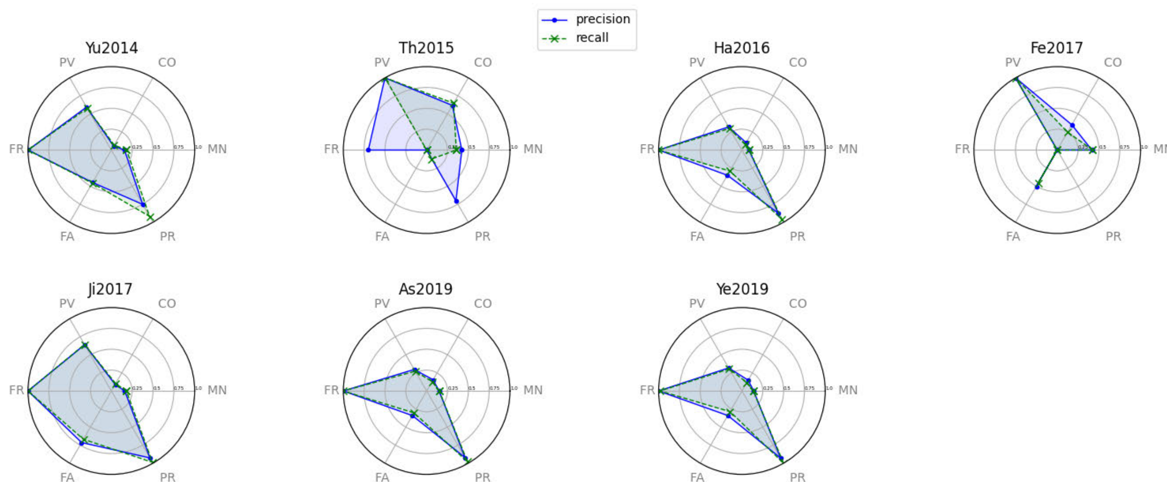

FIGURE 4. Radar charts showing the normalized impact each basic filter has when combining with other filters, per each approach.

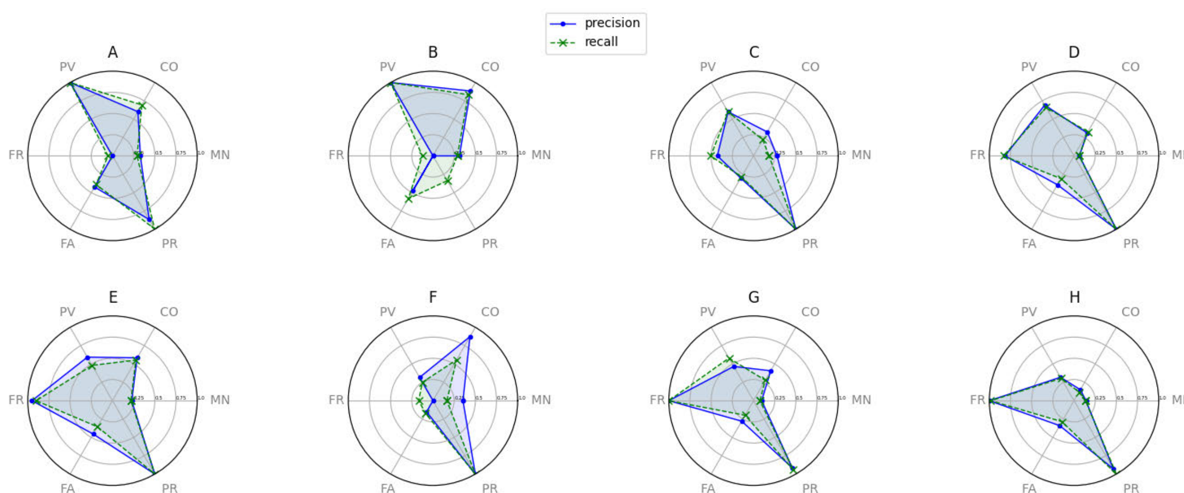

FIGURE 5. Radar charts showing the normalized impact each basic filter has when combining with other filters, per each dataset.

the given one. By subtracting the performance metrics of these two composite filters, we can measure how big are the contributions of the given filter. Repeating this procedure for each composite filter that contains the given filter and averaging the differences in performance, we can calculate its average contributions. This process is repeated for every basic filter, and the average contributions are normalized per approach.

When looking at **FIGURE 4**, we can see that each approach to some extent benefits from *Previous Voters*, with [3] and [10] having this filter as the dominant one. This is somewhat expected, as [3] and [10] do not consider reviewer activeness (the reviewer approving the PR or commenting on it), while all other approaches except [12] consider it, and the approaches that had the most benefits from a basic filter which filters out passive reviewers (reviewers which are only assigned to a PR but did not perform any action) are the approaches that do not consider reviewer activeness. The only anomaly is [12], which doesn't consider reviewer activeness, but also doesn't have a dominant *Previous Voters* filter.

Since [13] scores the reviewers by combining multiple functions, it can happen that a reviewer has a non-zero score even though he and the developer had no previous interactions. We can see that filtering out such cases has a positive effect on [13] (*Previous Reviewers*) and that a similar cutoff criterium for *File Reviewers* is also welcomed.

The *Manager* filter does help a bit in all the cases, but it's far from bringing major improvements in any of them. Although the *Colleagues* filter bears no significance in many of the approaches, it is relatively important for [3] and somewhat for [10], suggesting that maybe they were recommending global experts instead of local ones. While *File Reviewers* and *Previous Reviewers* were relatively important for every other approach, they didn't help [10], in a similar way [3] seems to be an exception for *File Authors*, but we are unsure why this occurred.

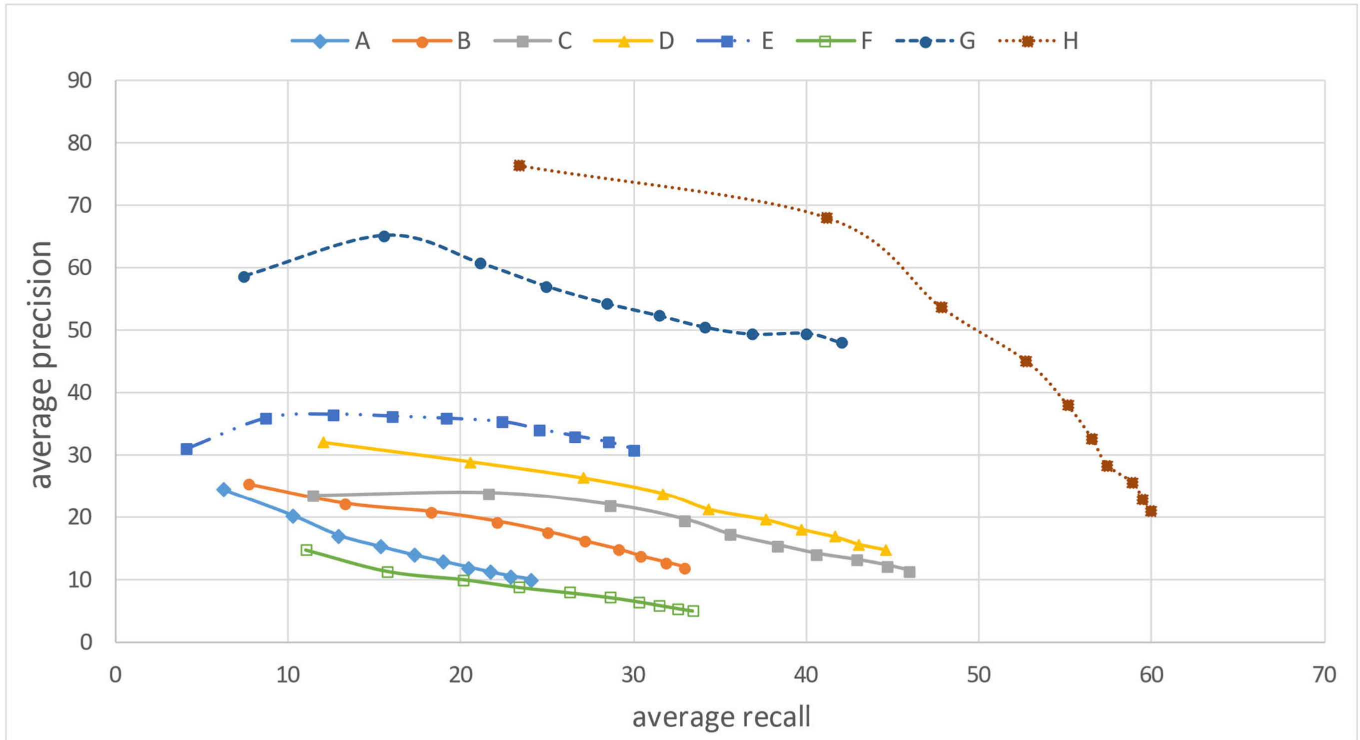

FIGURE 6. Precision-recall performance of all existing approaches averaged per dataset (A-H) for the top 10 recommendations.

TABLE 4. Maximum precision (a) and recall (b) improvements that any filter achieved per each combination of dataset and existing approach.

a) max precision

	Yu2014	Th2015	Ha2016	Fe2017	Ji2017	As2019	Ye2019
A	7.73	5.53	7.98	18.27	17.74	-0.31	6.54
B	8.25	13.31	15.92	18.03	16.57	-0.99	9.06
C	9.06	8.92	10.89	13.62	16.10	-3.56	3.08
D	12.71	3.87	5.87	13.82	15.10	-1.34	1.05
E	20.79	16.04	26.37	24.13	28.59	-3.90	13.00
F	0.48	9.12	8.77	8.19	7.35	-2.11	6.83
G	23.35	6.45	3.72	42.48	43.28	-2.55	0.01
H	2.33	-4.59	-4.10	18.70	-1.71	-5.44	-3.74

b) max recall

	Yu2014	Th2015	Ha2016	Fe2017	Ji2017	As2019	Ye2019
A	7.19	11.45	12.54	21.71	21.33	0.51	8.51
B	10.03	19.26	23.25	26.63	26.33	-2.43	10.63
C	14.15	19.11	21.48	30.94	30.11	-7.57	5.10
D	17.48	7.25	8.66	24.13	23.47	-2.62	1.50
E	8.58	12.50	16.95	15.46	15.63	-3.56	10.33
F	0.75	27.19	25.19	27.21	21.71	-5.16	13.56
G	7.52	8.20	5.02	19.74	15.70	-1.23	1.00
H	0.09	-5.29	-4.43	25.30	-1.18	-6.92	-4.28

D. RQ4

For RQ4, a similar aggregation of average contributions was done but per dataset (**FIGURE 5**). First off, we can see that *Previous Voters* contributes to each dataset, and additionally it seems to be more important for the larger datasets (A, B, C, D, E) than the smaller ones. Interestingly, some datasets (B, F, and to some extent A, E) seem to operate in teams that are based on hierarchy (with coworkers reporting to the same manager – *Colleagues*), while others still cooperate with coworkers under the same manager, but they also have strong collaborations with other developers as well. Moreover, we can see that some datasets (A, B, C, E, F, and somewhat H) have clear benefits from the *Manager* filter, while others don't (D, G). We believe that the teams from the first group might have a rule of thumb to include the developer's manager on PRs, while in other cases this isn't as common. We can see that *Previous Reviewers* plays a major role in all datasets except B. We are unsure why dataset B is an exception, but in general *Previous Reviewers* is expected to be impactful, as collaborations that existed in the past are more likely to continue in the future.

Some of the datasets (D, E, G, H) have a prominent *File Reviewer*, which might indicate that they have clearly defined code components and file ownerships. This would mean that, while anyone can change a given file, there is a designed part of the team or person that must review and signoff the changes to that file. On the other hand, some datasets (A, B, E) have strong *File Authors*, which could indicate that there is a strong task/feature componentization, where work is organized so that a part of the team works on a given feature and changes files across multiple modules. Due to that, the developers that work on the same feature are reviewers of

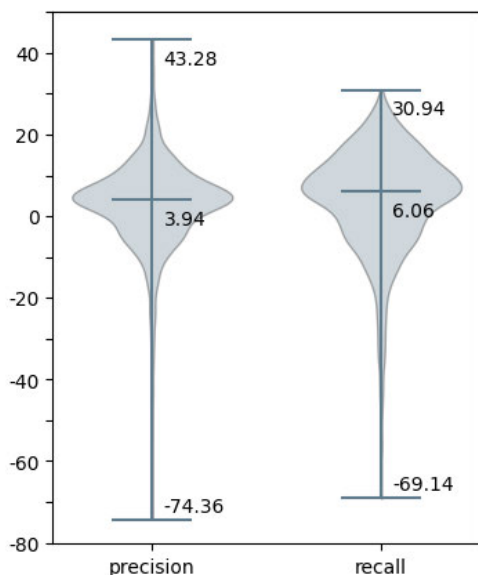

FIGURE 7. Violin plots for the precision and recall improvement obtained in RQ2, with highlighted extrema and median values.

each other's PRs, but there isn't necessarily strict code/file ownership enforcement.

E. DISCUSSION

While measuring the existing approaches for a baseline, we noticed that their performance tended to decrease with the increase of the repository sizes (**Figure 6**). This seems to support our theory that larger repositories need help to focus, as they tend to operate in clusters where groups of developers work relatively independently from each other, while still having some level of interaction between them. That is also consistent with what Asthana et al. [8] observed, where for larger repositories the majority of reviewers tend to be experts only for smaller subsets of the repository, with occasional senior reviewers whose expertise applies to a larger portion of the codebase.

When looking at the improvements we obtained, overall 5 out of the 7 existing approaches on average experienced improvements from just the basic filters (RQ1). By combining different basic filters into composite filters, apart from achieving higher performance improvements than by using just basic filters, the number of approaches that on average experienced benefits increased to 6 out of 7 (RQ2). However, the results from RQ2 show only the average improvements per composite filter and approach pairs over all datasets, which might not be ideal. As can be seen in the results of RQ3 and RQ4, both the existing approaches and the datasets are differently affected by the filters, and have different characteristics. Because of that, we think that apart from looking at the performance across the datasets, it would make sense to also include the best results (of any composite filter) per dataset per approach (**TABLE 4**).

Apart from the dataset with the smallest number of reviewers (H), the other datasets experienced improvements for all the approaches except [8]. However, even in the case of [8], we showed that it is possible to improve the performance in some cases (recall on dataset A). Additionally, with per dataset/repository tuning of the filters, we were able to achieve additional improvements compared to RQ2, which peaked at 43.28% better precision and 30.94% better recall. When looking at the violin plots of all the improvements achieved in RQ2 by all the different composite filters (**Figure 7**), we can see that that, although there are cases that had a very negative impact on the metrics, the median is positive. By testing out different filters, the one that achieves the best improvements for the given approach-repository pair can be found.

Training a given filter can be done with just one pass over the training data ($O(n)$, where n is the size of the training set) which would generate the set of reviewers that match the filters criterium. In order to test out every composite filter, one would need to train all the basic filters ($O(kn)$, where k is the number of basic filters) and then compute all their different unions in order to generate every composite filter ($O(2^k)$), resulting in $O\left(kn + 2^k\right)$ complexity of the training process. However, in order to test the impact of each composite filter on a baseline approach, every time an existing approach gives a recommendation on the validation set ($O(m)$, where m is the size of the validation set) it would need to be passed through all the filters, resulting in $O(2^k m)$ complexity for the validation.

Although it's far from being an ideal solution with guaranteed performance improvements, our technique did result in some improvements for all of the existing approaches we tested, and we believe that with a broader set of basic filters (i.e., utilizing PR titles and descriptions, content of comments, etc.), obtaining improvements would just be a matter of fine-tuning our technique for a given approach-repository pair by finding the right composite filter.

F. THREATS TO VALIDITY

One of the things that could have impacted the results of *Managers* and *Colleagues* is the fact that the snapshot of the management hierarchy was made in June 2022, as no historical data was available. Due to this, some results returned by the filters might have been inherently bad - i.e., if a developer changed teams between Jan and June 2022, then during the validation he would have been assigned to the new team instead of the old one, as the PR datasets were collected for 2021.

Additionally, although the authors tried to focus only on individual identities for the reviewer recommendation problem and tried to remove any accounts associated with automation or any distribution lists, there is no guarantee that the datasets have been completely cleaned. If a team had implemented a custom automation account that would always be added to their pull requests and it slipped into the datasets, then an account like that would have been a safe bet by any reviewer recommendation approach and could have influenced their performance metrics.

TABLE 5. The extended results of the RQ1 experiment, showing the obtained improvements for each basic filter, dataset and existing approach, as well as if the appropriate null-hypothesis (H_{0_1} for precision and H_{0_2} for recall) was accepted (shown as grayed out, meaning that there were no statistically significant improvements for those cases), or rejected (clear cells).

Technique	Dataset	Yu2014		Th2015		Ha2016		Fe2017		Ji2017		As2019		Ye2019	
Manager	A	H0_1	-5.04	H0_1	-12.1196	H0_1	-8.66997	H0_1	8.128985	H0_1	3.671787	H0_1	-30.0735	H0_1	-16.9369
		H0_2	-6.50	H0_2	-10.3298	H0_2	-8.68388	H0_2	6.213485	H0_2	1.935727	H0_2	-33.2529	H0_2	-19.9512
	B	H0_1	-5.39	H0_1	-6.7076	H0_1	-3.56403	H0_1	6.373734	H0_1	-0.30032	H0_1	-38.7743	H0_1	-14.8561
		H0_2	-8.29337	H0_2	-11.8138	H0_2	-7.55859	H0_2	6.19002	H0_2	-1.5447	H0_2	-55.4561	H0_2	-23.9796
	C	H0_1	-1.21687	H0_1	-6.19724	H0_1	-3.25762	H0_1	7.608895	H0_1	2.644827	H0_1	-36.5071	H0_1	-16.0029
		H0_2	-7.75304	H0_2	-17.2547	H0_2	-13.9224	H0_2	10.62109	H0_2	-2.25769	H0_2	-69.1443	H0_2	-38.4947
	D	H0_1	-5.08698	H0_1	-18.6307	H0_1	-15.4865	H0_1	3.957915	H0_1	-2.23994	H0_1	-45.2572	H0_1	-28.7773
		H0_2	-12.262	H0_2	-29.809	H0_2	-26.8159	H0_2	4.056152	H0_2	-7.42849	H0_2	-67.4359	H0_2	-46.5103
	E	H0_1	-7.11941	H0_1	-25.9253	H0_1	-11.984	H0_1	10.32414	H0_1	-5.39872	H0_1	-74.3577	H0_1	-30.1248
		H0_2	-5.84899	H0_2	-13.9542	H0_2	-7.02199	H0_2	3.678442	H0_2	-4.45734	H0_2	-49.5871	H0_2	-16.7371
	F	H0_1	-1.04284	H0_1	0.586754	H0_1	0.05898	H0_1	1.550758	H0_1	0.752742	H0_1	-22.0739	H0_1	-5.53086
		H0_2	-3.90802	H0_2	-6.38162	H0_2	-8.91876	H0_2	0.595567	H0_2	-2.63671	H0_2	-60.1981	H0_2	-29.2111
	G	H0_1	-19.982	H0_1	-56.7749	H0_1	-60.6903	H0_1	13.7327	H0_1	-12.8945	H0_1	-74.3082	H0_1	-65.28
		H0_2	-15.8246	H0_2	-25.5212	H0_2	-29.8007	H0_2	4.405453	H0_2	-15.227	H0_2	-42.132	H0_2	-34.7733
	H	H0_1	-17.6145	H0_1	-44.2774	H0_1	-43.9304	H0_1	8.245207	H0_1	-29.7271	H0_1	-50.9853	H0_1	-43.3257
		H0_2	-23.9718	H0_2	-56.8156	H0_2	-56.1793	H0_2	9.658501	H0_2	-34.7285	H0_2	-66.3074	H0_2	-55.7103
Colleagues	A	H0_1	-2.28469	H0_1	-6.94263	H0_1	-3.69698	H0_1	10.4113	H0_1	8.25573	H0_1	-20.5281	H0_1	-9.35476
		H0_2	-3.75861	H0_2	-3.20331	H0_2	-1.6538	H0_2	11.6564	H0_2	8.873079	H0_2	-22.6009	H0_2	-10.5507
	B	H0_1	0.323793	H0_1	4.662848	H0_1	7.713089	H0_1	11.33396	H0_1	10.57157	H0_1	-23.3609	H0_1	-2.13434
		H0_2	-2.14379	H0_2	3.785074	H0_2	8.033109	H0_2	17.50258	H0_2	13.0808	H0_2	-37.28	H0_2	-7.41545
	C	H0_1	-0.28821	H0_1	-3.66753	H0_1	-0.7659	H0_1	7.039806	H0_1	4.691299	H0_1	-32.2438	H0_1	-12.8482
		H0_2	-5.99012	H0_2	-13.835	H0_2	-10.4875	H0_2	12.02365	H0_2	0.982997	H0_2	-64.2695	H0_2	-34.1762
	D	H0_1	4.563348	H0_1	-6.15958	H0_1	-3.05716	H0_1	9.752426	H0_1	9.847786	H0_1	-29.5273	H0_1	-15.9056
		H0_2	2.323883	H0_2	-10.8036	H0_2	-7.78742	H0_2	17.87751	H0_2	11.7133	H0_2	-45.4592	H0_2	-26.6137
	E	H0_1	0.233535	H0_1	-12.2605	H0_1	1.688893	H0_1	18.29618	H0_1	8.067261	H0_1	-58.2772	H0_1	-15.596
		H0_2	-3.06683	H0_2	-7.41811	H0_2	-0.48533	H0_2	9.457216	H0_2	1.979876	H0_2	-41.9361	H0_2	-9.54813
	F	H0_1	-1.07284	H0_1	3.164024	H0_1	2.712417	H0_1	3.196829	H0_1	3.265032	H0_1	-16.1621	H0_1	-1.74248
		H0_2	-3.85802	H0_2	0.969298	H0_2	-1.47785	H0_2	6.893996	H0_2	3.801548	H0_2	-46.8705	H0_2	-18.0367
	G	H0_1	-8.6601	H0_1	-32.7756	H0_1	-36.6688	H0_1	34.05813	H0_1	11.3732	H0_1	-47.1428	H0_1	-38.665
		H0_2	-9.2977	H0_2	-14.7368	H0_2	-18.9998	H0_2	13.4894	H0_2	-4.80769	H0_2	-30.2094	H0_2	-22.9951
	H	H0_1	-21.4683	H0_1	-42.2133	H0_1	-41.8689	H0_1	8.196678	H0_1	-31.9028	H0_1	-49.2504	H0_1	-41.9203
		H0_2	-29.4983	H0_2	-57.1892	H0_2	-56.5556	H0_2	7.803493	H0_2	-38.1259	H0_2	-66.7985	H0_2	-56.3162
Previous Voters	A	H0_1	5.152865	H0_1	1.286369	H0_1	3.920999	H0_1	16.63063	H0_1	14.32537	H0_1	-10.6308	H0_1	0.758023
		H0_2	3.979282	H0_2	6.85662	H0_2	8.028794	H0_2	20.20848	H0_2	17.23986	H0_2	-10.8265	H0_2	1.19816
	B	H0_1	6.742907	H0_1	9.530955	H0_1	12.11547	H0_1	17.75781	H0_1	13.80636	H0_1	-13.8561	H0_1	4.482751
		H0_2	7.326855	H0_2	13.30557	H0_2	17.23876	H0_2	26.18252	H0_2	21.31599	H0_2	-22.5503	H0_2	3.794474
	C	H0_1	6.277979	H0_1	4.399737	H0_1	6.785204	H0_1	12.06246	H0_1	13.02845	H0_1	-17.586	H0_1	-3.43929
		H0_2	7.273501	H0_2	9.497918	H0_2	12.36362	H0_2	28.63499	H0_2	23.52309	H0_2	-34.2771	H0_2	-9.15313
	D	H0_1	11.09344	H0_1	1.075421	H0_1	3.388225	H0_1	13.81518	H0_1	13.08778	H0_1	-14.2139	H0_1	-4.54439
		H0_2	14.38193	H0_2	1.962553	H0_2	4.121675	H0_2	23.71394	H0_2	19.55654	H0_2	-23.3227	H0_2	-9.0586
	E	H0_1	12.17484	H0_1	-2.74355	H0_1	10.41595	H0_1	22.00041	H0_1	16.9914	H0_1	-47.6715	H0_1	-5.95444
		H0_2	2.276207	H0_2	-1.82215	H0_2	4.771375	H0_2	11.81569	H0_2	7.35173	H0_2	-36.1789	H0_2	-4.43767
	F	H0_1	-0.69208	H0_1	2.177429	H0_1	1.692155	H0_1	1.516925	H0_1	3.024333	H0_1	-20.6358	H0_1	-4.2771
		H0_2	-2.57874	H0_2	3.824929	H0_2	1.342786	H0_2	6.052115	H0_2	6.95769	H0_2	-50.135	H0_2	-19.3516
	G	H0_1	11.38544	H0_1	-17.5055	H0_1	-21.2553	H0_1	29.75491	H0_1	26.93727	H0_1	-33.6142	H0_1	-26.3765
		H0_2	4.001855	H0_2	-1.25569	H0_2	-5.44224	H0_2	18.19095	H0_2	9.535841	H0_2	-16.3645	H0_2	-10.0165
	H	H0_1	-5.47778	H0_1	-26.0929	H0_1	-25.7042	H0_1	16.36244	H0_1	-12.2563	H0_1	-33.8724	H0_1	-27.2512
		H0_2	-10.7232	H0_2	-35.7967	H0_2	-35.1081	H0_2	22.50113	H0_2	-14.6885	H0_2	-46.3527	H0_2	-36.4255
Previous Reviewers	A	H0_1	6.642454	H0_1	1.462435	H0_1	2.702929	H0_1	11.99872	H0_1	10.99935	H0_1	-4.98783	H0_1	2.806506
		H0_2	6.216004	H0_2	4.690161	H0_2	4.838104	H0_2	15.59444	H0_2	13.76049	H0_2	-4.97938	H0_2	4.687447
	B	H0_1	6.117625	H0_1	3.77934	H0_1	5.165336	H0_1	9.276382	H0_1	7.049959	H0_1	-8.36303	H0_1	2.192613
		H0_2	7.977291	H0_2	6.914368	H0_2	9.384031	H0_2	15.1942	H0_2	12.7497	H0_2	-13.0017	H0_2	3.881255
	C	H0_1	4.452957	H0_1	6.602295	H0_1	8.789662	H0_1	7.558286	H0_1	9.231734	H0_1	-10.575	H0_1	0.789217
		H0_2	9.295977	H0_2	15.5283	H0_2	18.31232	H0_2	21.56282	H0_2	23.12181	H0_2	-20.3691	H0_2	0.580007
	D	H0_1	6.261124	H0_1	1.887708	H0_1	3.317807	H0_1	6.098761	H0_1	5.735882	H0_1	-8.25669	H0_1	-1.09484
		H0_2	12.0337	H0_2	4.507276	H0_2	5.965537	H0_2	11.58572	H0_2	12.85877	H0_2	-13.0409	H0_2	-2.508
	E	H0_1	8.799418	H0_1	7.011232	H0_1	17.01542	H0_1	16.30888	H0_1	11.71122	H0_1	-24.5433	H0_1	5.676772
		H0_2	4.598509	H0_2	6.681619	H0_2	12.01826	H0_2	13.17412	H0_2	9.323441	H0_2	-17.3696	H0_2	5.425406
	F	H0_1	0.129083	H0_1	5.423714	H0_1	5.051802	H0_1	6.982476	H0_1	4.039044	H0_1	-11.217	H0_1	1.68858
		H0_2	-0.12	H0_2	19.37721	H0_2	17.19688	H0_2	24.40004	H0_2	16.1019	H0_2	-24.8327	H0_2	1.870275
	G	H0_1	5.080937	H0_1	-5.45701	H0_1	-7.99824	H0_1	13.47614	H0_1	16.72218	H0_1	-13.9804	H0_1	-9.76019
		H0_2	2.94926	H0_2	2.502144	H0_2	-0.64564	H0_2	9.311876	H0_2	8.267511	H0_2	-5.66873	H0_2	-2.24552
	H	H0_1	-2.11165	H0_1	-11.3606	H0_1	-10.946	H0_1	10.59136	H0_1	-7.02828	H0_1	-15.622	H0_1	-11.0873
		H0_2	-4.77558	H0_2	-13.3566	H0_2	-12.5876	H0_2	15.20369	H0_2	-7.26176	H0_2	-19.977	H0_2	-12.6933
File Authors	A	H0_1	1.999413	H0_1	-4.42078	H0_1	-2.20887	H0_1	4.368679	H0_1	6.010401	H0_1	-16.4602	H0_1	-3.96484
		H0_2	1.26625	H0_2	-1.63222	H0_2	-0.72206	H0_2	6.511313	H0_2	7.672646	H0_2	-19.5975	H0_2	-5.86233
	B	H0_1	3.722298	H0_1	5.010642	H0_1	7.068731	H0_1	9.671556	H0_1	10.13634	H0_1	-16.6441	H0_1	-1.40894
		H0_2	3.599121	H0_2	5.871842	H0_2	8.95571	H0_2	14.89856	H0_2	14.281	H0_2	-27.9641	H0_2	-4.41909
	C	H0_1	3.408344	H0_1	-0.08445	H0_1	2.393034	H0_1	9.323193	H0_1	8.501077	H0_1	-27.2956	H0_1	-9.54596
		H0_2	1.358937	H0_2	-3.93232	H0_2	-1.12976	H0_2	18.74539	H0_2	11.09054	H0_2	-52.1818	H0_2	-24.7037
	D	H0_1	7.561091	H0_1	-3.58461	H0_1	-1.04056	H0_1	7.739763	H0_1	10.71047	H0_1	-24.1236	H0_1	-11.6215
		H0_2	5.835263	H0_2	-8.18414	H0_2	-5.62578	H0_2	13.66739	H0_2	12.29336	H0_2	-40.7295	H0_2	-22.8081
	E	H0_1	5.015876	H0_1	-10.1684	H0_1	3.533629	H0_1	16.09848	H0_1	8.068535	H0_1	-56.6579	H0_1	-14.7525
		H0_2	-0.83043	H0_2	-6.29522	H0_2	0.521243	H0_2	8.248486	H0_2	2.816105	H0_2	-41.3371	H0_2	-9.29256
	F	H0_1	-0.4606	H0_1	1.906563	H0_1	1.266948	H0_1	2.327409	H0_1	2.440179	H0_1	-18.3053	H0_1	-3.36993
		H0_2	-2.0524	H0_2	3.278333	H0_2	0.617262	H0_2	8.482877	H0_2	5.587	H0_2	-46.0929	H0_2	-17.3959
	G	H0_1	-6.71006	H0_1	-36.1369	H0_1	-40.0339	H0_1	25.54758	H0_1	8.121437	H0_1	-53.5011	H0_1	-45.1151
		H0_2	-8.38238	H0_2	-16.0354	H0_2	-20.2742	H0_2	12.10692	H0_2	-5.17392	H0_2	-32.3219	H0_2	-25.1289
	H	H0_1	-9.70508	H0_1	-29.0359	H0_1	-28.7014	H0_1	13.48374	H0_1	-14.5661	H0_1	-36.5406	H0_1	-30.0643
		H0_2	-15.9715	H0_2	-40.7667	H0_2	-40.1631	H0_2	16.88728	H0_2	-18.5791	H0_2	-50.3479	H0_2	-41.4848
File Reviewers	A	H0_1	3.08834	H0_1	1.43981	H0_1	1.583316	H0_1	3.957447	H0_1	3.742424	H0_1	-4.95386	H0_1	0.687865
		H0_2	3.455396	H0_2	2.410518	H0_2	2.135754	H0_2	6.36627	H0_2	5.456127	H0_2	-8.10964	H0_2	0.884755
	B	H0_1	4.033442	H0_1	2.926607	H0_1	4.256413	H0_1	6.803662	H0_1	6.103637	H0_1	-10.1535	H0_1	0.372054
		H0_2	5.048899	H0_2	4.721608	H0_2	6.768908	H0_2	11.41399	H0_2	10.53403	H0_2	-17.5964	H0_2	-0.46065
	C	H0_1	6.600049	H0_1	4.738926	H0_1	6.357061	H0_1	10.33321	H0_1	11.31591	H0_1	-15.185	H0_1	-3.54905
		H0_2	8.382808	H0_2	9.571788	H0_2	11.06351	H0_2	23.71211	H0_2	19.88215	H0_2	-29.4114	H0_2	-9.4553
	D	H0_1	9.587226	H0_1	1.363081	H0_1	1.999751	H0_1	7.882607	H0_1	9.856269	H0_1	-8.90303	H0_1	-2.60436
		H0_2	13.69795	H0_2	2.459764	H0_2	3.094913	H0_2	15.08914	H0_2	15.68303	H0_2	-17.0364	H0_2	-6.12655
	E	H0_1	19.65771	H0_1	11.54141	H0_1	15.59308	H0_1	14.26302	H0_1	25.02299	H0_1	-14.4649	H0_1	6.958955
		H0_2	7.574547	H0_2	8.908212	H0_2	11.41161	H0_2	11.20519	H0_2	13.41521	H0_2	-12.6396	H0_2	5.588128
	F	H0_1	-0.11764	H0_1	2.44731	H0_1	2.009286	H0_1	3.448948	H0_1	3.189671	H0_1	-13.9073	H0_1	-2.14385
		H0_2	-0.99357	H0_2	7.332091	H0_2	5.320349	H0_2	10.99849	H0_2	9.88931	H0_2	-35.582	H0_2	-11.1737
	G	H0_1	18.22132	H0_1	-3.03828	H0_1	-6.02744	H0_1	29.11542	H0_1	36.63146	H0_1	-15.9186	H0_1	-10.7879
		H0_2	4.817544	H0_2	4.009643	H0_2	0.625782	H0_2	17.18418	H0_2	12.52665	H0_2	-8.00656	H0_2	-3.55959
	H	H0_1	-1.45764	H0_1	-10.6861	H0_1	-10.1197	H0_1	12.45918	H0_1	-6.91106	H0_1	-15.2073	H0_1	-10.8195
		H0_2	-5.85587	H0_2	-14.5556	H0_2	-13.7039	H0_2	16.95026	H0_2	-8.69788	H0_2	-21.2492	H0_2	-14.65

TABLE 6. The extended results of the RQ2 experiment, where for each union of techniques and for each existing approach, the datasets that experienced improvements are shown, along with the average improvements in precision and recall that were achieved compared to the baseline approach with no filters applied. The grayed-out cells are cases which experienced a negative impact on their performance.

Techniques						Yu 2014			Th2015			Ha2016			Fe2017			Ji2017			As2019			Ye2019		
FR	FA	PR	PV	CO	MA	Datasets w/ imp	Prec Imp	Rec Imp	Datasets w/ imp	Prec Imp	Rec Imp	Datasets w/ imp	Prec Imp	Rec Imp	Datasets w/ imp	Prec Imp	Rec Imp	Datasets w/ imp	Prec Imp	Rec Imp	Datasets w/ imp	Prec Imp	Rec Imp	Datasets w/ imp	Prec Imp	Rec Imp
0	0	0	0	0	1		-7.81	-10.55		-21.26	-21.49		-18.44	-19.86	A, B, C, D, E, G, H	7.49	5.68		-5.44	-8.29		-46.54	-55.44		-27.6043	-33.171
0	0	0	0	1	0		-3.58	-6.91	B	-12.02	-12.80	B	-9.24	-11.18	A, B, C, D, E, F, G, H	12.79	12.09	A, B, D, F	3.02	-0.31		-34.56	-44.43		-17.2708	-23.2065
0	0	0	0	1	1		-0.95	-4.10	B, F	-7.46	-7.11	A, B, F	-4.67	-5.49	A, B, C, D, E, F, G, H	15.46	16.23	A, B, C, D, E, F	7.01	4.72		-29.39	-38.07		-12.6008	-17.3378
0	0	0	1	0	0	A, B, C, D, E	5.83	3.24	B, C, F	-3.48	-0.43	A, B, C, D, E	-1.08	0.91	A, B, C, D, E, F, G, H	16.24	19.66	A, B, C, D, E, F, G	11.12	11.35		-24.01	-30.00	B	-8.32527	-10.4313
0	0	0	1	0	1	A, B, C, D, E	6.49	3.80	A, B, C, D, F	-1.43	1.61	A, B, C, D, E, F	0.98	2.97	A, B, C, D, E, F, G, H	**17.60**	21.17	A, B, C, D, E, F, G	12.56	12.82		-21.38	-27.42	A, B	-5.90161	-8.01374
0	0	0	1	1	0	A, B, C, D, E	7.17	4.54	A, B, C, D, E, F	0.86	4.15	A, B, C, D, E, F	3.25	5.48	A, B, C, D, E, F, G, H	16.67	20.35	A, B, C, D, E, F, G	14.36	14.79		-17.10	-22.46	A, B	-3.01932	-4.45613
0	0	0	1	1	1	A, B, C, D, E	7.66	5.04	A, B, C, D, E, F	2.08	5.82	A, B, C, D, E, F	4.46	7.17	A, B, C, D, E, F, G, H	17.10	**21.23**	A, B, C, D, E, F, G	15.18	15.97		-15.37	-20.18	A, B, E	-1.67724	-2.58881
0	0	1	0	0	0	A, B, C, D, E, G	4.42	4.77	A, B, C, D, E, F	1.17	5.86	A, B, C, D, E, F	2.89	6.81	A, B, C, D, E, F, G, H	10.29	15.75	A, B, C, D, E, F, G	7.31	11.12		-12.19	-14.90	A, B, E	-1.09858	-0.1253
0	0	1	0	0	1	A, B, C, D, E, G	4.83	5.08	A, B, C, D, E, F	2.29	6.92	A, B, C, D, E, F	4.01	7.89	A, B, C, D, E, F, G, H	11.01	16.39	A, B, C, D, E, F, G	8.16	11.87		-10.69	-13.37	A, B, C, E, F	0.270213	1.226922
0	0	1	0	1	0	A, B, C, D, E, G	5.24	5.44	A, B, C, D, E, F	3.43	8.03	A, B, C, D, E, F	5.14	9.00	A, B, C, D, E, F, G, H	10.50	15.33	A, B, C, D, E, F, G	9.20	12.87		-8.34	-10.45	A, B, C, E, F	1.61622	2.867061
0	0	1	0	1	1	A, B, C, D, E, G	5.52	5.72	A, B, C, D, E, F	4.01	8.90	A, B, C, D, E, F	5.72	9.87	A, B, C, D, E, F, G, H	10.71	15.71	A, B, C, D, E, F, G	9.63	13.47		-7.40	-9.08	A, B, C, E, F	2.289762	3.883724
0	0	1	1	0	0	A, B, C, D, E, G	4.42	4.77	A, B, C, D, E, F	1.17	5.86	A, B, C, D, E, F	2.89	6.81	A, B, C, D, E, F, G, H	10.29	15.75	A, B, C, D, E, F, G	7.31	11.12		-12.19	-14.90	A, B, E	-1.09858	-0.1253
0	0	1	1	0	1	A, B, C, D, E, G	4.83	5.08	A, B, C, D, E, F	2.29	6.92	A, B, C, D, E, F	4.01	7.89	A, B, C, D, E, F, G, H	11.01	16.39	A, B, C, D, E, F, G	8.16	11.87		-10.69	-13.37	A, B, C, E, F	0.270213	1.226922
0	0	1	1	1	0	A, B, C, D, E, G	5.24	5.44	A, B, C, D, E, F	3.43	8.03	A, B, C, D, E, F	5.14	9.00	A, B, C, D, E, F, G, H	10.50	15.33	A, B, C, D, E, F, G	9.20	12.87		-8.34	-10.45	A, B, C, E, F	1.61622	2.867061
0	0	1	1	1	1	A, B, C, D, E, G	5.52	5.72	A, B, C, D, E, F	4.01	8.90	A, B, C, D, E, F	5.72	9.87	A, B, C, D, E, F, G, H	10.71	15.71	A, B, C, D, E, F, G	9.63	13.47		-7.40	-9.08	A, B, C, E, F	2.289762	3.883724
0	1	0	0	0	0	A, B, D	0.60	-1.90	B, F	-9.56	-8.46	B	-7.22	-7.23	A, B, C, D, E, F, G, H	11.07	12.44	A, B, C, D, F	4.93	3.75		-31.19	-38.82		-14.9804	-18.8869
0	1	0	0	0	1	A, B, C, D	2.49	-0.22	B, F	-5.65	-4.44	B, C, F	-3.36	-3.25	A, B, C, D, E, F, G, H	13.89	15.64	A, B, C, D, E, F	8.03	7.07		-26.58	-34.11		-10.7221	-14.4869
0	1	0	0	1	0	A, B, C, D	3.46	0.72	B, C, F	-2.84	-1.73	B, C, D, E, F	-0.53	-0.50	A, B, C, D, E, F, G, H	14.20	15.92	A, B, C, D, E, F	10.46	9.36		-21.66	-29.07	B	-7.15856	-10.7936
0	1	0	0	1	1	A, B, C, D, E	4.90	2.24	B, C, D, E, F	-0.35	1.67	A, B, C, D, E, F	1.92	2.86	A, B, C, D, E, F, G, H	15.30	17.88	A, B, C, D, E, F, G	12.39	12.09		-18.59	-24.86	A, B	-4.53871	-7.18768
0	1	0	1	0	0	A, B, C, D, E, G	8.12	6.00	B, C, D, E, F	0.81	4.13	A, B, C, D, E, F	2.81	5.11	A, B, C, D, E, F, G, H	14.17	18.47	A, B, C, D, E, F, G	13.60	14.42		-16.54	-21.47	A, B	-2.97478	-4.24588
0	1	0	1	0	1	A, B, C, D, E, G	8.49	6.32	B, C, D, E, F	1.98	5.46	A, B, C, D, E, F	3.98	6.43	A, B, C, D, E, F, G, H	14.76	19.25	A, B, C, D, E, F, G	14.27	15.23		-14.82	-19.53	A, B, E	-1.62103	-2.67406
0	1	0	1	1	0	A, B, C, D, E, G	8.62	6.50	B, C, D, E, F	2.95	6.47	A, B, C, D, E, F	4.96	7.47	A, B, C, D, E, F, G, H	13.76	17.82	A, B, C, D, E, F, G	14.97	15.92		-12.45	-16.50	A, B, E	-0.33836	-0.97486
0	1	0	1	1	1	A, B, C, D, E, G	8.91	6.80	B, C, D, E, F	3.70	7.58	A, B, C, D, E, F	5.70	8.58	A, B, C, D, E, F, G, H	13.97	18.27	A, B, C, D, E, F, G	**15.39**	**16.57**		-11.23	-14.76	A, B, C, E, F	0.483054	0.249764
0	1	1	0	0	0	A, B, C, D, E, G	5.31	5.82	A, B, C, D, E, F	3.35	7.50	A, B, C, D, E, F	4.86	8.25	A, B, C, D, E, F, G, H	8.91	13.90	A, B, C, D, E, F, G	8.29	11.82		-8.11	-10.52	A, B, C, E, F	1.308806	2.183928
0	1	1	0	0	1	A, B, C, D, E, G	5.53	5.97	A, B, C, D, E, F	3.99	8.20	A, B, C, D, E, F	5.49	8.95	A, B, C, D, E, F, G, H	9.20	14.21	A, B, C, D, E, F, G	8.70	12.23		-7.13	-9.36	A, B, C, E, F	2.064325	3.057064
0	1	1	0	1	0	A, B, C, D, E, F, G	5.64	6.08	A, B, C, D, E, F, G	4.47	8.64	A, B, C, D, E, F	5.98	9.40	A, B, C, D, E, F, G, H	8.64	13.11	A, B, C, D, E, F, G	9.12	12.62		-5.73	-7.37	A, B, C, D, E, F	2.629916	3.816998
0	1	1	0	1	1	A, B, C, D, E, F, G	5.80	6.22	A, B, C, D, E, F, G	4.84	**9.21**	A, B, C, D, E, F	**6.34**	**9.98**	A, B, C, D, E, F, G, H	8.74	13.30	A, B, C, D, E, F, G	9.35	12.95		-5.06	-6.33	A, B, C, D, E, F	**3.048364**	**4.481953**
0	1	1	1	0	0	A, B, C, D, E, G	5.31	5.82	A, B, C, D, E, F	3.35	7.50	A, B, C, D, E, F	4.86	8.25	A, B, C, D, E, F, G, H	8.91	13.90	A, B, C, D, E, F, G	8.29	11.82		-8.11	-10.52	A, B, C, E, F	1.308806	2.183928
0	1	1	1	0	1	A, B, C, D, E, G	5.53	5.97	A, B, C, D, E, F	3.99	8.20	A, B, C, D, E, F	5.49	8.95	A, B, C, D, E, F, G, H	9.20	14.21	A, B, C, D, E, F, G	8.70	12.23		-7.13	-9.36	A, B, C, E, F	2.064325	3.057064
0	1	1	1	1	0	A, B, C, D, E, F, G	5.64	6.08	A, B, C, D, E, F, G	4.47	8.64	A, B, C, D, E, F	5.98	9.40	A, B, C, D, E, F, G, H	8.64	13.11	A, B, C, D, E, F, G	9.12	12.62		-5.73	-7.37	A, B, C, D, E, F	2.629916	3.816998
0	1	1	1	1	1	A, B, C, D, E, F, G	5.80	6.22	A, B, C, D, E, F, G	4.84	**9.21**	A, B, C, D, E, F	**6.34**	**9.98**	A, B, C, D, E, F, G, H	8.74	13.30	A, B, C, D, E, F, G	9.35	12.95		-5.06	-6.33	A, B, C, D, E, F	**3.048364**	**4.481953**
1	0	0	0	0	0	A, B, C, D, E, G	7.45	4.52	A, B, C, D, E, F	1.34	3.11	A, B, C, D, E, F	1.96	3.34	A, B, C, D, E, F, G, H	11.03	14.11	A, B, C, D, E, F, G	11.12	9.84		-12.34	-18.70	A, E	-2.73571	-4.86911
1	0	0	0	0	1	A, B, C, D, E, G	8.07	5.15	A, B, C, D, E, F	2.80	4.74	A, B, C, D, E, F	3.43	4.99	A, B, C, D, E, F, G, H	12.14	15.38	A, B, C, D, E, F, G	12.29	11.17		-10.46	-16.54	A, B, E	-1.13169	-3.08852
1	0	0	0	1	0	A, B, C, D, E, G	8.41	5.59	A, B, C, D, E, F, G	3.88	6.03	A, B, C, D, E, F	4.51	6.30	A, B, C, D, E, F, G, H	11.76	14.91	A, B, C, D, E, F, G	13.31	12.39		-7.99	-13.19	A, B, E	0.345967	-1.0354
1	0	0	0	1	1	A, B, C, D, E, G	8.84	6.17	A, B, C, D, E, F, G	4.73	7.42	A, B, C, D, E, F	5.38	7.69	A, B, C, D, E, F, G, H	12.09	15.69	A, B, C, D, E, F, G	13.96	13.49		-6.75	-11.23	A, B, E	1.238458	0.412545
1	0	0	1	0	0	A, B, C, D, E, G	9.23	6.95	A, B, C, D, E, F, G	4.11	7.05	A, B, C, D, E, F	4.70	7.29	A, B, C, D, E, F, G, H	11.26	15.41	A, B, C, D, E, F, G	13.09	13.04		-7.47	-11.98	A, B, E	0.726427	0.170836
1	0	0	1	0	1	A, B, C, D, E, G	9.34	7.05	A, B, C, D, E, F, G	4.60	7.65	A, B, C, D, E, F, G	5.20	7.89	A, B, C, D, E, F, G, H	11.51	15.73	A, B, C, D, E, F, G	13.34	13.34		-6.64	-10.94	A, B, C, E	1.305302	0.85818
1	0	0	1	1	0	A, B, C, D, E, G	9.41	7.23	A, B, C, D, E, F, G	5.07	8.36	A, B, C, D, E, F, G	5.68	8.62	A, B, C, D, E, F, G, H	10.72	14.70	A, B, C, D, E, F, G	13.81	14.00		-5.09	-8.42	A, B, C, E	1.998634	2.116308
1	0	0	1	1	1	A, B, C, D, E, G	**9.49**	**7.32**	A, B, C, D, E, F, G	5.34	8.84	A, B, C, D, E, F, G	5.95	9.10	A, B, C, D, E, F, G, H	10.79	14.90	A, B, C, D, E, F, G	13.93	14.23		-4.51	-7.48	A, B, C, E, F	2.292846	2.627442
1	0	1	0	0	0	A, B, C, D, E, G	5.33	5.58	A, B, C, D, E, F, G	3.51	6.90	A, B, C, D, E, F	4.08	7.16	A, B, C, D, E, F, G, H	7.39	11.67	A, B, C, D, E, F, G	7.48	10.10		-4.84	-7.57	A, B, C, D, E, F	1.423926	2.071994
1	0	1	0	0	1	A, B, C, D, E, G	5.41	5.64	A, B, C, D, E, F, G	3.80	7.27	A, B, C, D, E, F, G	4.37	7.53	A, B, C, D, E, F, G, H	7.56	11.85	A, B, C, D, E, F, G	7.63	10.26		-4.26	-6.81	A, B, C, D, E, F	1.793055	2.514528
1	0	1	0	1	0	A, B, C, D, E, F, G	5.48	5.71	A, B, C, D, E, F, G	4.08	7.64	A, B, C, D, E, F, G	4.66	7.90	A, B, C, D, E, F, G, H	7.04	10.99	A, B, C, D, E, F, G	7.94	10.64		-3.19	-4.95	A, B, C, D, E, F	2.170048	3.19564
1	0	1	0	1	1	A, B, C, D, E, F, G	5.53	5.77	A, B, C, D, E, F, G	4.23	7.93	A, B, C, D, E, F, G	4.81	8.20	A, B, C, D, E, F, G, H	7.10	11.10	A, B, C, D, E, F, G	8.00	10.76		-2.78	-4.25	A, B, C, D, E, F	2.346939	3.517021
1	0	1	1	0	0	A, B, C, D, E, G	5.33	5.58	A, B, C, D, E, F, G	3.51	6.90	A, B, C, D, E, F	4.08	7.16	A, B, C, D, E, F, G, H	7.39	11.67	A, B, C, D, E, F, G	7.48	10.10		-4.84	-7.57	A, B, C, D, E, F	1.423926	2.071994
1	0	1	1	0	1	A, B, C, D, E, G	5.41	5.64	A, B, C, D, E, F, G	3.80	7.27	A, B, C, D, E, F, G	4.37	7.53	A, B, C, D, E, F, G, H	7.56	11.85	A, B, C, D, E, F, G	7.63	10.26		-4.26	-6.81	A, B, C, D, E, F	1.793055	2.514528
1	0	1	1	1	0	A, B, C, D, E, F, G	5.48	5.71	A, B, C, D, E, F, G	4.08	7.64	A, B, C, D, E, F, G	4.66	7.90	A, B, C, D, E, F, G, H	7.04	10.99	A, B, C, D, E, F, G	7.94	10.64		-3.19	-4.95	A, B, C, D, E, F	2.170048	3.19564
1	0	1	1	1	1	A, B, C, D, E, F, G	5.53	5.77	A, B, C, D, E, F, G	4.23	7.93	A, B, C, D, E, F, G	4.81	8.20	A, B, C, D, E, F, G, H	7.10	11.10	A, B, C, D, E, F, G	8.00	10.76		-2.78	-4.25	A, B, C, D, E, F	2.346939	3.517021
1	1	0	0	0	0	A, B, C, D, E, G	7.39	4.77	A, B, C, D, E, F	1.49	3.54	A, B, C, D, E, F	2.10	3.76	A, B, C, D, E, F, G, H	10.73	14.03	A, B, C, D, E, F, G	11.14	10.23		-11.66	-17.33	A, B, E	-2.52934	-4.23906
1	1	0	0	0	1	A, B, C, D, E, G	7.98	5.38	A, B, C, D, E, F	2.93	5.14	A, B, C, D, E, F	3.55	5.36	A, B, C, D, E, F, G, H	11.84	15.25	A, B, C, D, E, F, G	12.29	11.53		-9.82	-15.21	A, B, E	-0.95242	-2.50219
1	1	0	0	1	0	A, B, C, D, E, G	8.27	5.69	A, B, C, D, E, F, G	3.94	6.26	A, B, C, D, E, F	4.57	6.50	A, B, C, D, E, F, G, H	11.43	14.63	A, B, C, D, E, F, G	13.23	12.53		-7.57	-12.31	A, B, E	0.417721	-0.72674
1	1	0	0	1	1	A, B, C, D, E, G	8.68	6.24	A, B, C, D, E, F, G	4.78	7.61	A, B, C, D, E, F	5.42	7.86	A, B, C, D, E, F, G, H	11.77	15.40	A, B, C, D, E, F, G	13.86	13.60		-6.35	-10.38	A, B, E	1.293905	0.688457
1	1	0	1	0	0	A, B, C, D, E, G	9.09	6.99	A, B, C, D, E, F, G	4.15	7.19	A, B, C, D, E, F	4.73	7.39	A, B, C, D, E, F, G, H	10.97	15.15	A, B, C, D, E, F, G	13.02	13.10		-7.03	-11.08	A, B, E	0.757767	0.380504
1	1	0	1	0	1	A, B, C, D, E, G	9.19	7.09	A, B, C, D, E, F, G	4.63	7.76	A, B, C, D, E, F, G	5.22	7.98	A, B, C, D, E, F, G, H	11.23	15.47	A, B, C, D, E, F, G	13.27	13.39		-6.22	-10.05	A, B, C, E	1.327594	1.055743
1	1	0	1	1	0	A, B, C, D, E, G	9.24	7.20	A, B, C, D, E, F, G	5.08	8.41	A, B, C, D, E, F, G	5.67	8.63	A, B, C, D, E, F, G, H	10.44	14.38	A, B, C, D, E, F, G	13.70	13.96		-4.81	-7.83	A, B, C, E	1.979461	2.1876
1	1	0	1	1	1	A, B, C, D, E, G	9.31	7.29	A, B, C, D, E, F, G	**5.34**	8.87	A, B, C, D, E, F, G	5.94	9.11	A, B, C, D, E, F, G, H	10.51	14.57	A, B, C, D, E, F, G	13.82	14.18		-4.25	-6.90	A, B, C, E, F	2.268573	2.691419
1	1	1	0	0	0	A, B, C, D, E, F, G	5.26	5.57	A, B, C, D, E, F, G	3.53	6.95	A, B, C, D, E, F	4.09	7.18	A, B, C, D, E, F, G, H	7.25	11.53	A, B, C, D, E, F, G	7.45	10.09		-4.52	-6.95	A, B, C, D, E, F	1.451261	2.191244
1	1	1	0	0	1	A, B, C, D, E, F, G	5.33	5.62	A, B, C, D, E, F, G	3.82	7.30	A, B, C, D, E, F, G	4.38	7.54	A, B, C, D, E, F, G, H	7.42	11.70	A, B, C, D, E, F, G	7.60	10.25		-3.96	-6.20	A, B, C, D, E, F	1.815195	2.626678
1	1	1	0	1	0	A, B, C, D, E, F, G	5.40	5.68	A, B, C, D, E, F, G	4.08	7.65	A, B, C, D, E, F, G	4.65	7.88	A, B, C, D, E, F, G, H	6.92	10.83	A, B, C, D, E, F, G	7.88	10.60		-2.97	-4.52	A, B, C, D, E, F	2.174737	3.246057
1	1	1	0	1	1	A, B, C, D, E, F, G	5.44	5.73	A, B, C, D, E, F, G	4.23	7.93	A, B, C, D, E, F, G	4.79	8.17	A, B, C, D, E, F, G, H	6.98	10.94	A, B, C, D, E, F, G	7.95	10.72		**-2.57**	**-3.83**	A, B, C, D, E, F	2.34861	3.563828
1	1	1	1	0	0	A, B, C, D, E, F, G	5.26	5.57	A, B, C, D, E, F, G	3.53	6.95	A, B, C, D, E, F	4.09	7.18	A, B, C, D, E, F, G, H	7.25	11.53	A, B, C, D, E, F, G	7.45	10.09		-4.52	-6.95	A, B, C, D, E, F	1.451261	2.191244
1	1	1	1	0	1	A, B, C, D, E, F, G	5.33	5.62	A, B, C, D, E, F, G	3.82	7.30	A, B, C, D, E, F, G	4.38	7.54	A, B, C, D, E, F, G, H	7.42	11.70	A, B, C, D, E, F, G	7.60	10.25		-3.96	-6.20	A, B, C, D, E, F	1.815195	2.626678
1	1	1	1	1	0	A, B, C, D, E, F, G	5.40	5.68	A, B, C, D, E, F, G	4.08	7.65	A, B, C, D, E, F, G	4.65	7.88	A, B, C, D, E, F, G, H	6.92	10.83	A, B, C, D, E, F, G	7.88	10.60		-2.97	-4.52	A, B, C, D, E, F	2.174737	3.246057
1	1	1	1	1	1	A, B, C, D, E, F, G	5.44	5.73	A, B, C, D, E, F, G	4.23	7.93	A, B, C, D, E, F, G	4.79	8.17	A, B, C, D, E, F, G, H	6.98	10.94	A, B, C, D, E, F, G	7.95	10.72		**-2.57**	**-3.83**	A, B, C, D, E, F	2.34861	3.563828

The existing approaches were implemented by hand based on the descriptions from their respective papers and should match the original implementation to the best of our knowledge, but there could be implementation errors or slight differences from the original implementation made by the authors. Additionally, only a subset of existing approaches was implemented, so in order to do a more thorough evaluation of our proposed technique ideally it should be validated on more existing approaches. The same can be said for the dataset corpus – a more comprehensive set of repositories could be used, both internal and public Microsoft repositories, and ultimately OSS repositories and ones from different companies.

VI. CONCLUSION AND FUTURE WORK

In this paper, we went over several existing reviewer recommendation approaches and determined that on average they have a tendency to have reduced performance for larger repositories. We then introduced a technique which uses a filter for helping pull request reviewer recommendation systems focus on relevant reviewers.

We then tested several filters and were able to achieve performance improvement for the majority of the existing approaches that we evaluated. Additionally, by combining the filters, we managed to achieve even greater improvements which, peaked at 17.60% better precision and 21.23% better recall over all datasets, while the improvements per dataset went as high as 43.28% for precision and 30.94% for recall. Although the performance benefits are not consistent across repositories and approaches, every approach experienced at least some level of improvements, with the majority of them experiencing stable improvements across all but the dataset with the smallest repository. Because of that, we believe that the proposed technique represents a valid approach when trying to improve the performance of existing reviewer recommendation approaches, especially for larger repositories with many reviewer candidates.

Our experiments showed that, although there are some filters that can be expected to benefit almost any case, there was a difference in the benefits each filter had for different datasets and approaches. We believe that the size of the repository and the organization of the development team does influence the magnitude of improvements, and we plan to investigate this further. We also plan to continue evaluating our techniques on more existing approaches and with a larger dataset corpus.

APPENDIX A
RQ1 – EXTENDED RESULTS

See Table 5.

APPENDIX B
RQ2 – EXTENDED RESULTS

See Table 6.

ACKNOWLEDGMENT

Nikola Pejić thanks Goran Lukić, Vlado Tošev, and Aleksandar Vujić for supporting the article by allowing a portion of his time designated for the development of Synapse SQL Serverless to be allocated for this research.

REFERENCES

[1] G. Gousios, M. Storey, and A. Bacchelli, ''Work practices and challenges in pull-based development: The Contributor's perspective,'' in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 285–296, doi: 10.1145/2884781.2884826.

[2] G. Gousios, A. Zaidman, M. Storey, and A. V. Deursen, ''Work practices and challenges in pull-based development: The Integrator's perspective,'' in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, vol. 1, May 2015, pp. 358–368, doi: 10.1109/ICSE.2015.55.

[3] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K. Matsumoto, ''Who should review my code? A file location-based code-reviewer recommendation approach for modern code review,'' in *Proc. IEEE 22nd Int. Conf. Softw. Anal., Evol., Reeng. (SANER)*, Mar. 2015, pp. 141–150, doi: 10.1109/SANER.2015.7081824.

[4] Y. Yu, H. Wang, G. Yin, and T. Wang, ''Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?'' *Inf. Softw. Technol.*, vol. 74, pp. 204–218, Jun. 2016, doi: 10.1016/j.infsof.2016.01.004.

[5] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey, ''Investigating code review quality: Do people and participation matter?'' in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2015, pp. 111–120, doi: 10.1109/ICSM.2015.7332457.

[6] A. Bosu, J. C. Carver, C. Bird, J. Orbeck, and C. Chockley, ''Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at Microsoft,'' *IEEE Trans. Softw. Eng.*, vol. 43, no. 1, pp. 56–75, Jan. 2017, doi: 10.1109/TSE.2016.2576451.

[7] C. Sadowski, E. Söderberg, L. Church, M. Sipko, and A. Bacchelli, ''Modern code review: A case study at Google,'' in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng., Softw. Eng. Pract. Track (ICSE-SEIP)*, May 2018, pp. 181–190, doi: 10.1145/3183519.3183525.

[8] S. Asthana, R. Kumar, R. Bhagwan, C. Bird, C. Bansal, C. Maddila, S. Mehta, and B. Ashok, ''WhoDo: Automating reviewer suggestions at scale,'' in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 937–945, doi: 10.1145/3338906.3340449.

[9] Y. Yu, H. Wang, G. Yin, and C. X. Ling, ''Reviewer recommender of pull-requests in GitHub,'' in *Proc. IEEE Int. Conf. Softw. Maintenance Evol.*, Sep. 2014, pp. 609–612, doi: 10.1109/ICSME.2014.107.

[10] M. Fejzer, P. Przymus, and K. Stencel, ''Profile based recommendation of code reviewers,'' *J. Intell. Inf. Syst.*, vol. 50, no. 3, pp. 597–619, 2018, doi: 10.1007/s10844-017-0484-1.

[11] J. Jiang, Y. Yang, J. He, X. Blanc, and L. Zhang, ''Who should comment on this pull request? Analyzing attributes for more accurate commenter recommendation in pull-based development,'' *Inf. Softw. Technol.*, vol. 84, pp. 48–62, Apr. 2017, doi: 10.1016/j.infsof.2016.10.006.

[12] C. Hannebauer, M. Patalas, S. Stünkelt, and V. Gruhn, ''Automatically recommending code reviewers based on their expertise: An empirical comparison,'' in *Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2016, pp. 99–110.

[13] X. Ye, ''Learning to rank reviewers for pull requests,'' *IEEE Access*, vol. 7, pp. 85382–85391, 2019, doi: 10.1109/ACCESS.2019.2925560.

[14] C. Bird, T. Carnahan, and M. Greiler, ''Lessons learned from building and deploying a code review analytics platform,'' in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, May 2015, pp. 191–201, doi: 10.1109/MSR.2015.25.

[15] R. Ranawana and V. Palade, ''Multi-classifier systems: Review and a roadmap for developers,'' *Int. J. Hybrid Intell. Syst.*, vol. 3, no. 1, pp. 35–61, Apr. 2006, doi: 10.3233/his-2006-3104.

[16] T. Joachims, ''Optimizing search engines using clickthrough data,'' in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2002, pp. 1–10, doi: 10.1145/775047.775067.

[17] T. Joachims, ''Training linear SVMs in linear time,'' in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. discovery data mining*, Aug. 2006, doi: 10.1145/1150402.1150429.

[18] P. Thongtanunam and A. E. Hassan, "Review dynamics and their impact on software quality," *IEEE Trans. Softw. Eng.*, vol. 47, no. 12, pp. 2698–2712, Dec. 2021, doi: 10.1109/TSE.2020.2964660.

[19] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Revisiting code ownership and its relationship with software quality in the scope of modern code review," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 1039–1050, doi: 10.1145/2884781.2884852.

[20] K. Herzig, L. Ghostling, M. Grothusmann, S. Just, N. Huang, A. Klimowski, Y. Ramkumar, M. McLeroy, K. Muslu, H. Sajnani, and V. Vadaga, "Microsoft CloudMine: Data mining for the executive order on improving the Nation's cybersecurity," in *Proc. IEEE/ACM 19th Int. Conf. Mining Softw. Repositories (MSR)*, May 2022, p. 639, doi: 10.1145/3524842.3528514.

NIKOLA PEJIĆ received the B.Sc. and M.Sc. degrees in electrical and computer engineering from the School of Electrical Engineering, University of Belgrade, Serbia, in 2018 and 2019, respectively, where he is currently pursuing the Ph.D. degree. In 2019, he joined the Microsoft Development Center Serbia as a Software Engineer, where he is currently a member of the Applied Sciences Group.

ZAHARIJE RADIVOJEVIĆ received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from the School of Electrical Engineering, University of Belgrade, Serbia, in 2002, 2006, and 2012, respectively. He is currently an Associate Professor with the University of Belgrade. He teaches several courses on computer architecture and organization, e-business infrastructure, and mobile device programming. His research interests include computer architecture and organization, concurrent and distributed programming, data analysis, simulations, and reverse engineering.

MILOŠ CVETANOVIĆ received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from the School of Electrical Engineering, University of Belgrade, Serbia, in 2003, 2006, and 2012, respectively. He is currently an Associate Professor with the University of Belgrade. He teaches several courses on databases and database software tools, information systems, and e-business infrastructure. His research interests include the area of database and information systems, artificial intelligence, big data, and reverse engineering.