Can Automated Text Classification Improve Content Analysis of Software Project Data?

John Noll, Dominik Seichter and Sarah Beecham
Lero, the Irish Software Engineering Research Centre
University of Limerick, Ireland

Abstract—Content analysis is a useful approach for analyzing unstructured software project data, but it is labor-intensive and slow. Can automated text classification (using supervised machine learning) be used to reduce the labor or improve the speed of content analysis?

We conducted a case study involving data from a previous study that employed content analysis of an open source software project. We used a human-coded data set with 3256 samples to create different size training sets ranging in size from 100 to 3000 samples to train an “ensemble” text classifier to assign one of five different categories to a test set of samples.

The results show that the automated classifier could be trained to recognize categories, but much less accurately than the human classifiers. In particular, both precision and recall for low-frequency categories was very low (less than 20%). Nevertheless, we hypothesize that automated classifiers could be used to filter a sample to identify common categories before human researchers examine the remainder for more difficult categories.

Index Terms—Qualitative Research; Content Analysis; Text Classification; Machine Learning; Software Engineering; Open Source Software.

I. INTRODUCTION

Software project repositories contain vast amounts of textual data that comprise important information about the project, such as requirements, design rationale, process and management policies and procedures, and community norms. This information is thus a rich source of empirical data about how real software projects work (and fail).

Analyzing large textual data sets using conventional content analysis requires human researchers to manually read and classify each text fragment, then reach agreement on both the classifications and meanings of fragments. This approach can yield valuable insights into the workings of a project [1], but is labor-intensive and time-consuming. This study, therefore, attempts to answer the following research question:

Can automated text classification be used to improve the speed or reduce the labor involved in content analysis of software engineering data?

To answer this question, a sample of 3,256 postings to the online forums of an open source software project was coded by the authors using conventional content analysis. An “ensemble” classifier was then trained using training set sizes ranging from 250 to 3000 coded postings selected from the coded sample; the ensemble was then applied to a test sample, and Cohen’s kappa statistic was calculated as a measure of classification accuracy for the ensemble of classifiers when compared to the codes assigned by the human researchers.

The results show that automated classification is unlikely to be useful as a replacement for conventional manual content analysis. However, ensemble agreement might be useful for reducing the size of a sample by filtering out the easy to classify samples, leaving human researchers to focus on the difficult fragments.

II. BACKGROUND

Automated text classification is applied for two distinct reasons: first, to automatically extract structured (or semi-structured) data, and second, to automatically classify or categorize the data according to pre-defined classes. This study deals with the second method, text classification, whereby text classifiers learn models for a given set of categories or classes, and apply these models to new unseen topics or class assignments [2]. Automated text classification is found to have important uses in the real world; for example it has been used to categorize news articles into topics, categorize and route email, build and maintain web directories, and create spam filters [2].

Machine learning is an inductive process that builds an automatic text classifier by learning the characteristics of the categories of interest from a set of pre-classified documents [3]. Software engineering researchers have been exploring how machine learning techniques might support software development activities; for example studies have been conducted in requirements prioritization where machine learning was used to induce requirements ranking approximations from the acquired data [4]. Other uses of machine learning in software engineering include support for requirements elicitation, traceability and transformation [5]; software development effort estimation [6], and software testing and fault prediction [7, 8].

Although automated text classifiers can be effective in domains where the underlying textual data follow a formal structure, many other less structured domains (such as web sites) report accuracy levels that are far from satisfactory; text classifiers have trouble with data that are varied, non-linear and do not follow set verbal communication rules [9].

III. METHOD

To assess the effectiveness of automated text classification for analyzing software project data, we used ensemble classifi-
TABLE I
FORUM POST CODES AND MEANINGS.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>impl</td>
<td>A post announcing code implementing new functionality, or enhancing existing functionality.</td>
</tr>
<tr>
<td>fix</td>
<td>An announcement of code to fix a bug, or a bug fix patch submitted to the bug fix or code review tracker.</td>
</tr>
<tr>
<td>prop</td>
<td>A description of a proposed enhancement or new feature, or request for enhancement or new functionality.</td>
</tr>
<tr>
<td>issue</td>
<td>A bug report, or request for help with some issue involving configuring or running the product.</td>
</tr>
<tr>
<td>other</td>
<td>None of the above.</td>
</tr>
</tbody>
</table>

cation [10], which is a machine learning approach that applies a set or “ensemble” of classification algorithms to each data sample. The algorithms “vote” on a category for the sample, and the category receiving the majority of votes is assigned to the fragment.

The specific steps employed in this study are as follows:

1. Create sample set of labelled text. The authors coded a set of 3,256 randomly selected messages posted to the OpenEMR [11] open source electronic medical record project discussion forums. These posts address a range of topics, from issues with using the OpenEMR software to proposals for new features.

The messages were divided equally among the authors, who then assigned one of five labels (see Table I) to each message, using a validated coding scheme developed for an earlier study[1]. This coding scheme was refined over a series of five iterations involving two researchers coding the same trial samples, then comparing agreement using Cohen’s kappa statistic [12]. After each iteration, disagreements were examined and the coding scheme and checklist were refined to resolve ambiguity.

2. Create input data sets. To see if the author and subject fields might help automated classifiers arrive at a correct label, we created three input data sets: post body only, with just the body of the post; post body and author, created by concatenating the author and body fields of a post; and, thread subject, post body, and post author, comprising samples created by concatenating the thread subject, the post author, and post body fields.

3. Create training sets. A collection of twelve training sets of increasing size (50, 100, 150, 250, 500, 750, 1000, 1500, 2000, 2500, and 3000 postings) was created by randomly selecting from the three input data sets after the test sets were removed. In total, this yielded 36 training sets.

4. Create test set. After the training set postings were removed from the 3,256 samples, test sets of 256, 512, or 1024 posts were randomly selected from the remaining samples, to create test sets to use for testing the ensemble classification.

5. Classify test set using ensemble classification. Each ensemble was applied to the test set corresponding to the fields included in the training set. This yielded twelve trials each of the body-only test set, the author-body test set, and the subject-author-body test set.

6. Measure performance. We computed precision (fraction of machine-coded samples that are correctly coded), recall (fraction of samples with a given human-assigned code that were identified by the machine), and Cohen’s kappa statistic [12], which measures agreement between two classifiers, taking into account the fact that some agreement would occur even when both classifiers assigned labels randomly.

IV. RESULTS

The results of this study are mixed. On the one hand, the ensemble classifiers were able to approach “moderate” agreement with the human coders on Landis and Koch’s kappa benchmark scale [21], on training sets as small as 500 samples (see Fig. 1). Further, precision for the issue, prop, and other codes was above 60%, and recall for the other code exceeded 60%. However, the classifiers failed to recognize any from the samples, converting text to lowercase, then “stemming” the remaining words (for example, reducing words such as “swimming” to “swim”) [13] to create a document-term matrix that shows which terms appear in which posts [14]. The ensemble included the following algorithms implemented in the RTextTools library: Boosting [15], Generalized Linear Models [16], Maximum Entropy [17], Random Forest [18], Support Vector Machine [19], and Regression Tree [20]. The result was a set of 36 ensemble classifiers, comprising twelve classifiers trained on each of the three training sets extracted from each the three input samples.

7. Measure performance. We computed precision (fraction of machine-coded samples that are correctly coded), recall (fraction of samples with a given human-assigned code that were identified by the machine), and Cohen’s kappa statistic [12], which measures agreement between two classifiers, taking into account the fact that some agreement would occur even when both classifiers assigned labels randomly.

IV. RESULTS

The results of this study are mixed. On the one hand, the ensemble classifiers were able to approach “moderate” agreement with the human coders on Landis and Koch’s kappa benchmark scale [21], on training sets as small as 500 samples (see Fig. 1). Further, precision for the issue, prop, and other codes was above 60%, and recall for the other code exceeded 60%. However, the classifiers failed to recognize any
fix posts, and recall for fix and prop posts was below 10% for most classifiers.

The precision results shown in Fig. 2 are not so surprising in light of the skewed nature of the data: over half of the samples have type other (Fig. 4), so we would expect the classifiers to be best at recognizing this code. Conversely, at most 5% of samples have code impl, or fix, so, we would expect that the classifiers would be best at recognizing other, issue, and prop.

Human coders have an advantage over machines when classifying this kind of data: a human can read an entire discussion thread and therefore use the context to interpret the meaning of a post. For example, the first message posted to a thread in the “users” forum is likely to be of type issue, especially if the title is a question. Also, fix type posts often appear at the end of a long discussion thread. Finally, posts to the “patch tracker” forum are usually of type impl.

V. CONCLUSIONS

In this study, we hoped to answer the question, Can automated text classification be used to improve the speed or reduce the labor involved in content analysis of software engineering data? The results were somewhat mixed. And limitations of the study, including the skewed nature of the sample data Fig. 4, make the results difficult to generalize: it is possible that the same approach applied to a different data set might yield far better results. Nevertheless, a study of this kind can be useful in generating hypotheses for further investigation. As such, we propose three hypotheses derived from our results and experiences conducting this study:

Hypothesis 1: Machines will not replace human researchers in the near future for classifying textual software project data.

This hypothesis is based on our experience with creating the training sets, tuning the input data, and marshaling the necessary computation resources: just the training of the ensemble classifiers required nearly 24 hours of CPU time on a 12-core compute node with 24 Gigabytes of RAM. By contrast, a human researcher can code up to two posts per minute, meaning three researchers can generate a significant data set in a few days. If this hypothesis holds, the answer to our research question would be “no.”

Hypothesis 2: Performance of automated classifiers when classifying discussion forum posts will be improved by providing them with context about the post.

For example, our experience shows that whether a post is the first in a thread, and the forum in which the post appears is the user forum, the post is likely to be an issue type post. If this hypothesis holds, the answer to our research question could be “yes,” depending on the amount of improvement.

Hypothesis 3: A hybrid approach combining automated text classifiers with human classification will be highly efficient.

This hypothesis is based on the observation that “fair” agreement is achieved by classifiers trained on as few as 500 samples, which is about the same number that we required to develop the coding scheme and checklist; as such, we essentially created a training set at no cost, as a side effect of developing the coding scheme. Thus, it seems appropriate to use a machine trained on a small training set to filter the data to identify the most common posts (type other in this case); then human researchers can concentrate on the remaining posts, which machines find more difficult to classify. If this hypothesis holds, the answer to our research question would be “yes.”
Fig. 4. Distribution of codes in sample data.

ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie). The SFI/HEA Irish Centre for High-End Computing (ICHEC) provided computational facilities and support.

REFERENCES


