On the Effectiveness of Labeled Latent Dirichlet Allocation in Automatic Bug-Report Categorization

Minhaz F. Zibran
University of New Orleans
2000 Lakeshore Drive, New Orleans, LA, USA
zibran@cs.uno.edu

ABSTRACT

Bug-reports are valuable sources of information. However, study of the bug-reports' content written in natural language demands tedious human efforts for manual interpretation. This difficulty limits the scale of empirical studies, which rely on interpretation and categorization of bug-reports. In this work, we investigate the effectiveness of Labeled Latent Dirichlet Allocation (LLDA) in automatic classification of bug-reports into a predefined set of categories.

CCS Concepts

• Software and its engineering → Software libraries and repositories;

Keywords

bug-report; automatic categorization; topic modelling; LLDA

1. INTRODUCTION AND MOTIVATION

Bug-repositories play a vital role in managing software defects over long term. Especially the long-lived software projects benefit from maintaining an accompanying bug-repository in activities such as bug triage, assignment of bug-fixing tasks, and keeping track of a history of known bugs. For purposes such as bug triage and bug-fixing task assignment, bug-reports in a bug-repository are typically kept characterized in several ways in terms of severity, priority, affected module, and open-close status of the bug-reports.

While these characterization of the bug-reports help in regular bug triage, additional means of characterizations are necessary for carrying out further analyses, given that bug-repositories are valuable sources of information used in software engineering research and development. For example, Zibran et al [19] studied bug-posts to determine the significances 22 factors, which affect the usability of published APIs (Application Programming Interfaces). They manually studied more than a thousand bug-posts, distinguished those bug-reports relevant to API usability issues, and characterized each of them based on what usability issues are reflected in the bug-posts. Due to unavailability of a reference-corpus, which could be used to train a NLP (Natural Language Processing) technique and automate the categorization process, they had to rely on their subjective analyses.

Indeed, manual investigation of thousands of bug-posts is a very tedious task. Devising a technique for automatic categorization of bug-reports (and documents written in natural languages) can save a lot of efforts and also allow large scale studies based on bug-report characterization. This work follows the study of Zibran et al [19]. In this paper, we present our ongoing work towards automatic bug-report categorization for large scale investigation of the significances of API usability factors reported earlier [18, 19]. We apply Labeled Latent Dirichlet Allocation (LLDA) [8], which is a probabilistic topic modelling technique emerged from the field of NLP. In particular, our work aims to make two contributions:

• We investigate the prospect and effectiveness of LLDA in automatically classifying bug-reports into a fixed set of predefined categories.
• We produce a curated data-set of bug-reports, which will benefit the research community in NLP tool evaluation and can be used as a training oracle in large scale studies, which rely on categorization bug-reports.

2. APPROACH

The set of 22 categories of API usability factors reported earlier [18, 19] is presented in Table 1. Further details about these 22 categories can be found elsewhere [18, 19]. In this work, we investigate the feasibility of LLDA in categorizing bug-reports into those closed set of 22 categories. Our study includes 1,138 bug-reports for three different projects namely Eclipse, GNOME, and Python 3.1 (Table 2).

<table>
<thead>
<tr>
<th>Table 1: Set of 22 categories of interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
</tr>
<tr>
<td>Naming</td>
</tr>
<tr>
<td>Caller's perspective</td>
</tr>
<tr>
<td>Documentation</td>
</tr>
<tr>
<td>Consistency</td>
</tr>
<tr>
<td>Error handling</td>
</tr>
<tr>
<td>Reference chain</td>
</tr>
<tr>
<td>API aging</td>
</tr>
<tr>
<td>API change</td>
</tr>
<tr>
<td>Data types</td>
</tr>
<tr>
<td>Use of attributes</td>
</tr>
</tbody>
</table>

2.1 Phase 1: Oracle Development

The unavailability of a reference-corpus is a severe difficulty in applying LLDA for classifying bug-reports into those
22 categories. Hence, we carefully go through each of these bug-reports and distinguish 428 of them as relevant to API usability. Then we further investigate each of these API-related 428 bug-reports. A bug-report is labeled with one or more of those 22 categories depending on whether the respective usability issues are found in the content and discussion of the bug-post. Thus, we develop an oracle of 428 bug-reports with known categorization.

### 2.2 Phase-2: Applying NLP

For applying LLDA, we use the Stanford Topic Modeling Toolbox (TMT) version 0.3.3 [1], which is a framework with core NLP algorithms including LLDA. We write Scala scripts making calls to appropriate APIs as required. Given a training-corpus of documents associated with multiple labels, LLDA is reported to be able to categorize documents by inferring the labels appropriate for each document in a separate inference-corpus [8]. Using the oracle developed in phase-1, we measure accuracy of the categorization in terms of the precision, recall, and F-score.

#### 2.2.1 Document Preparation and Cleanup

Before applying LLDA, each of the bug-reports is transformed into a document containing only the title and concatenated comments from the bug-report. The documents are further cleaned up with the help of TMT at the time of applying LLDA. The stop-words, numbers, punctuation characters, and words having length of one or two characters are filtered out.

#### 2.2.2 Invoking LLDA

A proper application of LLDA involves two stages: learning and inference [8]. At the learning stage, LLDA is applied on a training corpus (with documents having known labels) to develop a model, which is then used at the inference stage to infer labels of documents in inference-corpus. We split the oracle of documents corresponding to the 428 bug-reports (distinguished in phase-1) into two disjoint sets. The 266 documents corresponding to the GNOME bug-reports constitute the larger corpus, and the remaining 162 documents corresponding to the Eclipse and Python 3.1 bug-reports make the smaller corpus.

First, we train LLDA on the smaller and invoke it for inference on the larger corpus. Next, we use the larger corpus for training, and smaller one for inference. Then, we compare the results.

### 3. FINDINGS

Table 3 presents the accuracy of automated bug-report categorization when LLDA is trained on the smaller corpus and when the larger corpus is used for training. As we see, in Table 3, the accuracy in terms of precision, recall, and F-score substantially increased when the LLDA is trained on the larger corpus. The magnitude of increase in accuracy indicates that LLDA is potentially effective in automatic bug-report categorization if a sufficiently large corpus is used at the training stage.

### 4. THREATS TO VALIDITY

Our study may be subject to human errors in the interpretation of the bug-reports to produce the oracle. To minimize this threat, randomly picked 120 of the bug-report were verified for sanity check by two computer science practitioners other than the author. Exhaustive manual cross-check over all the bug-posts was not performed to minimize cost (time and effort) and that the random sanity check indeed confirmed correctness.

### 5. RELATED WORK

A number of earlier work [3, 7] attempted automatic binary classification to distinguish enhancement/feature requests from report addressing software defects. Bug-reports are also studied for detection of duplicate bug-reports [10, 12, 13], to facilitate bug triage [15], for predicting severity [5, 6], priority [14] of bug-reports, and estimating the time needed to fix a bug [2, 4, 16]. Attempts are also made for automatic summarization [9] of bug-reports and to develop hybrid techniques for bug-report classification [11, 17].

Although all of these work are more or less relevant to our work, ours significantly differs from them in terms of the objectives and approaches. For example, Somasundaram et al. [11] applied unsupervised LDA for automatic categorization of bug-reports, while we use supervised labeled LDA, which is an improved variant of LDA, particularly suitable in documents classification where a document can be classified to more than one categories in a closed set of categories [8], as such required for our work. Moreover, our work deals with an additional difficulty of not having an existing reference-corpus and we build one for our purpose.

### 6. CONCLUSION

In this work, we have investigated the prospect and effectiveness of LLDA-based topic modelling in automatic bug-report categorization. Based on a study on 1,138 bug-posts from bug-repositories for three different software projects, we derived that topic modelling can indeed be promising for such purposes when trained on large training corpus.

We, therefore, will extend this work by enlarging the oracle, which can be reused as a curated reference-corpus for tool evaluation and for conducting similar studies in larger scale. We also plan to carry out large scale studies to verify the results reported in earlier studies [18, 19], which relied on bug-report interpretation and categorization.

Acknowledgement: Thanks to Farjana Z. Eishita and Chanchal K. Roy for helping with random sanity check in the development of the oracle.
7. REFERENCES