Empirical Software Engineering (2022) 27:181
https://doi.org/10.1007/s10664-022-10228-0

Using Screenshot Attachments in Issue Reports for Triaging

Ethem Utku Aktas[1] · **Cemal Yilmaz**[2]

Accepted: 17 August 2022 / Published online: 24 September 2022

Abstract
In previous work, we deployed IssueTAG, which uses the one-line summary and the description fields of the issue reports to automatically assign them to the stakeholders, who are responsible for resolving the reported issues. Since its deployment on $January$ 12, 2018 at Softtech – the software subsidiary of the largest private bank in Turkey, IssueTAG has made a total of 301,752 assignments (as of $November$ 2021). One observation we make is that a large fraction of the issue reports submitted to Softtech has screenshot attachments and, in the presence of such attachments, the reports often convey less information in their one-line summary and the description fields, which tends to reduce the assignment accuracy. In this work, we use the screenshot attachments as an additional source of information to further improve the assignment accuracy, which, to the best of our knowledge, has not been studied before for automatic issue assignments. In particular, we develop a number of multi-source assignment models, which use both the issue reports and the screenshot attachments, as well as a number of single source models, which use either the issue reports or the screenshot attachments, and empirically evaluate them on real issue reports. Compared to the currently deployed single-source model in the field, the best multi-source model improved the assignment accuracy from 0.848 to 0.855 at an acceptable overhead cost, reducing the overall 3.3 percentage-point deficit between the human triagers and the deployed system by 0.7 points.

Keywords Issue triaging · Issule report assignment · Optical character recognition · Text classification · Support vector machines

Communicated by: Sigrid Eldh, Davide Falessi, Burak Turhan

This article belongs to the Topical Collection: *Software Engineering in Practice*

✉ Ethem Utku Aktas
utku.aktas@softtech.com.tr

Cemal Yilmaz
cyilmaz@sabanciuniv.edu

[1] Softtech Inc., Research and Development Center, Istanbul, 34947, Turkey

[2] Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, 34956, Turkey

1 Introduction

Issue assignment is the process of assigning the issue reports (also known as the *bug reports* or *problem reports*) to the stakeholders, who are responsible for resolving the reported issues. As this process is costly, tedious, and error-prone, automating it is of great practical importance, especially for the companies, which receive a large number of issue reports regularly from the field (Jonsson et al. 2016; Lee et al. 2017; Chen et al. 2019a).

Softtech[1], which constitutes the industrial setup in this work, is one such company. Being a subsidiary of IsBank[2] – the largest private bank in Turkey, Softtech receives an average of 350 issue reports from the field on a daily basis for its 400+ software products comprised of around 100 millions of lines of code (as of Nov 01, 2021). Since these issue reports are typically concerned with business-critical systems, they often need to be handled with utmost importance and urgency. To this end, Softtech and IsBank employ a total of 80 full-time employees, the sole purpose of which is to carry out the issue triaging process (Aktas EU and Yilmaz C 2020a). Even with this dedicated team of employees, the issue assignment process at Softtech was still suffering due to a number of factors, including the ineffectiveness of maintaining a knowledge base regarding the stakeholders and their responsibilities in an ad hoc manner (to help with the assignments), the "cost" of training new triagers, the inevitable friction between the triagers and the development teams in the presence of incorrect assignments, and all of the associated inefficiencies in the triaging process, such as increased turnaround time for resolutions.

To overcome these shortcomings, we, in a previous work, developed an automated issue assignment system, called *IssueTAG*, and deployed it at Softtech (Aktas EU and Yilmaz C 2020a). At a very high level, IssueTAG uses the natural language sentences present in the one-line summary and the description fields of the issue reports to assign the reported issues to the development teams (Section 3).

Since its deployment on Jan 12, 2018, IssueTAG has been making all the initial assignments in an automated manner (about 301,752 assignments as of Nov 27, 2021). Although the assignment accuracy of the system has been slightly lower than that of the human triagers (0.831 vs. 0.864, i.e., a 3.3 percentage-point (out of 100) deficit (Aktas EU and Yilmaz C 2020a)), this does not prevent the stakeholders from perceiving the deployment system as useful. This is also apparent from a survey we carried out where 79% of the participants "agreed" or "strongly agreed" that IssueTAG is useful (Aktas EU and Yilmaz C 2020a). One reason behind this is that IssueTAG helps the stakeholders defer the responsibility of making the assignments, which is a quite tedious and cumbersome task to carry out manually (Aktas EU and Yilmaz C 2020a). Another reason is that IssueTAG (together with all the modifications made to the triaging process around it) reduces the manual effort required for the assignments by about 5 person-months per year and improves the turnaround time for resolutions by about 20%, on average (Aktas EU and Yilmaz C 2020a).

We have nevertheless been working on further improving the assignment accuracy of the system, especially on figuring out the potential causes of the differences between the accuracy of the human triagers and that of the deployed system. To this end, one observation we make is that a majority of the issue reports submitted to Softtech (68%) have attachments and a majority of these attachments (84.3%) are the actual snapshots of the screens, on which the failures are observed. Although these attachments convey valuable information

[1] https://softtech.com.tr

[2] https://www.isbank.com.tr

for issue assignment, they are completely ignored by IssueTAG. As a matter of fact, we are not aware of any work, which utilizes the screenshot attachments in the issue reports for assignment.

Interestingly enough, we also observe that the issue reports with attachments tend to have lower assignment accuracy, compared to those without any attachments (0.80 vs. 0.88, see Section 4 for more information). An in-depth analysis revealed that this could be because the issue reports with the attachments tend to convey less information in their one-line summaries and descriptions as much of the information is already included in the attachments. We, in a study, indeed observed that while the issue reports with attachments had an average of 29 words, those without any attachments had 41 words (Aktas EU and Yilmaz C 2020a).

In this work, we develop and empirically evaluate a number of machine learning (ML) approaches, including the multimodal ones, which use the screenshot attachments in issue reports as an additional source of information for assignments. In previous work (a poster paper) (Aktas and Yilmaz 2020b), we briefly discussed the plausibility of the general idea and presented some preliminary results. In this work, on the other hand, we study the nature of the information present in screenshot attachments in an industrial setup; present a number of additional single-source (utilizing either the textual information or the screenshot attachments present in the issue reports) and multi-source (utilizing both the textual information and the screenshot attachments present in the issue reports) approaches for issue assignment; empirically compare the proposed approaches to a number of alternative approaches (i.e., comparing the multi-source approaches to the single-source approaches); and rigorously evaluate all of the presented approaches by using real issue reports.

More specifically, we address the following research questions in this work:

- RQ1: How frequently do the issue reports submitted to Softtech have screenshot attachments and does the automatic assignment accuracy get affected by the presence of such attachments?
- RQ2: How can the screenshot attachments in issue reports be used to further improve the accuracy of the assignments?
- RQ2.1: How do using the visual features and using the textual features extracted from the screenshot attachments compare in terms of the accuracy of the assignments obtained?
- RQ2.2: In the presence of screenshot attachments, how much information does the text present in the one-line summary and the description fields of the issue reports still provide for the assignments?
- RQ2.3: How do the multi-source models compare to single-source models in terms of the assignment accuracies they provide?
- RQ2.4: Does the assignment accuracy obtained from the multi-source model get affected by the absence of the screenshot attachments?
- RQ2.5: Are the differences between the assignment accuracies obtained from the best performing multi-source and single-source models statistically significant?
- RQ3: How does taking the screenshot attachments into account affect the overall performance of the system in terms of the training and the prediction times?

To this end, we have first developed 14 machine learning (ML) models; 3 models leveraging only the textual information present in the issue reports, 4 models leveraging only the screenshot attachments, and 7 models leveraging both sources of information. We have then carried out a series of experiments, in which we used a total of 84,972 real issue reports submitted to 68 distinct teams at Softtech between September, 2018 and August, 2019.

In these experiments, we have evaluated both the correctness of the assignments by using well-known metrics, namely accuracy, precision, recall, and F-measure, and the runtime performance of the models by using the training and the prediction times.

The results of our experiments strongly suggest that using screenshot attachments as an additional source of information can improve the accuracy of the assignments at an acceptable cost. In particular, compared to the currently deployed single-source model in the field, our best multi-source model improved the assignment accuracy for the issue reports with screenshot attachments from 0.843 to 0.858 with an overall improvement from 0.848 to 0.855 across all the issue reports with and without screenshot attachments. Furthermore, these improvements were achieved with an increase in the training and prediction (per issue report) times from 190.4 to 507.6 seconds and from 0.9 to 2.17 seconds, on average, respectively, both of which were in the range of acceptable overheads for Softtech.

Note that the practical value of these seemingly small differences in the assignment accuracies can better be understood when we consider how much they contribute to close the 3.3 percentage-point deficit between the human triagers and the deployed IssueTAG system (Aktas EU and Yilmaz C 2020a). Therefore, although some care must be taken when interpreting these numbers, as the last available accuracy measurement regarding the performance of the human triagers dates back before the deployment of IssueTAG (all the initial assignments have been automatically made by IssueTAG since its deployment), the overall 0.7 percentage-point increase (0.7 out of 3.3) is still of practical value to Softtech. After all, almost all of the issue reports submitted to Softtech are concerned with business-critical systems. Therefore, decreasing the turnaround time for fixing the issues by reducing issue tossing while at the same time automating the entire process as much as possible, is important for Softtech. Nevertheless, the percentage-point increase in the assignment accuracy for the issue reports with the screenshot attachments, which are the main focus of this work, was 1.5 (0.843 to 0.858). And, we showed that the improvements are, indeed, statistically significant, i.e., they are not likely to be by chance.

The remainder of the paper is organized as follows: Section 2 introduces the industrial setup we have; Section 3 provides background information on the previously deployed IssueTAG system; Section 4 discusses the motivation behind the work in detail and carries out a feasibility study to better understand the nature of the information present in the screenshot attachments; Section 5 presents the proposed approaches; Section 6 presents the experiments we carried out to evaluate the proposed approaches; Section 7 discusses threats to validity; Section 8 summarizes the related work; and Section 9 concludes with some future work ideas.

2 Case Description

Softtech receives an average of 350 software-related issue reports on a daily basis from the field. The reported issues include both the bank clerks having software failures and the bank customers facing software-related problems in any of the banking channels, including mobile, Web, and ATM. Each issue report contains a one-line summary, which captures the essence of the reported issue, and a description, which provides further information regarding the steps for reproducing the reported issues, expected behavior, and observed behavior. Both fields accept natural language sentences in Turkish. In the remainder of the paper, the content of these fields will be referred to as the *textual information present in the issue reports*.

Not all the reported issues require modifications in the codebase. Some issues, for example, are resolved by running fully automated scripts to pinpoint and fix the issues regarding the availability of the certain services. Some issues are even resolved by manually updating certain records in the databases (typically the ones related to the operations of the banking services). In either case, the reported issues, as they typically concern business-critical systems, need to be addressed with utmost importance and urgency.

To carry out the triaging process, two dedicated teams of 80 full-time employees are employed; IT Help Desk (IT-HD) and Application Support Team (AST). The IT-HD clerks, being consisted of 50 non-technical personnel employed at IsBank, are the first team receiving the issue reports from the field. To file an issue, both the bank clerks and the bank customers can either call or send an email possibly with attachments (such as screenshots) to IT-HD, describing the IT-related issues they are facing. Neither IsBank nor Softtech provides a specialized tool or a template to the bank customers for submitting the issue reports. The bank clerks, however, have an internal tool that they can use to create an issue report, which also allows the attachments (but not specialized functionally for taking the screenshots). After receiving a call, an email, or a pre-filled issue report created by a bank clerk using the aforementioned tool, an IT-HD clerk (if needed) files the actual issue report by making sure that all the necessary information required for resolving the reported issue is included in the report to the extent possible. Further communication with the reporter may follow to collect more information. If the reported issue is an issue that can be resolved by the IT-HD clerk using, for example, some basic troubleshooting guides, the IT-HD clerk resolves the issue and closes the report. Otherwise, the issue report needs to be dispatched to the proper unit at IsBank or Softtech.

In the case of a software-related issue, the report is dispatched to the AST team, a group of 30 somewhat technical personnel employed at Softtech. The AST members are embedded in the software development teams. They are not as technical as software engineers (e.g., they are not software developers), but more technical than the IT-HD clerks as they are capable of resolving most of the issues that do not require any modifications in the codebase by, for example, running scripts or making changes in the databases. Note that multiple issue reports may report the same issue. If this is pinpointed by the AST members, these reports are marked as the duplicate of each other. Note further that, the automatic identification of the duplicate reports is outside the scope of this paper. The issue reports that require changes in the codebase are, however, addressed by the software engineers.

Before the deployment of IssueTAG, IT-HD clerks were responsible for assigning the issue reports to the software development teams, who are responsible for resolving the reported issues. To this end, IT-HD clerks were using their experiences together with a keyword-based knowledge base, which they collectively maintained in an ad hoc manner. In the case of an incorrect assignment, the issue reports were returned to the IT-HD clerks for reassignment. This was, however, giving rise to issue tossing between the IT-HD clerks and the development teams, causing waste of time.

Note that Softtech prefers to designate the development teams as the assignees. The reason behind this decision is to let the development teams manage the issues regarding the team dynamics within the team. Otherwise, factors, such as the current workloads of the individual developers, the changes in the team structures, and the current status of the developers (e.g., developers on leave), would be quite difficult to take into account during the automated assignment process.

3 IssueTAG

IssueTAG automatically assigns the issue reports to the development teams (Aktas EU and Yilmaz C 2020a). Since the deployment of IssueTAG on *January* 12, 2018 at Softtech, all of the initial assignments (a total of 301,752 assignments as of *November* 2021) have indeed been made automatically by the system.

At a very high level, IssueTAG casts the problem of issue assignment to a classification problem, which takes as input the natural language descriptions present in the one-line summary and description fields of the issue reports and produces as output the assignments (Aktas EU and Yilmaz C 2020a). Note that the issue reports submitted to Softtech do not contain any other fields conveying information that could be used for automated assignment, such as product, component, and version information. The main reason is that the software systems maintained by Softtech are a part of a large business-critical ecosystem where they heavily interact with each other. Therefore, from the perspective of issue triaging, the boundaries of the software products and/or the components are not clear at all. For example, almost all of the GUI screens interact with the core banking system, which is maintained by a different set of development teams than the ones producing the screens. Furthermore, within a screen, there can be multiple widgets and/or tabs, each of which is maintained by a different team. Last but not least, carrying a financial transaction typically crosses the boundaries of multiple products/components developed by different teams.

IssueTAG also generates human readable explanations for the assignments, which can be interpreted even by non-technical stakeholders. This was indeed an actual need we discovered only after deploying IssueTAG; the development teams, especially for the incorrect assignments (as this may have an adverse effect on the score cards of the teams), tend to demand explanations as to why the assignments are made in the way they are.

Another feature implemented by IssueTAG, which is quite important for an automated assignment system operating in a business-critical environment, is a self-monitoring mechanism. In particular, IssueTAG monitors the accuracy of its predictions on a daily basis (by using a change point detection algorithm (Truong et al. 2018a; 2018b)) and re-trains the classification models when the assignment accuracy starts to deteriorate. Even in the absence of any deviations in the daily accuracies, the classification models are re-trained regularly at every month to account for the changes in the team structure. For the re-training, IssueTAG uses all the issue reports, which were resolved within the last 6 months, as the training set.

4 Motivation

We have been working on further improving the assignment accuracy of IssueTAG ever since its deployment. To this end, one observation we make is that although a majority of the issue reports submitted to Softtech have attachments, conveying valuable information that can be used toward improving the assignment accuracy, these attachments are completely ignored by IssueTAG. To analyze the existing state of affairs in detail, thus to address our first research question *RQ1: How frequently do the issue reports submitted to Softtech have screenshot attachments and does the automatic assignment accuracy get affected by the presence of such attachments?*, we carried out a feasibility study.

In the study, we used a total of 41,042 real issue reports, which were resolved during the months of March-August in 2019. In the remainder of the paper, these reports will be referred to as the *study data*. We made sure that all of the reports in the study data were actually closed with the “resolved” status, indicating that the reported issues were validated and fixed, and that the last assignee for the report (i.e., the one closing the report) is the correct assignee. Note that since the number of issue reports resolved by a development team is a key performance indicator at Softtech, the developers pay utmost attention to correctly indicate the teams closing the issue reports (thus, the correct assignees for the reports).

We first observed that about 68% (27,952 out of 41,042) of all the issue reports had at least one attachment and that the total number of attachments was 34,647. Figure 1 presents the summary statistics. In particular, 70% (out of 8,322), 69% (out of 7,598), 68% (out of 7,876), 67% (out of 5,334), 68% (out of 7,159), and 65% (out of 4,753) of the issue reports resolved in the months of March-August, respectively, had attachments.

We next observed that although the attachments were of variety of different types (including .png, .doc/docx, .xls/xlsx, .msg, .txt, .pdf, .htm/.html, .xml, and .sql), the most frequently appearing type of attachments was screenshots, capturing the image of the screens, on which the issues were encountered. In particular, among all the issue reports with attachments, 84.30% of them had screenshot attachments; 83.70%, 83.25%, 84.77%, 84.12%, 85.89%, and 84.06% for the months of March-August, respectively.

We then observed that the issue reports with attachments received lower assignment accuracies, compared to those without any attachments. More specifically, while the average assignment accuracy for the issue reports with attachments was 0.80, that for the ones without any attachments was 0.88 (Fig. 2). And, the monthly assignment accuracies for the former were 0.80, 0.78, 0.79, 0.80, 0.82, and 0.81 for the months of March-August, respectively, whereas those for the latter were 0.85, 0.87, 0.90, 0.87, 0.89, and 0.88.

An in-depth analysis revealed that one potential reason for this is that in the presence of attachments, the issue reports tend to convey less information as much of the information is already included in the attachments. This phenomenon is indeed also apparent from the number of words included in the issue reports with and without attachments. For example, while the average number of words in the issue reports with screenshot attachments is 29, that of the reports without any attachments is 41.

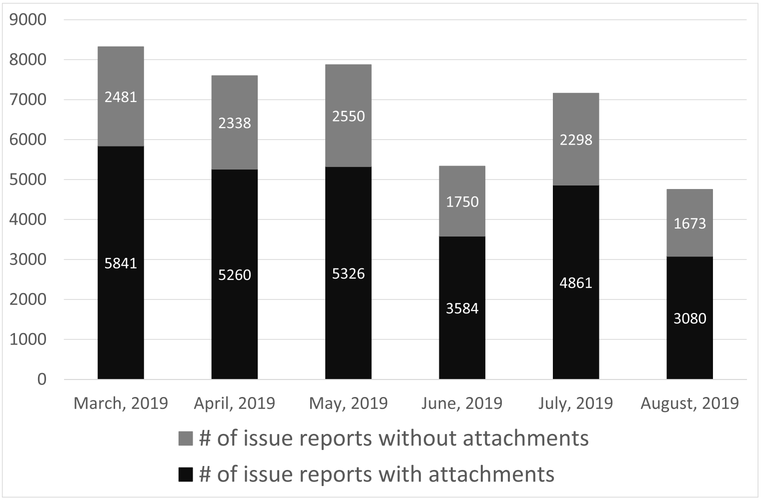


Fig. 1 The distribution of the issue reports with and without attachments

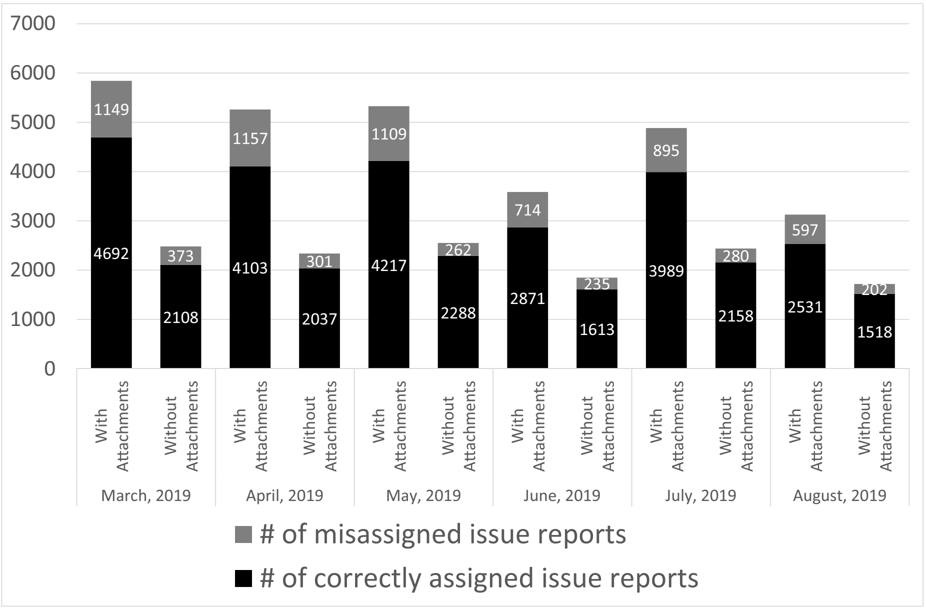


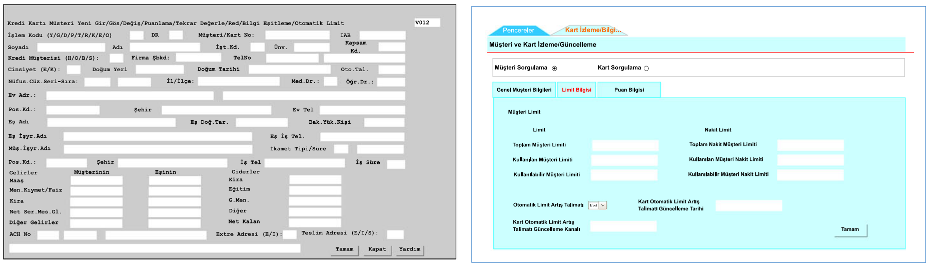
Fig. 2 Comparison of reassignments for the issue reports with and without attachments. The average accuracy obtained for the former was 0.80, that obtained for the latter was 0.88

In this work, since the screenshot attachments are the most frequently appearing type of attachments at Softtech and since there is still room for improving their assignment accuracies, we opted to solely focus on the screenshot attachments. We, in particular, conjecture that using screenshot attachment as an additional source of information (i.e., together with the textual information present in the issue reports), can improve the accuracy of the assignments.

Figure 3 presents two example screenshot attachments included in some issue reports. Note that, due to certain security and privacy concerns, we provide the hand-drawn versions of the actual screenshot attachments, presenting the textual information in the actual screenshots. Furthermore, the screen presented in Fig. 3 a is produced by a module written in COBOL programming language running on mainframes, whereas that presented in Fig. 3 b is a web-based screen created by using recent web technologies.

A majority of the screenshot attachments are sent by the bank clerks working at the branches. However, nothing prevents bank customers from sending screenshot attachments. Furthermore, neither IsBank nor Softtech provides any specialized tools to the clerks or the customers for taking the screenshots. The screenshots are typically taken by using the facilities that are directly supported by the underlying operating system or the web browser being used.

To better understand the nature of the information conveyed in screenshot attachments, we first carried out a feasibility study. The results of this study suggested that extracting the textual information from the screenshot attachments and using it together with the natural language descriptions in the issue reports, could help improve the assignment accuracy. The results were indeed instrumental in designing the solution approaches introduced in Section 5.



```
Kredi Kartı Müsteri Yeni Gir/Gös/Değiş/Puanlama/Tekrar Değerle/Red/Bilgi Eşitleme/Otomatik Limit      V012
İşlem Kodu (Y/G/D/P/T/R/K/E/O)      DR      Müşteri/Kart No:      IAB
Soyadı      Adı      İşt.Kd.      Ünv.      Kapsam Kd.
Kredi Müşterisi (H/O/B/S):      Firma Şbkd:      TelNo
Cinsiyet (E/K):      Doğum Yeri      Doğum Tarihi      Oto.Tal.
Nüfus.Cüz.Seri-Sıra:      İl/İlçe:      Med.Dr.:      Öğr.Dr.:
Ev Adr.:
Pos.Kd.:      Şehir      Ev Tel
Eş Adı      Eş Doğ.Tar.      Bak.Yük.Kişi
Eş İşyr.Adı      Eş İş Tel.
Müş.İşyr.Adı      İkamet Tipi/Süre
Pos.Kd.:      Şehir      İş Tel      İş Süre
Gelirler      Müşterinin      Eşinin      Giderler
Maaş                                    Kira
Men.Kıymet/Faiz                         Eğitim
Kira                                    G.Men.
Net Ser.Mes.Gl.                         Diğer
Diğer Gelirler                          Net Kalan
ACH No      Extre Adresi (E/I):      Teslim Adresi (E/I/S):
                                         Tamam   Kapat   Yardım
```

(a)

Müşteri ve Kart İzleme/Güncelleme

[✓ Müşteri Sorgulama] Kart Sorgulama

Müşteri Limit

Limit		Nakit Limit	
Toplam Müşteri Limiti		Toplam Nakit Müşteri Limiti	
Kullanılan Müşteri Limiti		Kullanılan Müşteri Nakit Limiti	
Kullanılabilir Müşteri Limiti		Kullanılabilir Müşteri Nakit Limiti	

Otomatik Limit Artış Talimatı: Evet

Kart Otomatik Limit Artış Talimatı Güncelleme Tarihi

Kart Otomatik Limit Artış Talimatı Güncelleme Kanalı

(b)

Fig. 3 Example screenshot attachments where (a) is produced by a module written in COBOL programming language running on mainframes and (b) is a web-based screen created by using recent web technologies. Note that, due to certain security and privacy concerns, only the hand-drawn versions of the actual screenshot attachments are provided

To carry out the feasibility study, we have manually analyzed a number of issue reports with screenshot attachments, which were incorrectly assigned by IssueTAG. Table 1 presents some examples, which we will use to summarize the insights we gained throughout the study. Note that due to certain security and privacy concerns, the table provides only the one-line summaries and the descriptions for the aforementioned issue reports where the actual error and transaction codes are obscured and the actual screenshots are omitted.

Regarding the first issue report (Table 1), one would expect that the screen code indicated in both the summary and the description fields of the report would be instrumental in assigning the report. It, however, turns out that this screen code has never occurred in any of the historical issue reports, which simply renders the natural language descriptions present in the report useless. When we manually analyzed the screenshot attachment in the report, we, to our surprise, observed that the error message mentioned in the description was a generic "HTTP 404 - Web page cannot be found" error, which is, indeed, not useful at all either. On the other hand, the textual information present in the remainder of the screen, such as, the titles of the open tabs, clearly indicated that the error message was indeed emitted by the retail loan management module. Had the text been extracted from the attached screenshot and used for the assignment, the report would have been assigned to the correct development team.

Table 1 Example issue reports with screenshot attachments

Issue	One-line summary	Description
1	$[ScreenCode]$	The screen $[ScreenCode]$ does not appear at all terminals in branch $[BranchCode]$; the error given in the attachment is observed.
2	$[ErrorCode]$	Although the requested limits have been updated, we receive the attached error during the approval process of the customer's ($[CustomerCode]$) request.
3	$[ScreenCode]$/$[TransactionCode]$	We receive the attached error for the transaction $[TransactionCode]$ on the screen $[ScreenCode]$.

Regarding the second issue report (Table 1), although, at a first glance, this report seems to be quite similar to the first report in the sense that both reports have a screenshot of the error message emitted as an attachment, a manual analysis of the attachment revealed some interesting differences. More specifically, the image attached to the first report was the screenshot of a screen created by the software module responsible for the failure. The image attached to the second report, on the other hand, was a screenshot obtained from a workflow engine, which enables the bank to create and automate repetitive processes and tasks occurring in a particular order. However, the workflow engine was not responsible for the reported failure. Instead, the failure was caused by a module responsible for handling one of the tasks in the visualized workflow. That is, the screenshot attachment alone was not enough for the assignment. More specifically, the textual information present both in the report and in the screenshot attachment should have been used together to correctly assign the report as combining both sources of information indicated that the failure was related to a module handling credit card limit operations.

Regarding the third issue report (Table 1), interestingly enough, the image of the screen attached to this report has quite a different look and feel, compared to the images of the screens attached to the first two reports. It turns out that the screenshot in this report comes from a module written in COBOL programming language running on mainframes (such as the one given in Fig. 3 a), whereas the other screenshots were images of some web-based screens created by using recent web technologies (such as the one given in Fig. 3b). Although this suggests that classifying the screen images (by using image classification) can help improve the assignment accuracy, an in-depth analysis quickly revealed that this may not be the case in practice (at least for Softtech). The reason is two folds. First, the different development teams at Softtech use the same graphical user interface (GUI) frameworks (we identified three such frameworks including the one used on the mainframes) with the same (or similar) strict GUI design guidelines. Therefore, the look and feel of the screens produced by different development teams are typically quite similar to each other (if not the same). Second, it is not unusual for a development team to use multiple GUI frameworks in their products. For example, a team can use a web-based GUI framework for the non-technical end users and a mainframe-based GUI framework for the more technical users. This, however, did not prevent us from experimenting with the single-source models based on the image classification of the screenshot attachments, which we indeed used as a baseline (Section 5). On the other hand, we clearly observed that, as was the case with the first two issue reports, extracting the textual information from the screenshot attachment in the third issue report would again help us correctly assign the report. This was because the text appearing on the screen indicated that the reported failure was related to a module handling the cheque transactions.

5 Approach

With all the insights we gained from our manual analysis in mind, we have developed a number of approaches to take the screenshot attachments into account when assigning the issue reports. Figures 4, 5 and 6 summarize these approaches. Note that while the names of the approaches provide clues about the classification models used, the sub-scripted symbols and the super-scripted numbers indicate the sources of information leveraged and the number of channels used in the models, respectively. In particular, the sub-scripts t and a

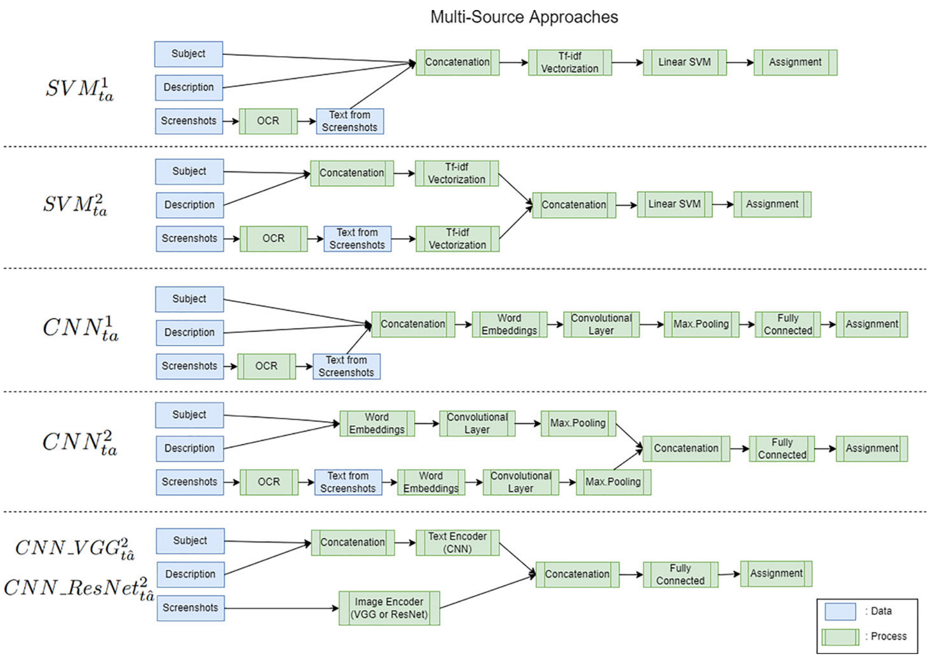


Fig. 4 Proposed multi-source approaches

indicate the inclusion of the textual information present in the issue reports and the inclusion of the screenshot attachments in modeling, respectively. Furthermore, the presence of a hat above a (i.e., $\hat{a}$) specifies that the screenshots are processed as images (i.e., visual features are used in the models), whereas the absence of the hat indicates that only the textual information extracted from the screenshots are processed. And, the number of channels indicate whether the features extracted from t and a are merged together (indicated by the super-script 1) or treated separately (indicated by the super-script 2).

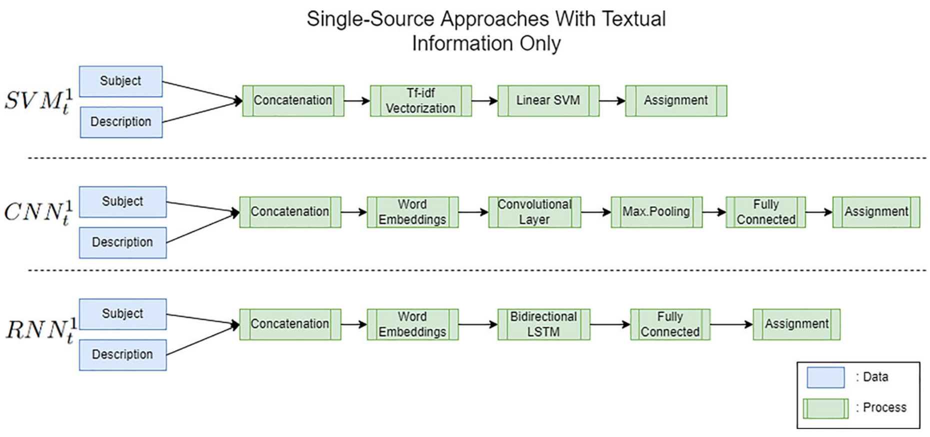


Fig. 5 Proposed single-source approaches using textual information only

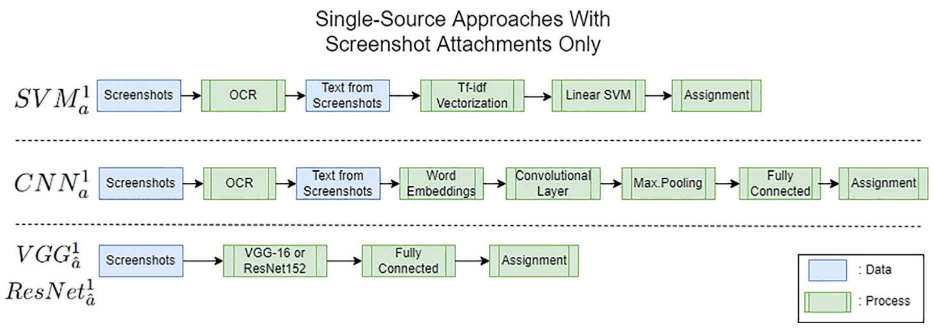


Fig. 6 Proposed single-source approaches using screenshots only

One commonality between these approaches is that they all cast the problem at hand to a classification problem where the class labels to be predicted represent the development teams, to which the issue reports should be assigned. Furthermore, the approaches, which analyze the text extracted from the one-line summaries, descriptions, and/or screenshot attachments, pre-process the text before any analysis. In particular, we first tokenize the words in the extracted text, then eliminate the special characters, such as symbols and punctuation characters, and finally remove the stop-words, which do not contribute to the assignments at all. We then use n-grams, i.e., consecutive sequences of n tokens, to construct our vocabulary. For the text present in the one-line summary and the description fields of the issue reports, n was 2. And, for the text extracted from the screenshot attachments, we opted to have $n = 1$; since the text is extracted by using OCR, consecutive sequences of words do not necessarily represent a coherent context.

The proposed approaches, however, differ from each other in the sources of information they use and in the way they model the classification problem. At a very high level, they can be grouped into three broad categories: *single-source with textual information only* (for short, *single-source-report*), *single-source with screenshot attachments only* (for short, *single-source-attachment*), and *multi-source*.

The single-source models (Sections 5.2 and 5.3) use either the textual information or the screenshot attachments in the issue reports (but not both) for the assignments. We use these models to separately evaluate the individual contributions of the two different sources of information to the assignment accuracy. The single-source models, therefore, serve as baselines for comparative analysis.

More specifically, the single-source-report models use only the one-line summary and the description fields of the issue reports for the assignments (Section 5.2). To this end, we evaluate three models, namely SVM_t^1, CNN_t^1, and RNN_t^1. The former model is a linear support-vector machine (SVM) model (Joachims 1998) trained by using the bag-of-words approach (Manning 2008). As this model represents the currently deployed model in the field (Aktas EU and Yilmaz C 2020a), using it in the comparisons allows us to evaluate the improvements provided by the multi-source models over the existing system. The latter two models, on the other hand, take the contextual information in the issue reports into account by using word embeddings (Goodfellow 2016) with convolutional neural networks (CNN) (Goodfellow 2016) and recurrent neural networks (RNN) (Goodfellow 2016), respectively. We chose these two models for the comparisons as they were recently used in some state-of-the-art issue assignment systems (Lee et al. 2017; Mani et al. 2019).

The single-source-attachment models, on the other hand, use only the screenshot attachments for the assignments (Section 5.3). We experimented with four such models, namely $VGG^1_{\hat{a}}$, $ResNet^1_{\hat{a}}$, SVM^1_a, and CNN^1_a. The former two models ($VGG^1_{\hat{a}}$ and $ResNet^1_{\hat{a}}$) leverage the visual features extracted from the screenshot attachments. To this end, we use the pre-trained and well-known image classification models, namely VGG (Simonyan and Zisserman 2014) and ResNet (He et al. 2016), with transfer learning (Goodfellow 2016).

We, however, conjectured that the visual features might not produce sufficient assignment accuracies. Since the development teams at Softtech use the same UI frameworks, the screens that they produce tend to have the same (or similar) look and feel, making it difficult to use the visual features for the assignments.

To address this issue, we have developed the latter two models (SVM^1_a and CNN^1_a), which extract the text present in the screenshot attachments using OCR and use the extracted text to train the SVM and CNN models, respectively. Note that since the RNN models (i.e., RNN^1_t), compared to the SVM (SVM^1_t) and the CNN models (CNN^1_t), produced inferior text classification results in our experiments (Section 6.3.1), we have decided not to use them in other machine learning pipelines.

The multi-source models, on the other hand, utilize both the textual information and the screenshot attachments present in the issue reports for the assignments (Section 5.1). To this end, we have developed and evaluated a number of different models, namely $CNN\_VGG^2_{t\hat{a}}$, $CNN\_ResNet^2_{t\hat{a}}$, SVM^1_{ta}, SVM^2_{ta}, CNN^1_{ta}, and CNN^2_{ta}.

The first two models ($CNN\_VGG^2_{t\hat{a}}$ and $CNN\_ResNet^2_{t\hat{a}}$) combine the text in the one-line summary and the description fields of the issue reports with the visual features extracted from the screenshot attachments using VGG and ResNet, respectively (Section 5.1.4).

As we have already discussed, however, visual features may be of little help in the Softtech case. Therefore, we have developed the remaining models (SVM^1_{ta}, SVM^2_{ta}, CNN^1_{ta}, and CNN^2_{ta}), such that they extract the text from the screenshot attachments using OCR (rather than extracting the visual features) and then combine it with the text in the issue reports (Sections 5.1.1-5.1.3). The aforementioned models differ from each other not only in the classification models they use (i.e., SVM vs. CNN), but also in the way they treat the text coming from these two different sources of information.

More specifically, the SVM^1_{ta} and CNN^1_{ta} models merge the text coming from both sources into a single text and then analyze it. However, the text extracted from the screenshots and the text present in the one-line summary and the description fields often exhibit different properties. More specifically, the latter is typically written using a formal language with little or no language errors at all (e.g., typos and grammar mistakes). The former, on the other hand, typically has many typos due to the OCR errors. And, interestingly enough, OCR tends to repeatedly make the same or similar mistakes. For example, the same sequence of characters tend to be recognized wrongly in exactly the same manner.

To account for these differences, we have developed two multimodal classification models (SVM^2_{ta} and CNN^2_{ta}). At a very high level, these models treat the text present in the issue reports and the text extracted from the screenshot attachments as text coming from two different channels (Sections 5.1.2 and 5.1.3).

In the experiments (Section 6.3.1), although the multi-source approaches improved the assignment accuracy for the issue reports with screenshot attachments, they tended to slightly reduce the accuracy for the issue reports without any attachments. We believe that this was because having no information flowing through the respective channel in the absence of any attachments tend to make the issue reports close to each other due the aforementioned commonality.

We have, therefore, also developed a hybrid approach (Section 5.4), called SVM_{hybrid}, by combining the best performing multi-source model in the experiments, i.e., SVM^2_{ta}, together with the best performing single-source model, i.e., SVM^1_t.

Next, we present all of these models in detail in an order, which makes the paper easier to read. More specifically, we first introduce the multi-source models (Section 5.1) and then discuss the single-source (Sections 5.2-5.3) and the hybrid models (Section 5.4).

5.1 Multi-Source Approaches

We have developed 6 different multi-source approaches, namely SVM^1_{ta}, SVM^2_{ta}, CNN^1_{ta}, CNN^2_{ta}, $CNN\_VGG^2_{t\hat{a}}$, and $CNN\_ResNet^2_{t\hat{a}}$.

5.1.1 SVM^1_{ta}

In this approach, we first extract the textual information present in the screenshot attachments using optical character recognition (OCR) (Smith 2007) and then merge it (i.e., into a single channel) with the textual information present in the one-line summary and the description fields of the issue reports (Fig. 4).

More specifically, we represent each issue report as a vector in a multi-dimensional space by using the bag of words (BoW) model with the well-known $tf\text{-}idf$ scoring scheme (Manning 2008). Each element in the vectorized form of an issue report, represents a term and the value of the element (i.e., the $tf\text{-}idf$ score of the respective term) depicts the importance of the term for the issue report. The more a term appears in an issue report (i.e., the higher the *term frequency* score tf) and the less it appears in other issue reports (i.e., the higher the *inverse document frequency* score idf), the more important the term becomes for the report (i.e., the higher the $tf\text{-}idf$ score, thus the weight, of the term is).

To train the classification models, we feed the $tf\text{-}idf$ representations of the issue reports to a linear SVM model (Pedregosa et al. 2011). We opted to use the linear SVM models with the BoW representations, because the results of our earlier studies strongly suggest that these models offer us the best prediction accuracy in our industrial setup with manageable training and prediction costs (in terms of the training and prediction times required as well as the amount of training data needed) (Aktas EU and Yilmaz C 2020a).

This is, indeed, the model that has been used by IssueTAG in the field since its deployment (Aktas EU and Yilmaz C 2020a) (Section 3). The difference is that while the deployed system leverages only the textual information present in the issue reports and ignores the attachments, SVM^1_{ta} leverages both sources of information.

Throughout the paper, we train the SVM models by using $scikit-learn$ (Pedregosa et al. 2011) with a linear kernel and extract the text from the screenshots by using $py-tesseract$ (Smith 2007).

5.1.2 SVM^2_{ta}

To account for the different attributes exhibited by the text coming from the two different sources of information, we have developed a multimodal classification approach (SVM^2_{ta}) by treating the text present in the issue reports and text extracted from the screenshot attachments as text coming from two different channels (Fig. 4). More specifically, while the combined text obtained from the one-line summaries and the descriptions forms a channel, the text extracted by using OCR forms another channel.

In this approach, although we encode the information flowing through each channel by using the BoW model with the tf-idf scoring scheme (as explained in Section 5.1.1), we compute the tf-idf scores on a per channel basis. That is, the term frequencies and the inverse document frequencies are computed separately for each channel. Therefore, given an issue report with a screenshot attachment, we compute two vectors (one per channel), which are then appended to each other before being fed to a linear SVM model.

5.1.3 CNN_{ta}^{1} and CNN_{ta}^{2}

The BoW models we used in the first two approaches (Sections 5.1.1 and 5.1.2), do not necessarily take the contexts of the terms appearing in the issue reports into account when making the assignments. To overcome this issue, we, in this section, use deep neural networks to generate word embeddings and use them for the assignments (Lee et al. 2017). In a nutshell, word embeddings are the vectorized forms of the words, such that the vectors (i.e., the embeddings) of the semantically similar (or related) words are close to each other in a multi-dimensional space.

Note that all the issue reports we are dealing with in this work are written in Turkish. Although the language used in these reports are quite formal, the reports include an extensive use of the finance jargon as well as the company jargon, which has been developed over the years with a great deal of abbreviations. We, therefore, chose to train our own word embeddings by using the issue database maintained at Softtech. To this end, we have used the Keras embedding layer (Chollet and et al 2015). In particular, the word embeddings were initialized with random weights and fine-tuned throughout the training process.

Once the word embeddings are learnt, we used them to train convolutional neural networks (CNN), an approach inspired from (Lee et al. 2017), which presents a state-of-the-art application of the word embeddings for issue assignment (Fig. 4). More specifically, for the text flowing through a channel (either from the one line-summaries and descriptions or from the screenshot attachments), we first represent it with the word embeddings, conveying the semantics. We then apply a convolution process using a sample-based discretization approach, called *max-pooling* (Goodfellow 2016). Finally, the outputs are concatenated and, through a fully-connected layer and softmax regression, the probabilities for the assignees are computed. To prevent overfitting, we apply dropout as well as L2 regularization. The interested reader can refer to (Lee et al. 2017) for further details. We, in particular, experiment with two different models, namely CNN_{ta}^{1} and CNN_{ta}^{2}. The former model uses a single channel, into which the text extracted from the screenshots attachments and from the textual information present in the issue reports are merged. The latter model, on the other hand, uses two channels by treating the text extracted from the screenshots and from the issue reports separately.

5.1.4 $CNN\_VGG_{t\hat{a}}^{2}$ and $CNN\_ResNet_{t\hat{a}}^{2}$

All of the approaches we have discussed so far (Sections 5.1.1-5.1.3) leverage the text extracted from the screenshot attachments by using OCR and completely ignore the visual features. The $CNN\_VGG_{t\hat{a}}^{2}$ and $CNN\_ResNet_{t\hat{a}}^{2}$ models, on the other hand, use the visual features extracted from the screenshots together with the text present in the one-line summaries and descriptions of the issue reports (Fig. 4).

At a very high level, these multimodal models build a basic architecture to integrate visual and textual features of the issue reports for our specific classification task. In both models, we use CNN as described in Section 5.1.3 for processing the text present in the issue reports and either the VGG (as is the case with $CNN\_VGG^2_{t\hat{a}}$) (Simonyan and Zisserman 2014) or the ResNet (as is the case with $CNN\_ResNet^2_{t\hat{a}}$) (He et al. 2016) image recognition model for processing the visual features extracted from the screenshots. The extracted textual and visual features are first combined (or fused) together for dimensionality reduction. The results are, then, passed through the activation function ReLU, after which a dropout is applied to forget some of the information learned by the network. The final representations are passed through a fully-connected layer and the softmax function is used for the classification. We use the Keras deep learning framework (Chollet and et al 2015) to build the $CNN\_VGG^2_{t\hat{a}}$ and $CNN\_ResNet^2_{t\hat{a}}$ models.

5.2 Single-Source Approaches using Textual Information Only

We use the approaches discussed in this section to evaluate the effect of the textual information present in the issue reports on the assignment accuracy. To this end, the aforementioned approaches use only the text in the one-line summary and the description fields of the issue reports, and completely ignore the screenshot attachments.

We experiment with three such models: SVM^1_t, CNN^1_t, and RNN^1_t (Fig. 5). The first two models use the SVM and the CNN approaches in the same manner discussed in Sections 5.1.2 and 5.1.3, respectively. The last model, on the other hand, uses the recurrent neural networks (RNNs) (Goodfellow 2016) for the same purpose. At a very high level, the RNN models are a type of artificial neural networks, which have loops to allow previous outputs to be used as inputs while having hidden states, thus exhibiting temporal dynamic behavior. More specifically, the RNN^1_t model uses a bidirectional LSTM (long-short term memory network) (Hochreiter and Schmidhuber 1997; Schuster and Paliwal 1997), which is a type of RNN network where the sequence of information in both directions, i.e., in backward direction from future to past and in forward direction from past to future, is preserved.

5.3 Single Source Approaches Using Attachments Only

While the approaches in Section 5.2 are used to evaluate the amount of information conveyed in the textual descriptions of issue reports, which can be used toward the assignments, the approaches we study in this section carry out the same analysis for the information conveyed in the screenshot attachments. To this end, we use only the screenshot attachments for assigning the issue reports to the stakeholders and completely ignore the one-line summaries and the descriptions of the reports. Note that the approaches we present both in this section and in the previous section (Section 5.2) also serve as a baseline for the multi-source approaches introduced in Section 5.1.

More specifically, we experiment with 4 models: SVM^1_a, CNN^1_a, $VGG^1_{\hat{a}}$, and $ResNet^1_{\hat{a}}$. The first two models extract the textual information present in the attachments using OCR and use the extracted text to train the SVM and CNN approaches in the same manner discussed in Section 5.1.2 and Section 5.1.3, respectively.

The other two approaches ($VGG^1_{\hat{a}}$ and $ResNet^1_{\hat{a}}$), on the other hand, use the visual features (rather than the textual features) extracted from the screenshot attachments. In particular, we use the well-known VGG-16 (Simonyan and Zisserman 2014) and ResNet152 (He et al. 2016) image recognition models implemented with Keras (Chollet and et al 2015). The VGG-16 model includes five convolutional blocks consisted of a total of thirteen convolutional layers, followed by three fully connected layers. The ResNet model, on the other hand, introduces a residual module, which is a block of two convolutional layers where the output of the second layer is added to the input of the first convolutional layer. We also use transfer learning in both models. That is, we fix the weights of all of the convolutional layers during training, replace the last fully connected layer with the new fully connected layer, and train only the newly added layer for assigning the issue reports. The first layer after the convolutional layers, flattens the input vector to obtain a one-dimensional vector, then a dense layer is used to reduce the dimension of the vector where a ReLU function is applied, and finally a fully connected layer and the softmax function is used for the classification.

5.4 Hybrid Approach

To account for the differences between the single- and multi-source models for the issue reports with and without screenshot attachments, we also develop a hybrid approach, called SVM_{hybrid}. To this end, we combine the best performing multi-source model in the experiments, i.e., SVM^2_{ta}, together with the best performing single-source model, i.e., SVM^1_t (Section 6). More specifically, we use the SVM^2_{ta} model for assigning the issue reports with screenshot attachments and the SVM^1_t model for assigning the ones without any attachments.

6 Experiments

To evaluate the proposed approaches, we have carried out a series of experiments. In these experiments, to study the individual contributions of the two different sources of information to the assignment accuracy, we use the single-source models, namely SVM^1_t, CNN^1_t, RNN^1_t, SVM^1_a, CNN^1_a, $VGG^1_{\hat{a}}$, and $ResNet^1_{\hat{a}}$. These models, indeed, serve as the baselines for the comparative studies.

Among the single-source models, SVM^1_t, CNN^1_t, and RNN^1_t are used to evaluate the contribution of the text present in the one-line summary and the description fields of the issue reports, whereas the SVM^1_a, CNN^1_a, $VGG^1_{\hat{a}}$, and $ResNet^1_{\hat{a}}$ models are used to evaluate the contribution of the screenshot attachments. Based on the level of assignment accuracy obtained by the deployed IssueTAG system, which uses only the former source of information, we conjecture that the former models would generally perform better than the latter ones.

In the presence of screenshot attachments, to evaluate whether the textual or the visual features extracted from the attachments would lead to better assignment accuracies, we use the SVM^1_a and CNN^1_a models with the textual features and the $VGG^1_{\hat{a}}$ and $ResNet^1_{\hat{a}}$ models with the visual features. Since the UIs produced by different development teams at Softtech tend to have the same or similar look and feel, we conjecture that the former models would generally perform better (see Section 4 for more information).

We, furthermore, use different types of ML models with the same set of features to avoid any bias in the analysis as much as possible. We, indeed, do this not only for the single-source models, but also for the multi-source models. For example, for the single-source models leveraging the textual features in the issue reports, we experiment with SVM (SVM^1_t), CNN (CNN^1_t), and RNN (RNN^1_t). And, for the single-source models leveraging the visual features in the screenshot attachments, we experiment with VGG ($VGG^1_{\hat{a}}$) and ResNet ($ResNet^1_{\hat{a}}$). We, in particular, chose these models because they are well-known models, which have been successfully used in the literature for the same or similar tasks

(Jonsson et al. 2016; Lee et al. 2017; Chen et al. 2019a; 2019b; Mani et al. 2019; Aktas EU and Yilmaz C 2020a; Simonyan and Zisserman 2014; He et al. 2016).

To evaluate whether leveraging both sources of information can improve the assignment accuracies, compared to using a single source of information, we use the multi-source models (namely, SVM_{ta}^1, SVM_{ta}^2, CNN_{ta}^1, CNN_{ta}^2, $CNN\_VGG_{t\hat{a}}^2$, and $CNN\_ResNet_{t\hat{a}}^2$). To this end, we opted not to further experiment with the single-source models, which produced inferior results, in our multi-source architectures. For example, we chose not to use RNN in the multi-source models because RNN (as a single-source model) provided lower assignment accuracies, compared to SVM and CNN. Based on the results of the feasibility studies we carried out in Section 4, we conjecture that multi-source models would generally produce better assignment accuracies, especially for the issue reports with screenshot attachments.

To study whether using the textual or the visual features extracted from the screenshot attachments in the multi-source models, would produce better assignment accuracies, we use the SVM_{ta}^1, SVM_{ta}^2, CNN_{ta}^1, and CNN_{ta}^2 models with the former set of features and the $CNN\_VGG_{t\hat{a}}^2$ and $CNN\_ResNet_{t\hat{a}}^2$ models with the latter set of features. As was also the case with the single-source models, we conjecture that using the text extracted from the screenshot attachments in the multi-source models would perform better (Section 4).

The text extracted from the screenshot attachments can be used in two different ways with the multi-source models. One way is to simply combine it with the text extracted from the issue reports (Sections 5.1.1 and 5.1.3). The other way is to treat the text coming from the issue reports and that coming from the screenshot attachments separately (Sections 5.1.2 and 5.1.3). In the experiments, we use the SVM_{ta}^1 and CNN_{ta}^1 models to evaluate the former approach and the SVM_{ta}^2 and CNN_{ta}^2 models to evaluate the latter approach. We conjecture that the latter models would produce better assignment accuracies as the characteristics of the text extracted from the issue reports differ from those of the text extracted from the screenshot attachments due to the inaccuracies of OCR (Section 5).

We evaluate all of the proposed models on real issue reports submitted to Softtech. We analyze both the correctness of the assignments by using the well-known accuracy, precision, recall, and F-measure metrics and the runtime performance of the models by using the training and the prediction times of the models. The results of these studies, indeed, support our basic hypotheses. We also carry out a series of statistical significance tests to figure out whether the differences between the proposed models are statistically meaningful.

6.1 Subject Issue Reports

In the experiments, we used the real issue reports submitted to Softtech. Table 2 presents the summary statistics for these issue reports.

For the initial set of experiments, where the goal was to evaluate all the proposed approaches introduced in Section 5, we used a total of 41,042 issue reports submitted to 63 distinct development teams between the months of March and August in 2019 (Table 2). In particular, we utilized the issue reports resolved in August as the test set and all the remaining issue reports resolved from March to July as the training set.

After determining the best performing multi-source and single-source approaches, we performed a series of statistical significance tests to figure out whether the differences between these approaches are statistically meaningful. To this end, we used an additional 43,930 issue reports submitted to 60 distinct teams between September, 2018 and February, 2019 (Table 2).

Table 2 Summary statistics for the issue reports used in the experiments

Month of creation	Issue reports with screenshots	Issue reports with attachments	Issue reports without attachments	Total	Distinct assignees
August, 2019	2,589	3,080	1,673	4,753	49
July, 2019	4,175	4,861	2,298	7,159	53
June, 2019	3,015	3,584	1,750	5,334	51
May, 2019	4,515	5,326	2,550	7,876	53
April, 2019	4,379	5,260	2,338	7,598	58
March, 2019	4,889	5,841	2,481	8,322	52
Initial set total	23,562	27,952	13,090	41,042	63
February, 2019	3,838	4,890	2,058	6,948	52
January, 2019	4,951	6,180	2,700	8,880	49
December, 2018	3,741	4,316	1,945	6,261	47
November, 2018	4,349	5,065	2,293	7,358	49
October, 2018	3,976	4,688	2,368	7,056	49
September, 2018	4,196	5,052	2,375	7,427	49
Additional set total	25,051	30,191	13,739	43,930	60
Grand total	48,613	58,143	26,829	84,972	68

We made sure that all of the issue reports used in the analyses (i.e., the ones mentioned above) were closed with the "resolved" status. This guaranteed that all of the selected reports actually indicated real issues and that the development teams closing the reports were the correct assignees for the respective reports.

Last but not least, we automatically identify the screenshot attachments in the issue reports by examining their file extensions. More specifically, the attachments with the well-known image extensions, including *.jpg*, *.png*, and *.tif*, are considered to be screenshot attachments. In the experiments, although the issue reports with non-screenshot attachments were also present in the data sets we used (as they were resolved during the time period studied in the experiments), no information was extracted from these attachments (i.e., non-screenshot attachments were simply ignored).

6.2 Evaluation Framework

We used a total of 84,972 real issue reports submitted to 68 distinct teams for the evaluations (Table 2).

To evaluate the correctness of the assignments (thus, to address our second research question), we have computed both the accuracy and the F-measure metrics for the assignments. More specifically, the accuracy (A) was computed as the ratio of the number of correctly assigned issue reports to the total number of issue reports in the test set. And, F-measure (F) was computed as the harmonic mean of the precision (P) and recall (R), giving equal importance to both metrics. For a given development team (i.e., for a given class), the precision of the assignments is computed as the ratio of the number of correctly assigned reports to the team to the total number of issue reports assigned to the team. The recall is, on the other hand, computed as the ratio of the number of correctly assigned reports to the team to the total number of issue reports that should have been assigned to the team. Since multiple classes were present in the experiments, we, in this work, report the weighted values. All of the aforementioned metrics take on a value between 0 and 1, inclusive. The larger the value, the better the assignments, thus the proposed approaches, are.

For the statistical significance tests, we have repeated the experiments 30 times for each experimental setup by utilizing different training and/or test sets (Section 6.3.1). In each repetition, we preserved the time information when partitioning the issue reports into training and test sets. That is, the creation dates for the issue reports included in the training sets were always earlier than those in the test sets.

Furthermore, since this work targets a system operating in a production environment, excessive runtime overheads are simply not acceptable. To evaluate the performance of the proposed approaches (thus, to address our third research question), we also measure the running times of the important tasks. We, in particular, measure the average amount of time required for both training the classification models and using them for predictions as well as the average running times required for extracting the text from screenshot attachments using OCR.

Almost all of the scripts we used in the experiments to implement our machine learning pipelines are publicly available [3] [4]. We can, however, publish neither the scripts regarding the SVM^2_{ta} model nor the issue reports (including the screenshots) used in the experiments due to certain security and privacy issues.

[3]https://github.com/ethemutku/issueTriageWithScreenshots

[4]https://figshare.com/projects/issueTriageWithScreenshots/144702

6.3 Data and Analysis

We have carried out all the experiments required to address our second and third research questions. Note that our first research question has already been addressed in Section 4.

6.3.1 Regarding RQ2: How can the screenshot attachments in issue reports be used to further improve the accuracy of the assignments?

Table 3 summarizes the results of the experiments we performed to address this research question. Next, we answer each of the sub-research questions from RQ2.1 to RQ2.5 separately.

Regarding RQ2.1: How do using the visual features and using the textual features extracted from the screenshot attachments compare in terms of the accuracy of the assignments obtained? Comparing the single-source models, which use only the screenshot attachments for the assignments (i.e., the single-source-attachment models), with each other, we first observed that using the visual features extracted from the screenshot attachments (i.e., $VGG_{\hat{a}}^1$ and $ResNet_{\hat{a}}^1$) did not perform well. More specifically, the overall assignment accuracies obtained from the $VGG_{\hat{a}}^1$ and the $ResNet_{\hat{a}}^1$ models were 0.046 and 0.386, respectively.

We believe that this was mainly due to the fact that a small number of user interface (UI) frameworks have been used throughout Softtech together with a set of quite strict guidelines regarding the UI designs, including the color palette to use and the general design templates to follow. Consequently, the screens produced by different development teams typically have the same or similar look-and-feel, which makes it quite difficult to distinguish between the producers of these screens by using only the visual features.

Note that the models that require the presence of screenshot attachments in order to operate, such as $VGG_{\hat{a}}^1$, cannot be evaluated on the issue reports without any attachments; explaining the missing values, i.e., the "-" symbols, in Table 3. We, therefore, report the accuracy of these models only for the issue reports with screenshot attachments.

We next observed that using the textual features present in the screenshots, compared to using the visual features, were profoundly better at making accurate assignments. More specifically, the best accuracy obtained from the single-source-attachment models, i.e., an accuracy of 0.705 (Table 3), was obtained from the SVM_a^1 model, which leverages the text extracted from the screenshots for the assignments. Indeed, these results further support our claims that the text present in the screenshot attachments convey information, which can be leveraged for issue assignment.

Regarding RQ2.2: In the presence of screenshot attachments, how much information does the text present in the one-line summary and the description fields of the issue reports still provide for the assignments? We then observed that, even in the presence of screenshot attachments, the textual information present in the one-line summary and the description fields of the issue reports were still quite valuable for the assignments. Among all the single-source-report models, the best accuracy for the issue reports with screenshot attachments, which is 0.843, was obtained from the SVM_t^1 model. The accuracies obtained from the CNN_t^1 and RNN_t^1 models were 0.819 and 0.795, respectively.

We believe that this was mainly due to the fact that, in the software systems maintained by Softtech, developing a single screen typically requires the involvement of multiple teams. For example, a screen associated with the credit card operations, which is maintained by

Table 3 Accuracy (A), precision (P), recall (R), and F-measure (F) values obtained from different approaches where the best values are reported in bold

Approach	Model	Test data w/o screenshots				Test data w/ screenshots				all test data			
		A	P	R	F	A	P	R	F	A	P	R	F
Multi-source	SVM_{ta}^1	0.844	0.851	0.844	0.837	0.821	0.814	0.821	0.812	0.832	0.826	0.832	0.823
	SVM_{ta}^2	0.848	0.855	0.848	0.840	**0.858**	**0.851**	**0.858**	**0.848**	0.854	**0.850**	0.854	**0.846**
	CNN_{ta}^1	0.825	0.831	0.825	0.819	0.789	0.794	0.789	0.779	0.819	0.810	0.819	0.804
	CNN_{ta}^2	0.819	0.826	0.819	0.812	0.833	0.820	0.833	0.821	0.826	0.827	0.826	0.819
	$CNN\_VGG_{t\hat{a}}^2$	-	-	-	-	0.822	0.823	0.822	0.815	0.822	0.823	0.822	0.815
	$CNN\_ResNet_{t\hat{a}}^2$	-	-	-	-	0.823	0.829	0.823	0.818	0.823	0.829	0.823	0.818
Single-source w/ textual information (single-source-report models)	SVM_t^1	**0.851**	**0.860**	**0.851**	**0.845**	0.843	0.836	0.843	0.834	0.848	0.845	0.848	0.839
	CNN_t^1	0.826	0.839	0.826	0.825	0.819	0.817	0.819	0.812	0.828	0.818	0.828	0.816
	RNN_t^1	0.800	0.806	0.800	0.805	0.795	0.800	0.795	0.788	0.802	0.801	0.802	0.791
Single-source w/ attachments (single-source-attachment models)	SVM_a^1	-	-	-	-	0.705	0.701	0.705	0.690	0.705	0.701	0.705	0.690
	CNN_a^1	-	-	-	-	0.696	0.712	0.696	0.684	0.696	0.712	0.696	0.684
	$VGG_{\hat{a}}^1$	-	-	-	-	0.046	0.008	0.046	0.007	0.046	0.008	0.046	0.007
	$ResNet_{\hat{a}}^1$	-	-	-	-	0.386	0.402	0.386	0.345	0.386	0.402	0.386	0.345
Hybrid	SVM_{hybrid}	**0.851**	**0.860**	**0.851**	**0.845**	**0.858**	**0.851**	**0.858**	**0.848**	**0.855**	**0.850**	**0.855**	**0.846**

Note that the approaches that require the presence of screenshot attachments cannot be evaluated on the issue reports with no attachments the credit cards team, can use services in the background, which are developed by the customer information management team and the commercial/individual credit team. Therefore, a failure observed on this screen may be caused from any of these services, which are maintained by different development teams. Consequently, given a screen, without knowing the symptoms of the reported issue, it may not be possible to determine the development team responsible for resolving the reported issue. And, the symptoms are typically specified in the one-line summary and the description fields of the issue reports.

Note that, in the analysis above, we used only the issue reports with screenshot attachments, so that different approaches could fairly be compared by using exactly the same set of issue reports. Note further that the ultimate goal of our experiments is to evaluate the effect of using the screenshot attachments for the assignments. Consequently, in the remainder of the analysis, we (unless otherwise stated) focus on the results obtained from the issue reports with screenshot attachments. We, however, still report the overall accuracy of the proposed models in Table 3, because the assignment accuracies of the models should not adversely be affected in the absence of screenshot attachments.

Regarding RQ2.3: How do the multi-source models compare to single-source models in terms of the assignment accuracies they provide? Comparing the multi-source models with the single-source-attachment models, which use only the attachments for the assignments, we observed that all of the multi-source models performed better than the single-source-attachment models (Table 3). While the minimum accuracy obtained from the multi-source models was 0.789, the maximum accuracy obtained from the single-source-attachment models was 0.705.

Comparing the multi-source models with the single-source-report models, which use only the text present in the issue reports, we observed that all of the two-channel multi-source models performed better than their single-source counterparts (Table 3). To this end, we compare SVM_t^1 to SVM_{ta}^2 and CNN_t^1 to $CNN\_VGG_{t\hat{a}}^2$, $CNN\_ResNet_{t\hat{a}}^2$, and CNN_{ta}^2. Note that the only difference between the models compared with each other, is the use of the attachments for the assignments. Otherwise, the pair of single- and multi-source models being compared employ the same approach (i.e., either CNN or SVM) for processing the text present in the issue reports.

More specifically, using the visual features extracted from the screenshot attachments slightly increased the assignment accuracies, compared to completely ignoring the attachments. The assignment accuracies obtained from the $CNN\_VGG_{t\hat{a}}^2$ and $CNN\_ResNet_{t\hat{a}}^2$ models were 0.822 and 0.823, respectively, whereas that obtained from the CNN_t^1 model was 0.819.

However, extracting the text from the screenshot attachments and combining it with the text present in the issue reports by treating each source of information as separate channels, provided better assignment accuracies. For example, the accuracy of the CNN_{ta}^2 model was 0.833, which was better than the ones obtained from the $CNN\_VGG_{t\hat{a}}^2$ and $CNN\_ResNet_{t\hat{a}}^2$ models (i.e., 0.822 and 0.823).

Indeed, the best-performing model was SVM_{ta}^2 with an accuracy of 0.858. Note that the accuracy of the corresponding single-source model, i.e., SVM_t^1, which represents the deployed model in IssueTAG, was 0.843.

We also observed that the SVM models (i.e., SVM_{ta}^1 and SVM_{ta}^2) generally performed better than the CNN models (i.e., CNN_{ta}^1 and CNN_{ta}^2), a phenomenon we observed a number of times in our previous works when it comes to analyzing the issue report repository of Softtech (Aktas et al. 2020c). The best accuracy obtained from the former models was 0.858, whereas that obtained from the latter models was 0.833 (Table 3).

We believe that this was because, when the CNN models are used for text classification, they are typically used with the word embeddings that were trained on large corpora. In our case, however, the issue reports are written in Turkish and they tend to have a heavy use of the company-specific jargon with a great deal of abbreviations. We, therefore, trained our own word embeddings, which resulted in the assignment accuracies no better than the ones obtained from the SVM models using the bag-of-words approach.

Last but not least, using a two-channel multi-source model to combine the text coming from the issue reports and the text coming from the screenshot attachments, performed better than using a single-channel multi-source model. While the accuracy of SVM_{ta}^2 was 0.858, that of SVM_{ta}^1 was 0.821. Similarly, the accuracy of CNN_{ta}^2 was 0.833, whereas that of CNN_{ta}^1 was 0.789.

Regarding RQ2.4: Does the assignment accuracy obtained from the multi-source model get affected by the absence of the screenshot attachments? When we compared the assignment accuracy of the best performing multi-source model, i.e., SVM_{ta}^2, to that of the best performing single-source model, i.e., SVM_t^1, for the issue reports without any screenshot attachments, we observed the SVM_{ta}^2 model slightly reduced the assignment accuracy; 0.851 vs. 0.848. We believe that this is because, in the absence of any attachments, having no information flowing through the respective channel in the multi-source models tend to make the issue reports close to each other.

Using our hybrid model SVM_{hybrid}, on the other hand, resolved this issue. Although the SVM_{hybrid} model provided a similar overall accuracy with the SVM_{ta}^2 model (0.855 vs. 0.854), the former prevented the assignment accuracy of the issue reports without any screenshot attachments from suffering (Table 3). In particular, compared to using SVM_{ta}^2 for the issue reports without any attachments, using SVM_{hybrid} (as it actually leverages the SVM_t^1) model for these reports) increased the accuracy from 0.848 to 0.851.

Regarding RQ2.5: Are the differences between the assignment accuracies obtained from the best performing multi-source and single-source models statistically significant? To further compare the assignment accuracies obtained from the best performing multi-source models (i.e., SVM_{ta}^2 and SVM_{hybrid}) to those obtained from the best performing single-source model (i.e., SVM_t^1), we carried out an additional set of experiments. Note that the SVM_t^1 model is also the currently deployed model in the field.

In the first set of experiments, we used exactly the same test set with the experiments discussed above (i.e., all the issue reports resolved in August 2019). We, however, varied the training set by choosing a subset of all the issue reports resolved within the last 6 months of August 2019. We did this in such a way that the training and test sets correspond to the 80% and 20% of all of the issue reports selected, respectively. We, furthermore, repeated the experiments 30 times.

Figure 7 presents the distributions of the accuracies obtained from the SVM_t^1, SVM_{ta}^2, and SVM_{hybrid} models. Furthermore, Table 4 reports the summary statistics for these results (Look for the month of August). Note that we report the results of SVM_{hybrid} for the entire test set only as this model uses either the SVM_t^1 or the SVM_{ta}^2 model for the assignments, depending on the presence of the attachments.

We observed exactly the same trends with our original experiments. In particular, SVM_t^1 generally performed better than SVM_{ta}^2 for the issue reports without any attachments; an

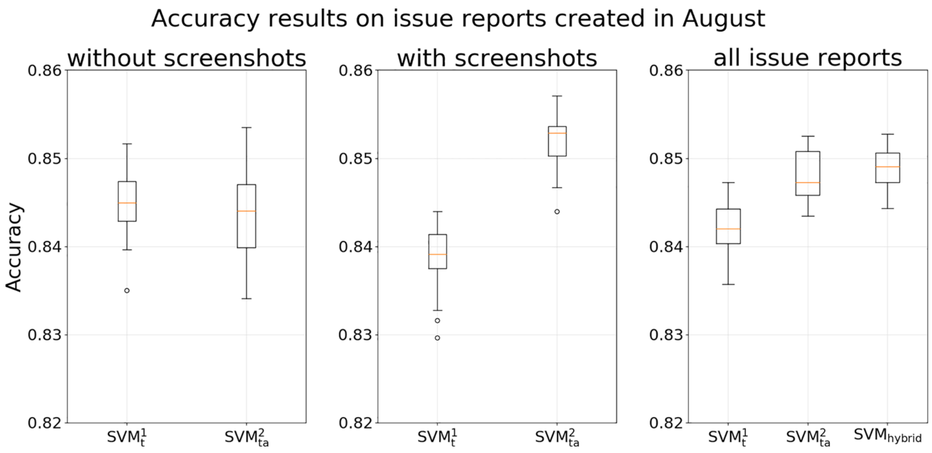


Fig. 7 Box-whisker plots of the accuracies obtained on the issue reports resolved in August 2019. The plots (from left right) present the distributions of the results obtained from the issue reports without and with attachments, and those obtained from all the issue reports in the test set, respectively. For each category, the experiments were repeated 30 times

average accuracy of 0.845 vs. 0.844. For the issue reports with attachments, on the other hand, SVM_{ta}^2 provided better assignment accuracies; an average accuracy of 0.852 vs. 0.839

Overall, i.e., when all the issue reports with and without screenshot attachments are taken into account, SVM_{hybrid} performed better than the currently deployed model in the field (i.e., SVM_t^1); an average accuracy of 0.849 vs. 0.842 (Table 4).

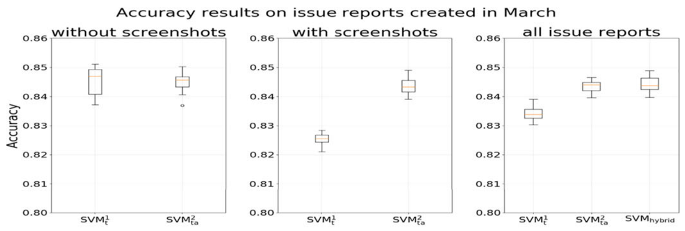
In the second set of experiments, we have also varied the test sets. In particular, we repeated experiments we carried out for August for each remaining month m from March to July (inclusive) by using all the data resolved in the month of m as the test set and by randomly picking a training set from the issue reports resolved within the last 6 months of m, such that the training and test sets represent 80% and 20% of all the issue reports selected, respectively. The experiments were again repeated 30 times for each experimental setup. For this set of experiments, we used an additional set of 43,930 distinct issue reports (Table 2).

Figure 8 presents the results we have obtained. We observed exactly the same trends with the previous set of experiments: 1) SVM_t^1 generally performed better than SVM_{ta}^2 for the issue reports without any screenshot attachments; 2) for the ones with screenshot attachments, however, SVM_{ta}^2 performed better than SVM_t^1; and 3) overall, SVM_{hybrid} was the best performing model.

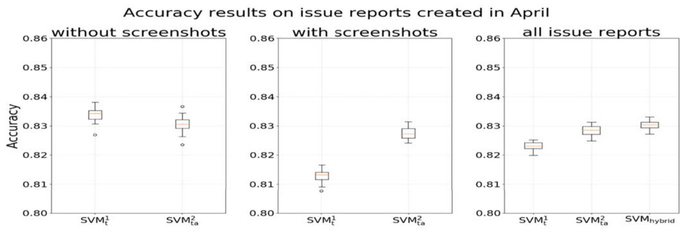
We have then used the non-parametric Wilcoxon rank sum test (Wilcoxon 1992) for the analysis of statistical significance. To this end, we carried out pairwise tests between the SVM_t^1, SVM_{ta}^2, and SVM_{hybrid} models by using the data set summarized in Table 2. The results of these tests can be found in Table 5 where the entries in bold represent the statistically significant results with the significance level of 0.05. All the differences in the assignment accuracies, except for 2 (out of 30), were found to be statistically significant. Since the study involved multiple comparisons of hypotheses, we have also applied the Bonferroni correction (Weisstein 2004) where we decreased the significance level to 0.0017 (i.e., 0.05/30 as we have 30 comparisons). Out of 30 comparisons, 24 of them were found to be statistically significant at the adjusted significance level, including all the comparisons

Table 4 The summary statistics for the accuracy (A) and the measure (F) values obtained in different experimental setups where the best values are reported in bold

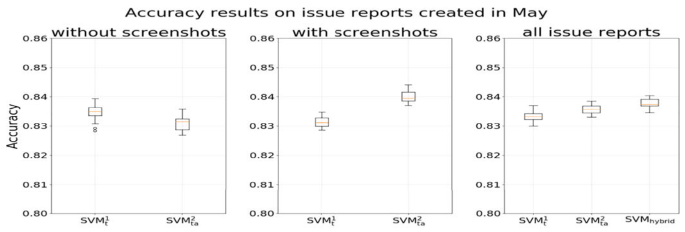
Month	Stat	Test set w/o screenshots				Test set w/ screenshots				All test set					
		SVM_t^1		SVM_{ta}^2		SVM_t^1		SVM_{ta}^2		SVM_t^1		SVM_{ta}^2		SVM_{hybrid}	
		A	F	A	F	A	F	A	F	A	F	A	F	A	F
August	mean	**0.845**	**0.836**	0.844	0.834	0.839	0.826	**0.852**	**0.839**	0.842	0.831	0.848	**0.837**	**0.849**	**0.837**
	std.	0.004	0.004	0.005	0.005	0.004	0.004	0.003	0.003	0.003	0.003	0.003	0.003	0.002	0.002
	max	0.852	**0.843**	**0.854**	**0.843**	0.844	0.833	**0.857**	**0.845**	0.847	0.836	**0.853**	**0.842**	**0.853**	**0.842**
	min	**0.835**	**0.826**	0.834	0.821	0.830	0.817	**0.844**	**0.829**	0.836	0.825	**0.844**	0.832	**0.844**	**0.833**
July	mean	**0.851**	**0.835**	0.847	0.830	0.843	0.832	**0.853**	**0.842**	0.846	0.833	0.850	0.838	**0.852**	**0.840**
	std.	0.003	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
	max	**0.856**	**0.839**	0.851	0.835	0.849	0.837	**0.858**	**0.847**	0.850	0.836	0.855	**0.842**	**0.856**	**0.842**
	min	**0.845**	**0.830**	0.843	0.827	0.837	0.826	**0.848**	**0.836**	0.841	0.828	0.846	0.835	**0.848**	**0.836**
June	mean	**0.831**	**0.822**	0.828	0.819	0.835	0.820	**0.841**	**0.826**	0.834	0.822	0.835	0.824	**0.836**	**0.825**
	std.	0.004	0.004	0.004	0.004	0.003	0.003	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002
	max	**0.840**	**0.830**	0.834	0.827	0.842	0.827	**0.845**	**0.831**	0.840	0.828	0.840	0.829	**0.841**	**0.830**
	min	**0.823**	**0.816**	0.818	0.810	0.830	0.815	**0.835**	**0.821**	0.828	0.817	0.831	**0.821**	**0.832**	**0.821**
May	mean	**0.835**	**0.818**	0.831	0.814	0.831	0.818	**0.840**	**0.827**	0.833	0.819	0.836	0.821	**0.838**	**0.823**
	std.	0.003	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
	max	**0.839**	**0.823**	0.836	0.818	0.835	0.822	**0.844**	**0.831**	0.837	0.822	0.839	0.824	**0.840**	**0.826**
	min	**0.828**	**0.812**	0.827	0.809	0.829	0.816	**0.837**	**0.824**	0.830	0.815	0.833	0.819	**0.835**	**0.820**
April	mean	**0.834**	**0.814**	0.831	0.809	0.813	0.800	**0.828**	**0.813**	0.823	0.806	0.828	0.811	**0.830**	**0.813**
	std.	0.003	0.003	0.003	0.003	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
	max	**0.838**	**0.818**	0.837	0.816	0.817	0.803	**0.832**	**0.817**	0.825	0.808	0.831	0.814	**0.833**	**0.816**
	min	**0.827**	**0.808**	0.824	0.802	0.808	0.794	**0.824**	**0.810**	0.820	0.802	0.825	0.808	**0.827**	**0.810**
March	mean	**0.845**	**0.829**	**0.845**	0.828	0.825	0.813	**0.844**	**0.832**	0.834	0.820	**0.844**	0.830	**0.844**	**0.831**
	std.	0.004	0.004	0.003	0.003	0.002	0.002	0.003	0.003	0.002	0.002	0.002	0.002	0.003	0.003
	max	**0.851**	**0.835**	0.850	0.833	0.828	0.816	**0.849**	**0.839**	0.839	0.825	0.847	0.833	**0.849**	**0.836**
	min	**0.837**	**0.823**	**0.837**	0.822	0.821	0.809	**0.839**	**0.826**	0.830	0.816	**0.840**	**0.826**	**0.840**	**0.826**



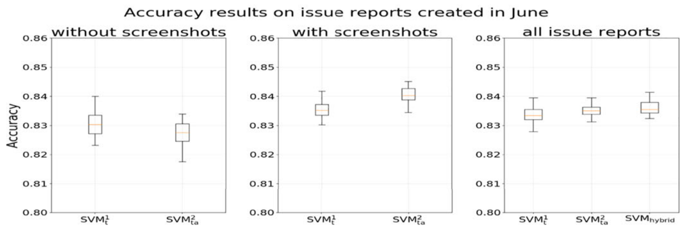
(a)



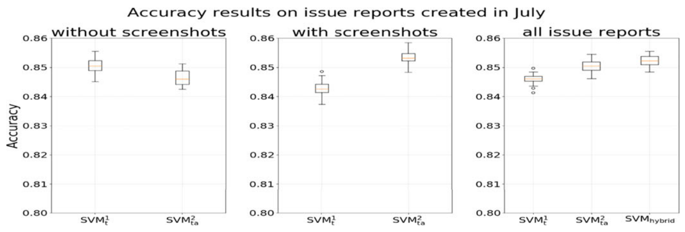
(b)



(c)



(d)



(e)

Fig. 8 Box-whisker plots of the accuracies obtained for the issue reports resolved in a) March, b) April, c) May, d) June, and e) July of 2019. For each category (i.e., for each box plot), the experiments were repeated 30 times

Table 5 The results of the non-parametric Wilcoxon rank-sum tests. The summary statistics for the accuracy values used in the comparisons are given in Table 4

Model 1	Model 2	Test set	August		July		June		May		April		March	
			p-value	effect size	p-value	effect size	p-value	effect size	p-value	effect size	p-value	effect size	p-value	effect size
SVM_t^1	SVM_{ta}^2	w/ screenshots	**1.731e-6**	0.62	**1.724e-6**	0.62	**4.273e-6**	0.59	**1.732e-6**	0.62	**1.727e-6**	0.62	**1.731e-6**	0.62
SVM_t^1	SVM_{ta}^2	w/o screenshots	6.260e-2	0.24	**1.726e-6**	0.62	**2.596e-4**	0.47	**1.721e-6**	0.62	**6.941e-6**	0.58	**1.727e-6**	0.62
SVM_t^1	SVM_{ta}^2	all	**1.725e-6**	0.62	**1.732e-6**	0.62	**1.953e-3**	0.40	**5.175e-6**	0.59	**1.733e-6**	0.62	**3.967e-2**	0.27
SVM_t^1	SVM_{hybrid}	all	**1.726e-6**	0.62	**1.733e-6**	0.62	**3.100e-5**	0.54	**1.729e-6**	0.62	**1.730e-6**	0.62	**1.719e-6**	0.62
SVM_{ta}^2	SVM_{hybrid}	all	**1.622e-2**	0.31	**2.841e-6**	0.60	**6.276e-3**	0.35	**1.725e-6**	0.62	**2.544e-6**	0.61	5.927e-1	0.07

A p-value less than 0.05 is considered to be statistically significant, which is reported in bold between the SVM_t^1 and SVM_{hybrid} models (Table 5). Note that, in all of these 24 cases, the effect sizes were between 0.47 and 0.62 with one "moderate effect" and 23 "large effects" according to the Cohen's classification of effect sizes (Cohen 1988).

The results of these statistical tests should, however, be interpreted with care. Since the training sets we used in the analysis shared some common issue reports (although the test sets did not suffer from this problem), the assumption of independence in the significance tests was violated.

This is, indeed, a common issue when the statistical hypothesis tests are used to compare different machine learning models (Dietterich 1998; Bouckaert and Frank 2004). More specifically, in the presence of a single data set (or limited amount of data), which is also the case in this work, some resampling methods (such as, n-fold cross-validation) may need to be employed for the analysis. And, when the same observations end up being used in multiple training and/or test sets, the assumption of independence is violated. Therefore, it is very well known that the correct use of the statistical tests is quite challenging in this context, especially in the absence of a large volume of data, which allows several independent data sets to be used for the analysis (Dietterich 1998; Bouckaert and Frank 2004; Witten et al. 2005). From this perspective, our work is no exception.

Nevertheless, some approaches have been proposed in the literature to alleviate these issues to the extent possible (Dietterich 1998; Bouckaert and Frank 2004; Nadeau and Bengio 1999). Many of these approaches, however, assume that multi-fold cross-validation is an appropriate resampling method for the data set at hand (Dietterich 1998; Nadeau and Bengio 1999). At a very high level, these approaches repeat the cross-validation experiments and analyze the results by using some modified hypothesis tests, which account for the lack of independence (Dietterich 1998; Nadeau and Bengio 1999). We, on the other hand, could not use these approaches as cross-fold validation is not a valid resampling method in our case. More specifically, there is a time dependency between the training and the test sets we can use; all the issue reports in the training set should have been resolved before the issue reports in the test set, thus invaliding the use of multi-fold cross-validation.

An alternative approach would be to evaluate the replicability of the significance tests (Bouckaert and Frank 2004). However, we could not use this approach either as it requires a large number of independent data sets for the analysis, such as analyzing the issue reports submitted to multiple different companies, which was not feasible for us.

We are, therefore, left with the McNemar's test, which is, indeed, frequently used in the presence of a single data set (Everitt 1992; Dietterich 1998). We applied this test to our original data set where we used all the issue reports resolved between February to July as the training set and those resolved in August as the test set.

Table 6 presents both the contingency table and the p-values we obtained. Considering a p-value less than 0.05 to be statistically significant, we observed that both the SVM_{ta}^2 and SVM_{hybrid} models performed significantly better than the deployed SVM_t^1 model, except for one case. The exceptional case occurred in the absence of the screenshot attachments where, as expected, the difference between the SVM_{ta}^2 and SVM_t^1 models was not significant.

6.3.2 Regarding RQ3: How does taking the screenshot attachments into account affect the overall performance in terms of the training and the prediction times?

To address this research question, we compared the performance of our best performing multi-source model (i.e., SVM_{ta}^2) with that of the deployed model in the field (i.e., SVM_t^1). All the experiments were carried out on a Dual-Core Intel(R) Core(TM) i7-6600U CPU

Table 6 The contingency table and the p-values obtained when the McNemar test is applied; a p-value less than 0.05 is considered to be statistically significant, which is reported in bold

Model 1	Model 2	Test set	number of assignments where				p-values
			Model 1 is correct Model 2 is correct	Model 1 is correct Model 2 is incorrect	Model 1 is incorrect Model 2 is correct	Model 1 is incorrect Model 2 is incorrect	
SVM_t^1	SVM_{ta}^2	w/ screenshots	2150	60	96	283	**0.00491**
SVM_t^1	SVM_{ta}^2	w/o screenshots	1843	18	20	283	0.87141
SVM_t^1	SVM_{ta}^2	all	3993	78	116	566	**0.00773**
SVM_t^1	SVM_{hybrid}	all	4011	60	96	586	**0.00491**

@2.60 GHz computer with 16GB of RAM running Windows 10 Enterprise 2017 as the operating system. In particular, we used three metrics for the comparisons; OCR time, response time, and training time. The first metric measures the end-to-end processing time required for extracting the text from a single screenshot attachment, including the time required for loading the attachment to the memory. We report this metric on a per screenshot attachment basis as this step can easily be parallelized; multiple attachments can be processed in parallel. The second metric, i.e., the response time of a model, is measured as the end-to-end time required for assigning a given issue report. Note that the response times also include the OCR times (for SVM_{ta}^2), because the text in a given screenshot attachment needs to be extracted before the assignment can be made. And, the third metric, i.e., the training time of a model, is measured as the time it takes to train the model once all the data to flow through the channels is fed as input. That is, OCR times are not included in the training times. The reason for this is two folds. First, the model employed by IssueTAG is re-trained as needed (Aktas EU and Yilmaz C 2020a). Therefore, the texts extracted from the screenshot attachments for the purpose of assigning the respective issue reports to the development teams, can be saved to re-train the model in the future. That is, the OCR step needs to be carried out only once during the assignment, which is, indeed, accounted for in the response times. Second, as discussed before, the OCR step can easily be parallelized (e.g., for the training of the very first model).

Table 7 presents the time measurements (in seconds) we obtained by repeating each experiment at least 30 times. We observed that, although taking the screenshot attachments into account when assigning the issue reports, expectedly increased the training and the response times, all of these overheads were acceptable at Softtech. More specifically, the SVM_{ta}^2 model, compared to the SVM_t^1 model, increased the average training time from 190.4 to 317.2 seconds and the average response time from 0.9 to 2.17 seconds. A significant portion of the response time for the SVM_{ta}^2 model (2.11 out 2.17) was indeed spent for OCR.

Note further that, since the SVM_{hybrid} model requires both the SVM_t^1 and SVM_{ta}^2 models to be trained, the training time for this model will be 507.6 seconds (the sum of the training times for the required models). And, the response time of the model will be 0.9 seconds for the issue reports without any screenshot attachments (as the SVM_t^1 model is used) and 2.17 seconds for the ones with the screenshot attachments (as the SVM_{ta}^2 model is used), on average.

Note that using the screenshot attachments for the assignments increased both the training and prediction times. Regarding the training times, as long as the total training times stay under few hours, such that the new models can be trained overnight, the increases in the training times are acceptable at Softtech. Note further that Softtech regularly maintains the backups for both the current and the previously used models, allowing a quick response

Table 7 Running times (in seconds) of various operations

metric	mean	std.	max	min	median
OCR time	2.11	1.05	8.51	0.57	1.85
Training time for SVM_t^1	190.4	6.69	202	180	190.5
Training time for SVM_{ta}^2	317.2	17.08	348	291	314.5
Response time for SVM_t^1	0.9	0.03	1.01	0.86	0.9
Response time for SVM_{ta}^2	2.17	0.09	2.54	2.07	2.16

mechanism when the current model is corrupted (due to corrupted files, for example) or when the performance of the current model substantially deviates from that of the previous model (which can temporarily be handled by replacing the current model with the previous one). Therefore, the increase in the training times from 190.4 seconds to 507.6 seconds is quite acceptable for Softtech.

Similarly, the increase in the prediction times was about 1.27 seconds (from 0.9 to 2.17 seconds) per assignment, on average. Note that once an IT-HD clerk (Section 2) enters an issue report to the system, the assignment is made in an asynchronous and fully automated manner in the background without requiring the IT-HD clerk to wait until the assignment is made. Therefore, the IT-HD clerks do not get affected by the increase in the prediction times. And, considering that the turnaround time for closing the issue reports at Softtech is about 2.61 days, on average (Aktas EU and Yilmaz C 2020a), the 1.27-second increase in the prediction times does not practically affect the AST team or the development teams either.

Therefore, the improvements in the assignment accuracies were obtained with little or no practical cost to Softtech. Indeed, Softtech was willing to accept much larger increases in both the training and the prediction times as long as the assignment accuracy was improved.

7 Threats to Validity

7.1 Construct Validity

To circumvent the construct threats, we used the well-known accuracy metric together with the other frequently used metrics, namely precision, recall, and F-measure (Murphy and Cubranic 2004; Anvik et al. 2006; Baysal et al. 2009; Anvik and Murphy 2011; Jeong et al. 2009; Bhattacharya et al. 2012; Jonsson et al. 2016; Dedík and Rossi 2016; Manning 2008). The discussions in the paper mainly focused on the accuracy results as this metric has been the choice of discussion in some of the recent related works (Aktas EU and Yilmaz C 2020a; Jonsson et al. 2016). After all, as also reported in the paper, the remaining metrics exhibited the same (or similar) trends with the accuracy metric.

Sometimes multiple teams may need to work together to resolve a reported issue. In such cases, more than one team could be treated as the correct assignment for the report. IssueTAG, on the other hand, assigns an issue report to a single team. Note, however, that treating multiple teams as correct assignees may increase (but not decrease) the assignment accuracies.

We, furthermore, measured the cost for different models in terms of the amount of time required to carry out the integral tasks regarding both the construction and the uses of the models. We did this because the running times of the proposed approaches were the most important concern at Softtech.

7.2 Internal Validity

To alleviate the threats to internal validity, we used mature tools to carry out the integral computations required by the proposed approaches. More specifically, we used py-tesseract (Smith 2007) for OCR; scikit-learn (Pedregosa et al. 2011) for tf-idf vectorization and linear SVM classification; Keras (Chollet and et al 2015) for word embeddings and CNN classification; and PyTorch deep learning framework (Paszke et al. 2019) for the multimodal model.

We have, furthermore, employed well-known and frequently-used pre-processing steps to analyze the text extracted from the issue reports and the screenshot attachments, including tokenization and removal of non-letter characters (Manning 2008). Similarly, the architectures of the machine learning models we used to extract and analyze the visual features present in the screenshot attachments, namely $VGG_{\hat{a}}^{1}$, $ResNet_{\hat{a}}^{1}$, $CNN\_VGG_{t\hat{a}}^{2}$ and $CNN\_ResNet_{t\hat{a}}^{2}$, were also published in the literature (Simonyan and Zisserman 2014; Joulin et al. 2017; Bojanowski et al. 2017; He et al. 2016).

We (unless otherwise stated) used the machine learning models with their default configurations. The performance of these models in the experiments might have been dependent on the underlying configurations. Note, however, that optimizing the configurations could have only improved the accuracy of the models.

Last but not least, we have checked the validity of the results manually by using manageable-size test sets. We also repeated the experiments by using different collections of training and test sets and observed the same trends.

7.3 External Validity

One external threat is that all of the issue reports used in the study were submitted to only one company, namely Softtech. However, Softtech is the largest software development company owned by domestic capital in Turkey. As Softtech produces and maintains dozens of business-critical systems comprised of hundreds of millions of lines of code, it shares many characteristics with the vendors of other business-critical systems, such as developing custom software systems; having a large, evolving codebase maintained by dozens of development teams; and receiving a large number of issue reports from the field, each of which generally needs to be addressed with utmost importance and urgency.

Another threat is that the development teams at Softtech typically use a small number of UI frameworks with a quite strict guidelines for designing the user interfaces. This makes the products produced by different teams to have the same/similar look and feel, which we believe was the main reason as to why the visual features extracted from the screenshot attachments were not helpful at all in the assignments. Therefore, we believe that, in scenarios where the look and feel of the products varies depending on the development teams, the visual features can still play an important role in the assignments.

A related threat is that we only use the valid issue reports in this work as the ground truth (i.e., the actual developments teams, to which the issue reports shall be assigned) for both training and evaluating the assignment models. However, since the identification of invalid issue reports is typically carried out before the assignments are made (Chaparro et al. 2017; Pandey et al. 2017; Herbold et al. 2020), the proposed approach can readily be used with the aforementioned approaches.

Furthermore, we have used only some of the well-known image extensions, including *.jpg*, *.png*, and *.tif*, to identify screenshot attachments. However, some other file formats, such as *.pdf*, may also need to be processed as they can contain screenshot attachments.

7.4 Conclusion Validity

All the issue reports used in this work were real issue reports submitted to Softtech. Furthermore, the period of time selected for the study was representative of the issue report database maintained by Softtech, in terms of the number of issue reports submitted, the percentage of the issue reports with screenshot attachments, and the number of development teams, to which these issue reports are assigned.

Furthermore, we used only the issue reports, which were marked as closed with the "resolved" status, in order not to introduce any bias in the assignment accuracies. At Softtech, the issue reports are closed by the development teams, who resolve the reported issues. Since the number of issue reports resolved by a team is used as a key performance indicator at Softtech, the developers pay utmost attention to correctly indicate the teams closing the issue reports. For a given issue report, we, therefore, used the development team, who closed the report, as the ground truth.

8 Related Work

Automated triaging of issue reports is still of practical interest (Jonsson et al. 2016; Lee et al. 2017; Zhang 2020; Aung et al. 2021). Therefore, a number of approaches have been proposed in the literature to automate the process of issue triaging (Ahsan SN et al. 2009; Alenezi et al. 2013; Anvik and Murphy 2011; Podgurski et al. 2003; Anvik et al. 2006; Baysal et al. 2009; Jeong et al. 2009; Bhattacharya et al. 2012; Lin et al. 2009; Helming et al. 2010; Park et al. 2011; Xia et al. 2013; Xie et al. 2012; Hu et al. 2014; Dedík and Rossi 2016; Jonsson et al. 2016; Lee et al. 2017; Chen et al. 2019a; 2019b; Gu et al. 2020; Zhang 2020; Sajedi-Badashian and Stroulia 2020; Aung et al. 2021; Chmielowski and Kucharzak 2021; Su et al. 2021).

Many of these approaches use only the one-line summaries and/or descriptions of the issue reports to assign them to the stakeholders for resolutions. There are, however, approaches that utilize different sources of additional information to further improve the assignment accuracy. Chen et al. utilize the discussions between the stakeholders for continuous incident triaging where the assignments are continuously refined until the correct assignments are made (Chen et al. 2019b). Su et al. determine the software components, with which the issue reports are concerned, by using an issue-tossing knowledge graph, which incorporates not only the information regarding the history of issue tossings, but also the information regarding the software components, such as their descriptions (Su et al. 2021). Hu et al. analyze the relationships between the software components and the issues associated with these components as well as the relationships between the components and the developers to recommend a human triager a list of suitable developers for each reported issue (Hu et al. 2014). Bettenburg et al. use the duplicated issues reports for the assignments (Bettenburg et al. 2008b). Anvik et al. offer human triagers a ranked list of recommended assignees, so that the triagers can select the assignee that they think fit best (Anvik et al. 2006). Bhattacharya et al. focus on utilizing the expertises of the developers, which are automatically inferred from the attributes of the software products/components that they modified to resolve the previously reported issue reports (Bhattacharya et al. 2012). Lin et al. and Jonsson et al. leverage the additional features extracted from the issue reports, such as the priority, the submitter, the affiliation of the submitter, the site from where the report was submitted, and the version of the product, for which the issue report was submitted (Lin et al. 2009; Jonsson et al. 2016). Helming et al. analyze the relationships between the issue reports resolved by the stakeholders and the respective functional requirements for making the assignments (Helming et al. 2010).

Our work differs from these existing works in that we use a different source of additional information, more specifically the screenshot attachments, for issue assignment. From this perspective, our approach and these existing approaches are complementary; they do not aim to replace one another.

Some recent works focus on using video recordings as issue reports (Cooper et al. 2021; Bernal-Cárdenas et al. 2020). More specifically, (Bernal-Cárdenas et al. 2020) use video recordings to automatically reproduce the reported issues and (Cooper et al. 2021) use them to identify duplicated reports. These approaches, however, mainly target mobile platforms and their applicability to the other computing platforms, especially for the stateful applications, the behaviors of which depend on the often persisted states (such as the ones stored in databases), is still an open question. For example, in the real issue reports submitted to Softtech, we did not have any screen recordings as attachments. Furthermore, the assignment problem, which is the main focus of this work, has not been addressed by the aforementioned works. Last but not least, our work uses screenshot attachments as an additional source of information for the assignments, not as the only source of information. However, using screen recordings in a similar manner for issue assignment is certainly an interesting avenue for future research.

While many of the existing works were evaluated on open source projects (Murphy and Cubranic 2004; Anvik et al. 2006; Baysal et al. 2009; Ahsan SN et al. 2009; Jeong et al. 2009; Anvik and Murphy 2011; Bhattacharya et al. 2012; Park et al. 2011; Alenezi et al. 2013; Xia et al. 2013; Xie et al. 2012; Sajedi-Badashian and Stroulia 2020; Aung et al. 2021; Su et al. 2021), few were evaluated on commercial, closed-source projects (Dedík and Rossi 2016; Helming et al. 2010; Jonsson et al. 2016; Lee et al. 2017; Lin et al. 2009; Chen et al. 2019a; 2019b; Gu et al. 2020; Zhang 2020; Chmielowski and Kucharzak 2021; Oliveira et al. 2021). Compared to the former set of works, we evaluate the proposed approach in a large industrial setup where hundreds of millions of lines of mostly business-critical codes were maintained by dozens of development teams.

Compared to the latter set of works, our system, IssueTAG, is actually deployed in the field, automatically making all the initial assignments since its deployment on $January$ 2018 (301,752 issue reports as of $November$ 2021). We have indeed been constantly maintaining IssueTAG by fine tuning it to further improve the assignment accuracy and by enhancing it with additional features, including the use of the other types of attachments; automatically breaking up the reports into parts, such as steps to reproduce, expected behavior, and observed behavior; and the knowledge of the recent changes in the codebase.

One difference we observe when it comes to the issue reports submitted to open source projects and the ones submitted to Softtech is that, the former tend to have more technical information, including the information regarding the internal workings of the systems. The latter, on the other hand, typically describe the symptoms of the failures as they are observed from outside the system by non-technical end users. Furthermore, the latter set of issue reports tend to be written more formally with little or no language errors at all, whereas the former set of issue reports tend to be written informally with grammar mistakes and typos.

Many of the existing works prefer to assign the issue reports to the individual developers (Ahsan SN et al. 2009; Alenezi et al. 2013; Anvik et al. 2006; Baysal et al. 2009; Bhattacharya et al. 2012; Jeong et al. 2009; Murphy and Cubranic 2004; Park et al. 2011; Xia et al. 2013; Xie et al. 2012; Hu et al. 2014; Sajedi-Badashian and Stroulia 2020; Aung et al. 2021; Su et al. 2021). In this work, however, we assign them to the development teams as with (Jonsson et al. 2016; Chmielowski and Kucharzak 2021). This, which was also the case before the deployment of IssueTAG, is indeed a decision deliberately made by Softtech to take the team dynamics into account during the assignments, which are quite difficult to model, such as the current workloads of the individual developers, the changes in the team structures, and the current status of the developers. After all many of the development teams at Softtech are close-knit teams following agile development processes. Note further that since the assignees are modeled as classes, there is no theoretical limit to the application of the proposed models for assigning the issue reports to the individual developers.

There are also other issue triaging-related tasks, including the identification of duplicated bug reports (Runeson et al. 2007; Bettenburg et al. 2008b; Cooper et al. 2021); determination of the severity levels for the reported issues (Menzies and Marcus 2008; Lamkanfi et al. 2010); estimation of the effort required for resolving the reported issues (Weiss et al. 2007; Giger et al. 2010; Zhang et al. 2013); separation of the issue reports indicating defects from the ones not indicating any defects (Antoniol et al. 2008); and the identification of the missing information in the issue reports (Bettenburg et al. 2008a). We believe that in all these tasks leveraging the screenshot attachments as an additional source of information can improve the performance of the proposed approaches.

9 Conclusion

In this work, we presented a number of approaches to use the screenshot attachments present in issue reports as an additional source of information for automated assignment. We, furthermore, evaluated all of the proposed approaches empirically by using a total of 84,972 real issue reports submitted to Softtech.

The results of the experiments strongly support our basic hypothesis that using screenshot attachments can further improve the assignment accuracy. We have arrived at this conclusion by noting that 1) a large fraction of all the issue reports submitted to Softtech has screenshot attachments; 2) in the presence of screenshot attachments, the one-line summary and the description fields of the issue reports often contain less information, compared to the issue reports without any attachments, which tend to reduce the assignment accuracy; 3) the screenshot attachments, on the other hand, convey invaluable information towards having better assignments; 4) for the issue reports with screenshot attachments, the assignment models, which use both sources of information (i.e., both the textual information present in the issue reports and the screenshot attachments) provided better accuracies compared to the models using a single source of information (i.e., either the textual information present in the issue reports or the screenshot attachments); and 5) all of these improvements were obtained at acceptable costs.

One potential avenue for future research is to further improve the assignment accuracy by focusing on the "important" regions in a given screenshot to filter out the parts, which create superficial commonalities between the issue reports. Another avenue is to use not only the screenshots, but also the other types of attachments, such as documents written in natural language and spreadsheets, for the assignments. Yet another avenue is to develop and evaluate transformer models with self-attention for automated issue triaging. Furthermore, we also plan to carry out user studies to qualitatively evaluate the practical effects of leveraging the attachments for the assignments. Last but not least, the attachments can also be leveraged in other types of bug triaging-related analyses, including the determination of the severity levels, identification of the duplicates, and the estimation of the efforts required for resolving the reported issues.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

Ahsan SN, Ferzund J, Wotawa F (2009) Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine. In: 2009 fourth international conference on software engineering advances. IEEE, pp 216–221

Aktas EU, Yilmaz C (2020a) Automated issue assignment: results and insights from an industrial case. Empirical Soft Eng 25(5):3544–3589

Aktas EU, Yilmaz C (2020b) An exploratory study on improving automated issue triage with attached screenshots. In: 42nd international conference on software engineering: companion proceedings. ACM/IEEE, pp 292–293

Aktas EU, Yeniterzi R, Yilmaz C (2020c) Turkish issue report classification in banking domain. In: 2020 28th signal processing and communications applications conference (SIU). IEEE, (pp 1-4)

Alenezi M, Magel K, Banitaan S (2013) Efficient bug triaging using text mining. JSW 8(9):2185–2190

Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc YG (2008) Is it a bug or an enhancement?: a text-based approach to classify change requests. In: CASCON, vol 8, pp 304-318

Anvik J, Murphy GC (2011) Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM Trans Soft Eng Method (TOSEM) 20(3):10

Anvik J, Hiew L, Murphy GC (2006) Who should fix this bug? In: Proceedings of the 28th international conference on Software engineering. ACM, pp 361–370

Aung TWW, Wan Y, Huo H, Sui Y (2021) Multi-triage: a multi-task learning framework for bug triage. J Syst Softw, pp 111133

Baysal O, Godfrey MW, Cohen R (2009) A bug you like: a framework for automated assignment of bugs. In: 2009 IEEE 17th international conference on program comprehension. IEEE, pp 297–298

Bernal-Cárdenas C, Cooper N, Moran K, Chaparro O, Marcus A, Poshyvanyk D (2020) Translating video recordings of mobile app usages into replayable scenarios. In: Proceedings of the ACM/IEEE 42nd international conference on software engineering, pp 309–321

Bettenburg N, Just S, Schröter A, Weiss C, Premraj R, Zimmermann T (2008a) What makes a good bug report? In: Proceedings of the 16th ACM SIGSOFT international symposium on foundations of software engineering, ACM, pp 308–318

Bettenburg N, Premraj R, Zimmermann T, Kim S (2008b) Duplicate bug reports considered harmful... really? In: 2008 IEEE international conference on software maintenance. IEEE, pp 337–345

Bhattacharya P, Neamtiu I, Shelton CR (2012) Automated, highly-accurate, bug assignment using machine learning and tossing graphs. J Syst Softw 85(10):2275–2292

Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146

Bouckaert RR, Frank E (2004) Evaluating the replicability of significance tests for comparing learning algorithms. In: Pacific-Asia conference on knowledge discovery and data mining. Springer, Berlin, (pp 3-12)

Chaparro O, Lu J, Zampetti F, Moreno L, Di Penta M, Marcus A, Bavota G, Ng V (2017) Detecting missing information in bug descriptions

Chen J, He X, Lin Q, Xu Y, Zhang H, Hao D, Gao F, Xu Z, Dang Y, Zhang D (2019a) An empirical investigation of incident triage for online service systems. In: 2019 IEEE/ACM 41st international conference on software engineering: software engineering in practice (ICSE-SEIP). IEEE, (pp 111-120)

Chen J, He X, Lin Q, Xu Y, Zhang H, Hao D, Gao F, Xu Z, Dang Y, Zhang D (2019b) Continuous incident triage for large-scale online service systems. In: 2019 34th IEEE/ACM international conference on automated software engineering (ASE). IEEE, (pp. 364-375)

Chmielowski L, Kucharzak M (2021) Impact of software bug report preprocessing and vectorization on bug assignment accuracy. In: Progress in image processing, pattern recognition and communication systems. Springer, (pp. 153-162), Cham

Chollet et al (2015) Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Cohen J (1988) Statistical power analysis for the behavioral sciences. Routledge

Cooper N, Bernal-Cárdenas C, Chaparro O, Moran K, Poshyvanyk D (2021) It takes two to tango: combining visual and textual information for detecting duplicate video-based bug reports. In: 2021 IEEE/ACM 43rd international conference on software engineering (ICSE). IEEE, (pp 957-969)

Dedík V, Rossi B (2016) Automated bug triaging in an industrial context. In: 2016 42th euromicro conference on software engineering and advanced applications (SEAA). IEEE, pp 363–367

Dietterich TG (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput 10(7):1895–1923

Everitt BS (1992) The analysis of contingency tables. CRC Press

Giger E, Pinzger M, Gall H (2010) Predicting the fix time of bugs. In: Proceedings of the 2nd international workshop on recommendation systems for software engineering. ACM, pp 52–56

Goodfellow I (2016) Deep learning. MIT press, Courville A

Gu J, Wen J, Wang Z, Zhao P, Luo C, Kang Y, Zhou Y, Yang L, Sun J, Xu Z, Qiao B, Li L, Lin Q, Zhang D (2020) Efficient customer incident triage via linking with system incidents. In: Inproceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp 1296–1307

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Helming J, Arndt H, Hodaie Z, Koegel M, Narayan N (2010) Automatic assignment of work items. In: International conference on evaluation of novel approaches to software engineering, Springer, pp 236–250

Herbold S, Trautsch A, Trautsch F (2020) On the feasibility of automated prediction of bug and non-bug issues. Empir Softw Eng 25(6):5333–5369

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Hu H, Zhang H, Xuan J, Sun W (2014) Effective bug triage based on historical bug-fix information. In: 2014 IEEE 25th International symposium on software reliability engineering, (pp 122-132). IEEE

Jeong G, Kim S, Zimmermann T (2009) Improving bug triage with bug tossing graphs. In: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM, pp 111–120

Joachims T (1998) Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th european conference on machine learning. Springer, ECML'98, pp 137–142

Jonsson L, Borg M, Broman D, Sandahl K, Eldh S, Runeson P (2016) Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. Empir Softw Eng 21(4):1533–1578

Joulin A, Grave E, Bojanowski P, Mikolov T (2017) Bag of tricks for efficient text classification. In: Proceedings of the 15th conference of the European chapter of the association for computational linguistics, vol 2, Short papers, Association for computational linguistics, pp 427–431

Lamkanfi A, Demeyer S, Giger E, Goethals B (2010) Predicting the severity of a reported bug. In: 2010 7th IEEE Working conference on mining software repositories (MSR 2010). IEEE, pp 1–10

Lee SR, Heo MJ, Lee CG, Kim M, Jeong G (2017) Applying deep learning based automatic bug triager to industrial projects. In: 2017 11th joint meeting on foundations of software engineering, pp 926–931

Lin Z, Shu F, Yang Y, Hu C, Wang Q (2009) An empirical study on bug assignment automation using chinese bug data. In: 2009 3rd international symposium on empirical software engineering and measurement. IEEE, pp 451–455

Mani S, Sankaran A, Aralikatte R (2019) Deeptriage: exploring the effectiveness of deep learning for bug triaging. In: Proceedings of the ACM India joint international conference on data science and management of data, pp 171–179

Manning CD (2008) Raghavan p. Introduction to information retrieval. Cambridge University Press, Schütze H

Menzies T, Marcus A (2008) Automated severity assessment of software defect reports. In: 2008 IEEE international conference on software maintenance. IEEE, pp 346–355

Murphy G, Cubranic D (2004) Automatic bug triage using text categorization. In: Proceedings of the sixteenth international conference on software engineering & knowledge engineering,Citeseer

Nadeau C, Bengio Y (1999) Inference for the generalization error. Adv Neural Inf Process Syst, vol 12

Oliveira P, Andrade R, Nogueira TP, Barreto I, Bueno LM (2021) Issue auto-assignment in software projects with machine learning techniques. arXiv:2104.01717

Pandey N, Sanyal DK, Hudait A, Sen A (2017) Automated classification of software issue reports using machine learning techniques: an empirical study. Innov Syst Softw Eng 13(4):279–297

Park JW, Lee MW, Kim J, Hwang Sw, Kim S (2011) Costriage: a cost-aware triage algorithm for bug reporting systems. In: Twenty-fifth AAAI conference on artificial intelligence

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G et al (2019) Pytorch: an imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 32:8026–8037

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: Machine learning in python. J Mach Learn Res 12(Oct):2825–2830

Podgurski A, Leon D, Francis P, Masri W, Minch M, Sun J, Wang B (2003) Automated support for classifying software failure reports. In: 25th international conference on software engineering, 2003. Proceedings. (pp 465-475). IEEE

Runeson P, Alexandersson M, Nyholm O (2007) Detection of duplicate defect reports using natural language processing. In: 29th international conference on software engineering, 2007. Proceedings, pp 499–510

Sajedi-Badashian A, Stroulia E (2020) Vocabulary and time based bug-assignment: a recommender system for open-source projects. Softw Pract Exp 50(8):1539–1564

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556

Smith R (2007) An overview of the Tesseract OCR engine. In: 2007 International conference on document analysis and recognition (ICDAR). IEEE, pp 629–633

Su Y, Xing Z, Peng X, Xia X, Wang C, Xu X, Zhu L (2021) Reducing bug Triaging confusion by learning from mistakes with a bug tossing knowledge graph. In: 2021 36th IEEE/ACM international conference on automated software engineering (ASE) (pp 191-202). IEEE

Truong C, Oudre L, Vayatis N (2018a) ruptures: change point detection in python. arXiv:180100826

Truong C, Oudre L, Vayatis N (2018b) Selective review of offline change point detection methods. arXiv:180100718

Weiss C, Premraj R, Zimmermann T, Zeller A (2007) How long will it take to fix this bug? In: Fourth international workshop on mining software repositories (MSR'07: ICSE workshops 2007). IEEE, pp 1–1

Weisstein EW (2004) Bonferroni correction. https://mathworld.wolfram.com/

Wilcoxon F (1992) Individual comparisons by ranking methods. In: Breakthroughs in statistics (pp 196–202). Springer, New York

Witten IH, Frank E, Hall MA (2005) Credibility: Evaluating what's been learned. Data mining: Practical machine learning tools and techniques, pp 143–186

Xia X, Lo D, Wang X, Zhou B (2013) Accurate developer recommendation for bug resolution. In: 2013 20th Working Conference on Reverse Engineering (WCRE). IEEE, pp 72–81

Xie X, Zhang W, Yang Y, Wang Q (2012) Dretom: Developer recommendation based on topic models for bug resolution. In: Proceedings of the 8th international conference on predictive models in software engineering. ACM, pp 19–28

Zhang H, Gong L, Versteeg S (2013) Predicting bug-fixing time: an empirical study of commercial software projects. In: Proceedings of the 2013 international conference on software engineering. IEEE Press, pp 1042–1051

Zhang W (2020) Efficient bug triage for industrial environments. In: 2020 IEEE International conference on software maintenance and evolution (ICSME). IEEE, (pp 727-735)

Ethem Utku Aktas received the BS degree in industrial engineering and MS degree in informatics from Middle East Technical University, Ankara, Turkey in 2000 and 2004, respectively. He received the PhD degree in computer science and engineering from Sabanci University, Istanbul, Turkey in 2021. Between 2000 and 2011, he worked at IsBank, Istanbul, Turkey as a software developer. Since then he has been working at Softtech Inc., Istanbul, Turkey as a senior software engineer. His current interests include software engineering and software quality assurance.

Cemal Yilmaz received the BS and MS degrees in computer engineering and information science from Bilkent University, Ankara, Turkey, in 1997 and 1999, respectively. In 2005, he received the PhD degree in computer science from the University of Maryland at College Park, MD, USA. Between 2005 and 2008, he worked as a post-doctoral researcher at IBM Thomas J. Watson Research Center, Hawthorne, New York. He is currently an associate professor of computer science in the Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey. His current research interests include software engineering, software security, and software quality assurance.