Assisting Software Projects with Bug Report Assignment Recommender Creation

John Anvik, Marshall Brooks, Henry Burton and Justin Canada
Department of Computer Science
Central Washington University
Ellensburg, WA 98926
{janvik,brookmar,burtonh,canadaj}@cwu.edu

Abstract—Software development projects receive many change requests each day and each report must be examined to decide how the request will be handled by the project. One decision that is frequently made is to which software developer to assign the change request. Efforts have been made toward semi-automating this decision, with the most promising approaches using machine learning algorithms. However, using machine learning to create an assignment recommender is a complex process that must be tailored to each individual software development project. This paper presents the Creation Assistant for Easy Assignment (CASEA), a tool that leverages a project member’s knowledge to assist in creating an assignment recommender specific to the software development project.

Keywords—bug report triage; assignment recommendation; machine learning; recommender creation

I. INTRODUCTION

Large software development projects can receive hundreds of bug reports per day [1]. Each of these bug reports needs to be analyzed to determine if the report will result in development activity. For example, a report may be examined in order to determine whether the report is a duplicate or if the problem is reproducible. In cases where the problem needs action from a developer, a decision needs to be made about to whom the work will be assigned. This decision process is called bug triage and must be done for all incoming reports.

Bug triage takes significant time and resources [2]. Bug report assignment recommenders have been proposed as a method for reducing this overhead. Conceptually, the creation of an assignment recommender using machine learning is straightforward. Six questions need to be answered to create an assignment recommender [1]. These questions relate to which bug reports to use and how many, determining who fixed a report, which developers to recommend, what data from the bug reports to use, and which algorithm to use.

In practice, the creation of such recommenders for a specific software development project is challenging. If a project member were to begin creating an assignment recommender for their project s/he would first have to research machine learning libraries and algorithms, as well as methods for obtaining bug reports from their repository, and that’s just for a basic recommender. Also, finding answers to these questions requires significant experimentation to construct an optimal recommender due to interdependencies between questions. For example, how the reports are labelled with developer names affects the set of names recommended, and the choice of valid names affects which reports and how many of them will be used for creating the recommender. Finally, further research may need to be done to find ways to better format the data such as word stemming, parts of speech analysis and feature reduction. Creating an assignment recommender could take many hours to days or weeks for someone who is unfamiliar with the process.

Some answers for assignment recommender creation appear to be consistent across projects (e.g. the text from the bug report summary and description is used as the data [1] [3] [4]). Other questions are specific to the project. For example, how to determine which developer fixed a bug requires the use of heuristics, and these heuristics differ from project to project [1]. The heuristics that are used for labeling reports from one project, such the Eclipse Platform project1, are not the same as the heuristics used for another project, such as the Firefox project2. Creating these heuristics is a time consuming process. It was observed to take between half a day to a full day depending on the complexity of the heuristics [1].

Establishing these heuristics once may not be sufficient. The recommender may need to be reconfigured if any significant changes are made in a project’s use of the issue tracking system. For example, a project may add or deprecate bug lifecycle states over time. Such changes would require a reconfiguration of the project-specific labeling heuristics and the set of reports used for creating the assignment recommender.

The initial configuration and ongoing adjustments for creating an assignment recommender represents a cost to the project. For example, the Mozilla project at one time had four different web browser projects. There existed subsets of developers that either worked exclusively on one of these projects, or overlapped with one or more these projects. If the Mozilla project had decided to incorporate an assignment recommender into its development process at that time, then a different assignment recommender would have had to be created for each subproject. At the limit, the Mozilla project, which currently has thirty-six subprojects, would have to create and maintain thirty-six different assignment recommender configurations. This could pose a substantial cost to the project.

1 http://www.eclipse.org/jdt
2 http://www.mozilla.org/firefox
If the use of assignment recommenders is to be made practical, the costs associated with creating and maintaining recommender configurations needs to be reduced. As reducing the number of different configurations that will need to be created and maintained by a project is unlikely, seeking to lower the costs associated with each individual configuration offers a viable solution. Also, the fact that the techniques for creating an assignment recommender extend to other types of triage recommenders [1] makes reducing such costs all the more important.

This paper presents the Creation A$tistant for Easy A$signment (CASEA), an implementation of the approach proposed in [1] that leverages a project member’s knowledge to assist in creating and managing bug report assignment recommender configurations. Specifically, CASEA assists in the creation of project-specific heuristics and the selection of valid developer recommendations. CASEA also allows project members to experiment with these two parameters to quickly determine if the use of an assignment recommender would benefit a project. Finally, CASEA helps project members to ensure that the assignment recommender is trained with the best data possible by considering the actual data being used to train the recommender.

This paper proceeds as follows. First, background information about bug reports and machine learning is presented. Next, details of how CASEA uses a project member’s knowledge to create an assignment recommender are given. The paper then concludes with related work.

II. BACKGROUND

This section presents background information about bug reports, their life cycles, and machine learning.

A. Bug Reports

Bug reports, also known as change requests, provide a means for users to communicate software faults or feature requests to software developers. They also provide a means for developers to manage software development tasks.

Bug reports contain a variety of information, some of which is categorical and some of which is descriptive. Categorical information includes such items as the report’s identification number (i.e. bug id), its resolution status (e.g., NEW or RESOLVED), the component the report is believed to involve, and which developer has been assigned the work. Descriptive information includes the title of the report, the description of the report, and discussions about possible approaches to resolving the report. Finally, a report may contain other information, such as attachments or links to other reports.

B. Bug report lifecycles

All bug reports have a lifecycle, although the specific states can vary between projects. When a bug report first enters a project’s issue tracking system (ITS), it is in a state such as UNCONFIRMED or NEW. The bug report then moves through different states, depending on the project’s development process, and arrives at a resolution state, such as FIXED or INVALID. The lifecycle of a bug report can be used to categorize bug reports [1].

C. Machine learning algorithms

Machine learning is the development of algorithms and techniques that allow computers to learn [5]. Machine learning algorithms fall under three categories: supervised learning, unsupervised learning, and reinforcement learning. Bug report assignment recommenders primarily use supervised learning algorithms, such as Support Vector Machines (SVM) and Naive Bayes.

Understanding how a machine learning algorithm creates a recommender requires understanding three concepts: the feature, the instance and the class. A feature is a specific piece of information that is used to determine the class, such as a term that appears in one or more of a set of bug reports. An instance is a collection of features that have specific values, such as all of the terms in the description of a specific bug report. Finally, a class is the collection of instances that all belong to the same category, such as all of the bug reports fixed by a developer. In supervised machine learning, training instances are labeled with their class. A recommender is created from a set of instances and the output of the recommender is a subset of the classes to which the instances belong.

III. ASSISTING ASSIGNMENT RECOMMENDER CREATION

This section presents CASEA, a software tool to assist a software project in creating and maintaining bug assignment triage recommenders. Although CASEA only supports the creation of assignment recommenders, it can be extended to assist with the creation and maintenance of other triage recommenders. Through the use of CASEA, a user would be able to use their project knowledge to create an assignment recommender in a short period of time.

CASEA guides a project member through the assignment recommender creation process in four steps: Data Collection, Data Formatting, Recommender Training, and Evaluation. The remainder of this section presents the details of how CASEA assists with each of these steps.

1) Data Collection

The first step in recommender creation is to gather the data to be used for creating the recommender. Specifically, bug reports are extracted from the project’s issue tracking system. CASEA provides two methods for data collection.

The first option is to import two XML files containing the bug report data: one file containing the data for the reports to be used for training, and the other containing data to be used for evaluation. On importing the file, CASEA’s Document Object Model (DOM) is populated.

The other option is for the project member to provide the URL of the project’s ITS. An XML-RPC library is used to pull the data from the ITS and populate the DOM model. The user specifies the initial bug report creation date from which to start pulling reports, and the number of reports to gather. Reports that have a resolution status of RESOLVED, VERIFIED or CLOSED are chronologically gathered from the starting date. Every tenth report collected is designated as a testing report to create an unbiased set for evaluation.
2) Data Formatting

Having collected the data from the project’s ITS, the next step is to format the data into a collection of instances to be used for training the recommender and evaluating its performance. However, before the instances are created, the data is first filtered to produce the highest quality training set. Two types of filtering are performed: automatic and assisted.

a) Automatic Filtering

Recall that an instance contains a label and a set of features. For bug report assignment the features are the terms that occur in the summary and description of the bug report. The automatic filtering performs three actions on the textual data.

First, terms that are stop words are removed. Stop words are commonly occurring words that provide little to no information content. Examples of stop words include ‘a’, ‘the’, and ‘an’. After stop words are removed, the next step is stemming. Stemming reduces all of the terms to their respective root values. Stemming is done so that words such as ‘user’ and ‘users’ are treated as the same word, ensuring a common vocabulary between the reports. Finally, punctuation and numeric values are removed, except where the punctuation is important to the term, such as class names (e.g. “org.eclipse.jdt”) or URLs.

b) Assisted Filtering

CASEA assists the user with two types of filtering: label filtering and instance filtering. In the case of assignment recommender creation, label filtering is the selection of developer names to be recommended and instance filtering is the selection of the bug reports to be used for recommender training and evaluation.

To assist with label filtering, CASEA presents the user with a label frequency graph. For an assignment recommender, this graph presents bug fixing statistics for the project developers based on all the reports in the training data set. For an assignment recommender, one would want to recommend from a core set of developers, not from the set of all developers, as a developer who has only solved one report in the past is unlikely to solve another. CASEA allows the user to select a threshold using a slider, such that only a core set of developers are recommended. By default, CASEA sets the threshold to five resolved reports. As the user moves the slider, the horizontal red cutoff line is adjusted.

Instance filtering is done using project-specific heuristics. The heuristics have two parts: a grouping rule and a label source. The grouping rule is used to categorize the data into groups for which the label source will be used for labeling the instances. For an assignment recommender, the grouping rule is a bug report lifecycle and the label source is either a field from the bug report, such as the assigned-to field, or other labelling information that can be extracted from the bug report, such as the user that attached a patch or marked the report as resolved. The default label source is the assigned-to field.

The lifecycle states occurring between the initial state and the resolution state may affect which field(s) can be used for labelling. For example, bug reports with the lifecycle NEW → FIXED would never be formally assigned to a project developer and were probably resolved by the person who created the report. Therefore the reporter field is likely to be the correct label source for this group of bug reports. For bug reports that follow the lifecycle NEW → ASSIGNED → FIXED, the assigned_to field is likely the best labeling choice given that the report was triaged and an appropriate developer was selected.

All of the reports in the training data set are used to determine the specific bug report life cycles for the project and the user is presented with a statistical summary of the categories. Using this information, the user can create heuristics for the most common occurring categories. For each of these categories, the user then selects the label source from a drop down, including a choice to not label reports in the group.

The user can also choose the number of heuristics to be applied, to a maximum of ten. An observation from using CASEA on a number of projects has shown that approximately 70% of reports are covered by the top five bug report lifecycles. To label the reports not covered by a configured heuristic, the user selects a default label source.

As was mentioned, dependencies exist between choices made in configuring an assignment recommender. In CASEA, there is a dependency between the two types of assisted filtering. The values in the label frequency graph come from the application of the labelling heuristics to the data, and changes to the set of heuristics cause the label filtering graph to be updated. In other words, the developer activity graph reflects the set of names from the use of the current heuristics.

c) Formatting

After filtering the data to create the set of training and evaluation instances for the recommender, the data is then formatted for use with the machine learning algorithm. Specifically, two tables are created. First, a term frequency table is created where each column is a feature from the instances and the rows are instance ids. For an assignment recommender, the rows are bug report ids and the columns are terms. The content of the table is the term-frequency/inter-document frequency (tf-idf) values for each term in each report of the data set. Tf-idf provides a measure of the importance of a term within a document. This table is only created for the training instances.

The second table is an index of instances to labels. In this case, the table is an index of bug report ids to developer name given by the heuristics. This table is created for both the training and evaluation data sets.

3) Recommender Training

Once the user has filtered the data and the data is formatted, the recommender is created using a machine learning algorithm. The two most commonly used machine learning algorithms for assignment recommenders are Support Vector Machines and Naïve Bayes [1] [4]. CASEA uses a multi-class SVM algorithm with a Gaussian kernel for creating a recommender. The time to train a recommender was found to be between a few seconds and a few minutes, depending on the size of the training dataset, the configuration parameters, and the computer used.

472
4) Evaluation

Once the user starts the recommender creation process, the user is moved to an Analysis tab that presents progress information, such as the time taken to train the recommender and evaluation results. The user can then return to the Configuration tab, adjust the values for label and instance filtering, and create a new recommender. This process continues until the user is either satisfied with the created recommender, or the user has determined that an assignment recommender cannot be created with a high enough accuracy to benefit the project. At any time the user can save the recommender and return at a later date.

To evaluate the performance of the recommender, the measures of precision and recall are used. Precision and recall are defined as follows:

\[
\text{Precision} = \frac{\text{# correct recommendations}}{\text{# recommendations}}
\]

\[
\text{Recall} = \frac{\text{# correct recommendations}}{\text{# with correct expertise}}
\]

To accurately compute these values, the set of developers that had the expertise to resolve each bug report is needed. However, this information can be difficult to determine and approximations are necessary [1]. CASEA uses a list of developers that resolved reports in the same component as an approximation of bug fixing expertise. Note that this approximation will overestimate developer expertise, trading off a more accurate precision value for a less accurate recall value. As high precision is favored over high recall this tradeoff is acceptable. CASEA presents results for the top one, three and five ranked recommendations.

IV. RELATED WORK

A. Developer Assignment Recommendation

Many prior researchers have proposed using assignment recommenders to reduce the amount of resources to be put towards triage, such as [1] [3] [4] [6] [7]. All previous work has sought to find specific answers to the assignment recommender creation questions; however none of these works, except for [1], have addressed the problem of making the use of such recommenders practical.

B. Assisting with Recommender Creation

Making the use of machine learning recommenders practical has been done in other areas, such as bioinformatics. However, these tools typically focus on explaining the results of the recommendations [8] or helping to visualize how machine learning works [9].

There are two commercial products that could be considered similar to CASEA. Skytree Server [10] is similar in intent to CASEA; however to use Skytree Server still requires advanced knowledge of machine learning and statistics [11]. Big ML [12] also provides assistance with using machine learning, but is restricted to a single machine learning algorithm shown to be not effective for assignment recommendation [1].

V. CONCLUSION

This paper presented CASEA, a tool to assist in the creation of bug report assignment recommenders. CASEA assists a user in labelling and filtering the bug reports used for creating a project-specific assignment recommender. Although there have been many efforts towards determining a good approach to assignment recommender creation, these approaches do not address how to leverage a project member’s knowledge in the creation of the recommender. CASEA seeks to remedy this situation.

REFERENCES


