Towards an Improvement of Bug Report Summarization Using Two-Layer Semantic Information

Cheng-Zen YANG†, Member, Cheng-Min AO†, and Yu-Han CHUNG†, Nonmembers

SUMMARY Bug report summarization has been explored in past research to help developers comprehend important information for bug resolution process. As text mining technology advances, many summarization approaches have been proposed to provide substantial summaries on bug reports. In this paper, we propose an enhanced summarization approach called TSM by first extending a semantic model used in AUSUM with the anthropogenic and procedural information in bug reports and then integrating the extended semantic model with the shallow textual information used in BRC. We have conducted experiments with a dataset of realistic software projects. Compared with the baseline approaches BRC and AUSUM, TSM demonstrates the enhanced performance in achieving relative improvements of 34.3% and 7.4% in the F1 measure, respectively. The experimental results show that TSM can effectively improve the performance.

key words: bug report summarization, anthropogenic information, procedural information, textual information, semantic model

1. Introduction

In software maintenance, bug reports have abundant information for developers to improve software quality. During the bug fixation process, a developer may refer to historical bug reports for bug resolution. However, past studies indicate that the process of comprehending historical bug reports is time-consuming and tedious[1], [2]. One reason is that the amount of knowledge to be understood is likely huge for each technical point mentioned in the bug reports[1]. Another reason is that to understand a sentence in a bug report may need to search for other similar historical bug reports for more details[1], [2]. Sometimes, the searching process may incur more human-hours to filter out much redundant information or duplicate bug reports. Therefore, the research issue of developing approaches to reduce the comprehension time for developers has received much attention to improve bug fixation process.

Bug report summarization is an emerging research direction to help developers easily find the important information for bug fixation. Many summarization schemes have been proposed[1]–[7]. Because bug reports provide a large amount of textual information to describe software malfunction problems, various text mining techniques are used in the summarization schemes. Figure 1 is an example showing part of a Firefox bug report # 449596 in which there are 13 textual comments from 6 people. In the bug report discussion thread, there are 57 sentences in total.

According to the conversation characteristics, Rastkar et al. propose a conversation-based approach BRC (Bug Report Classifier) to summarize bug report discussion threads using lexical features, participant features, length features, and structural features[1], [4]. Rather than use the shallow textual information as BRC, Mani et al. use semantic information in a summarizer called AUSUM to generate summaries[3]. In AUSUM, a filtering scheme called Noise Reducer (NR) is used to identify sentences into four semantic classes: Question, Code, Investigation, and Others. In PRST (PageRank-based Summarization Technique)[7], Jiang et al. further utilize the additional information in associated duplicate bug reports to generate the summaries. In this study, we focus on generating a summary only based on the single bug report discussion thread without considering the additional information from duplicate bug reports. The main reason is that many bug reports have only a single discussion thread and they do not have associated duplicate bug reports[8].

In this paper, we propose a two-layer semantic model (TSM) by using a semantic filtering model based on the NR model[3] as the first layer to screen out informative sentences and then training the summarizer based on the BRC textual features extracted in the second layer from the informative sentences. Although layering the semantic filtering model and the BRC textual model has the benefits of re-
Table 1 The anthropogenic sentences in the 17 gold standard sentences for Firefox bug report # 449596.

| ID | Sentence |
|----|----------|
| 1.1 | That pref was thought to be for extensions ... |
| 1.2 | While this has worked ... we’ve had several ... |
| 1.3 | Furthermore, there are several code points ... |
| 1.4 | Instead of ... I’d much rather encourage ... |
| 1.5 | This would also make the lives of those ... |
| 1.6 | A problem with this patch is that the ... |
| 1.7 | Some users do not want the ... |
| 1.8 | Then again, we don’t save any data ... |
| 1.9 | I’d rather introduce a different pref or ... |
| 2.1 | You should be able to keep that component ... |
| 2.2 | I currently have only one problem, ... |
| 2.3 | Fair point. I’m not sure. |

Table 2 All anthropogenic sentences in Firefox bug report # 449596.

| ID | Sentence |
|----|----------|
| 1.2 | While this has worked ..., we’ve had several ... |
| 1.4 | Instead of ... I’d much rather encourage ... |
| 5.3 | Some users do not want the ... |
| 6.2 | > Some users do not want the ... |
| 6.5 | Then again, we don’t save any data ... |
| 6.6 | I’d rather introduce a different pref or ... |
| 7.2 | > Then again, we don’t save any data ... |
| 7.3 | Fair point. I’m not sure. |
| 1.9 | I’d rather introduce a different pref or ... |
| 11.2 | Instead of disabling ... you’ll now have ... |
| 11.3 | You should be able to keep that component ... |
| 13.1 | I currently have only one problem, ... |
| 14.2 | > I currently have only one problem, ... |
| 14.3 | You’ve got several options for that: |

Table 3 The steps-to-reproduce segment in KDE bug report # 164545.

| ID | Sentence |
|----|----------|
| 1.7 | Steps to reproduce: |
| 1.8 | 1. Zoom out. |
| 1.9 | 2. Add Activity. |
| 1.10 | 3. Zoom in. |
| 1.11 | 4. Zoom out. |

Producing less informative sentences, the filtering mechanism may mistakenly remove informative sentences. Therefore, two new semantic classes, Anthropogenic and Procedural, are proposed to extend the NR model for improving the sentence identification. Then a supervised logistic regression model is trained according to the BRC features of the sentences in the Question, Code, Investigation, Anthropogenic, and Procedural classes.

The Anthropogenic class is related to the sentences whose subjects are people having some ideas or operations for the bugs. The reason of introducing these two classes is that the NR model considers only Question, Code, Investigation sentences for the summary and ignores Others sentences. However, the Others class may still contain some informative sentences that should be included into the summary. For example, “I currently have only one problem, how to disable the restore after restart.” in the thread of Firefox bug report # 449596 shows that the commenter wants to know the steps to disable the restore function. It is labeled as having important information to help developers to fix the bug. However, the NR semantic model classifies this sentence into the Others class and filters it out. Take Firefox bug report # 449596 in the dataset of [1] for example. Table 1 shows that there are 7 anthropogenic sentences in the 17 gold standard summary sentences. There are totally 14 anthropogenic sentences in the discussion thread of # 449596 wherein 4 sentences are quotations. Table 2 shows all 14 anthropogenic sentences.

The other new semantic class, i.e., the Procedural class, is related to the sentences describing a series of action steps. The steps-to-reproduce information has been considered most relevant for programmers during bug fixing in an empirical survey [9]. For example, Table 3 shows a series of steps to reproduce the malfunctioning situation in KDE bug report # 164545. These steps-to-reproduce sentences are all annotated as the gold standard summary sentences of [1]. However, the sentences like 1.8 and 1.9 in Table 3 are classified into the Others class in the NR model, and thus filtered out.

To investigate the effectiveness of the TSM scheme, we have conducted experiments on the dataset used in [1], [2] from four open source projects: Eclipse, Gnome, KDE, and Mozilla. The experimental results show that TSM can effectively improve the summarization performance. Compared with the baseline approaches BRC and AUSUM, TSM demonstrates the enhanced performance in achieving relative improvements of 34.3% and 7.4% in the F1 measure, respectively.

The rest of the paper is organized as follows. In Sect. 2, we briefly review the previous related work. In Sect. 3, we introduce the summarization process and explain the proposed two-layer semantic model. In Sect. 4, the performance of the proposed two-layer semantic model is analyzed and discussed. Finally, Sect. 5 concludes the paper.

2. Related Work

Document summarization models can be mainly divided into two classes: extractive and abstractive [10]. Extractive summarization extracts representative sentences from the original document as a summary. Abstractive summarization uses semantic relations, domain knowledge, and sentence rules to generate new sentences as a summary. Although abstractive summarization produces summaries similar to handwriting ones, the complexity and the requirement of domain knowledge are two challenging problems. Hence, the extractive approach is commonly used in bug report summarization schemes, e.g., [1], [3], [4], [7].

Because a bug report generally has a conversational discussion thread, Rastkar et al. propose a summariza-
tion scheme called BRC [1], [4] using a conversation-based method [11] to consider 24 shallow textual features of the bug report threads: *Lexical, Participant, Length, and Structural* features. Based on the leave-one-out procedure, BRC uses a supervised logistic regression model to train the classifiers and performs cross validation for performance evaluation. The experiential results of BRC show that it can achieve an F1 score of 0.40 by having a precision score of 0.57 with a relative low recall of 0.35.

In [3], Mani et al. propose AUSUM using the Noise Reducer (NR) model to identify sentences and distinguish their importance in a bug report discussion thread. The NR model classifies sentences into four semantic classes: *Question, Code, Investigation,* and *Others*. The sentences like “Hello”, “Hi”, “Thank you”, and “Thanks” are classified into the *Others* class because these sentences cannot help developers capture critical information and thus they are not suitable in a summary. Therefore, the sentences in *Others* are regarded as noise information and removed. As indicated in [3], four unsupervised summarizers, Centroid [12], MMR [13], Diverse Rank [14], and Grasshopper [15] are studied in AUSUM, and these four unsupervised summarizers outperform BRC by achieving the highest F1 score of 0.50.

In [5], Lotufo et al. use a network model to summarize bug report discussion threads. They consider that developers use a skimming way to read bug report discussion threads and seek useful information. The reading action is similar to the browsing behavior of the Internet users. Thus, they apply the network model to bug report discussion threads and use PageRank to calculate the relationships between sentences. They consider not only the relationships between sentences but also the cosine similarity between a sentence and the topic of bug report discussion threads. A higher similarity score means that the sentence is more representative than others and should be included in the summary. The results in [5] show that their summarizer outperforms BRC with the F1 score of 0.41.

In [7], Jiang et al. propose PRST (PageRank-based Summarization Technique) using the additional information in associated duplicate bug reports to generate the summaries. Moreover, PRST considers three similarity variants in PageRank calculation: the Cosine similarity, the Jaccard coefficient, and the WordNet similarity. To justify the effectiveness of PRST, Jiang et al. build a modified BRC-dataset of 28 bug reports containing duplicate bug reports and a new OSCAR dataset having complete duplicate bug report information. The experiment results demonstrate that PRST is benefited from the associated duplication information.

### 3. Proposed Summarization Model

The proposed two-layer semantic model (TSM) uses two layers to find the informative sentences for bug report summaries. The first layer is an extended NR (ENR) model to preserve informative sentences according to their semantic information. In the second layer, the textual features are extracted from these informative sentences based on the BRC textual model. Finally, a logistic regression model is used to train the TSM classifier based on the textual features.

#### 3.1 Summarization Process

Figure 2 illustrates the summarization process. The first layer performs the sentence filtering based on the ENR model. In the first layer, all sentences of training bug reports are extracted. These sentences are classified into six ENR classes: *Question, Code, Investigation, Anthropogenic, Procedural,* and *Others*. The sentences in the *Others* class are filtered out. Before performing the ENR classification, a keyword dictionary is generated from these sentences. In the second layer, the BRC textual features of the sentences in the five classes are extracted to train the summarizer. In this work, a supervised logistic regression model is used for summarization model training. For testing, all testing bug reports are also processed through the two layers as the training bug reports. The summarization model ranks the sentences of the testing bug report threads according to their probability scores. The sentences with high scores are selected to generate the corresponding summaries.

#### 3.2 TSM Semantic Model

In this paper, we propose an enhanced semantic model by extending the NR model [3] with two new semantic classes: *Anthropogenic* and *Procedural*. Because a sentence may satisfy the requirements of several classes, a rule-based al-
algorithm. Algorithm 1, is used to assign each sentence to one class. The sentences in Question, Code, Anthropogenic, and Procedural are usually informative. They are preserved for summary generation. On the contrary, the sentences in Others are not considered for summary generation.

Algorithm 1 Sentence identification algorithm.

Input:  
- $R_i$: a sentence  
- $L_T$: the word count threshold

Output:  
- $s_j$: is in the informative classes or not
  1. if $s_j$ meets the regular expressions of Question then  
  2. $s_j$ is classified into Question
  3. else if $s_j$ meets the regular expressions of Code then  
  4. $s_j$ is classified into Code
  5. else if $s_j$.length $\geq L_T$ AND $s_j$ contains two keywords then  
  6. $s_j$ is classified into Investigation
  7. else if the subject of $s_j$ is a personal pronoun AND $s_j$ contains one keyword then  
  8. $s_j$ is classified into Anthropogenic
  9. else if $s_j$.starts with a number then  
  10. $s_j$ is classified into Procedural
  11. else
  12. $s_j$ is classified into Others
  13. end if

3.2.1 Anthropogenic Class

The Anthropogenic class is used to identify the sentences whose subjects are people having actions on bugs or describing the bug information. Sentences starting with the personal pronouns, such as “I” and “you”, may contain much information of software bugs. These personal pronouns are important clues to classify these informative sentences into the Anthropogenic class.

To avoid that Anthropogenic sentences contain useless information like “I fully agree with Dan aswell.” in KDE bug # 188311, we construct a keyword dictionary to decide the importance of the sentences. Each bug report $R_i$ has its own keyword dictionary $KD_i$ generated from the bug report discussion thread according to the TF-IDF (Term Frequency-Inverse Document Frequency) values of terms in the thread. If the TF-IDF value of a term $t_k$ is larger than the average TF-IDF value of $R_i$, $t_k$ is included in $KD_i$. If a sentence has a keyword and its subject is a personal pronoun, it is identified as an Anthropogenic sentence.

3.2.2 Procedural Class

Sentences describing the steps-to-reproduce for the problem encountered by reporters usually have important information for bug fixation. These sentences are often short as shown in Fig. 3. They may be also described by general words because reporters may not have specific domain knowledge. Therefore, these sentences cannot be identified into Question, Investigation, or Code. However, they can help developers to know how to reproduce the problem.

We construct regular expression rules to identify sentences starting with numbers, such as “1.” or “(2)”. If the beginning of a sentence satisfies the regular expression rules, it is identified as a procedural sentence for the following classification.

3.2.3 Question, Code, and Investigation Classes

The Question, Code, and Investigation classes have been used in the NR semantic model [3]. In TSM, we use the regular expression rules as shown in Fig. 3 to identify the Question sentences. If a sentence satisfies one of these regular expressions or its sentence parse tree has an SQ or SBARQ node1, it is identified as a Question sentence. For example, “can you add a pref for this, or some other way to do it?” (ID=13.4) in Firefox bug report # 449596 is a Question sentence.

Sentences of the Code class include code fragments, stack traces, or command outputs. These Code sentences may have important information to explain the problems or suggest the possible solutions for developers [9]. Various patterns are used to decide whether a sentence is a code sentence or not, because certain words and punctuation marks usually appear in fixed positions. Figure 4 shows the rules used to identify the Code sentences. If a sentence satisfies one of these rules, it is identified as a Code sentence. For different software projects, the rules should be adjusted according to the programming specifications.

Sentences of the Investigation class may provide suggestions that can help developers further investigate the malfunction situations in detail. The Investigation sentences are usually descriptions related to the problems. We use the keyword dictionary $KD_i$ to identify the investigation sentences for each bug report $R_i$. A sentence $s_j$ is regarded as an investigative sentence if its length $L_j \geq L_T$ and it has more than two keywords. For example, “That pref was thought to be for extensions which wanted to completely replace our own Session Restore functionality.” (ID=1.1) in Firefox bug report # 449596 is an Investigation sentence.

\* In this work, we used Stanford Parser (https://nlp.stanford.edu/software/lex-parser.html) for parsing. SQ is a yes/no question and SBARQ is a direct question introduced by a wh-element.
3.2.4 Others Class

Sentences in the Others class include the greeting sentences like “Hello” and “thank you for your support”. These sentences are less informative and not used to train the summarizer. In Algorithm 1, if a sentence cannot be classified to the other five classes, it is included in the Others class. The sentences of the Others class are removed before the following textual analysis.

3.3 Textual Information Processing

The BRC textual model is used to extract 24 features for the following classifier training. Before the feature extraction, all sentences are preprocessed at first:

- **Tokenization**: A sentence is divided into tokens which are the basic language units. We use the Stanford Tokenizer\(^1\) for tokenization.
- **Stopword Removal**: After tokenization, stopword removal is performed. A stopword is a word like “a”, “an”, “the”, or “is”, which is not helpful for classification. We use a stopword list\(^1\) containing 571 words to remove these stopwords.
- **Stemming**: After stopword removal, all words are converted to their word roots using Porter Stemmer\(^1\). Therefore, words in similar forms are represented using the same term. For example, “create” and “created” are converted to “creat”.

After word preprocessing, the BRC rules are used to extract the textual features of each sentence as shown in Table 4. Because a bug report discussion thread is in a conversation form, a turn is defined as a discussion sequence by one participant. We follow the work of [1] to calculate Lexical, Participant, Length, and Structural features. For example, TLOC is the line number of a sentence in a turn. To calculate the Lexical features, \(S_{pr}\) and \(T_{pr}\) need to be first calculated as follows:

\[
S_{pr}(t) = \max_S P(S|t),
\]

where \(t\) is a term, \(S\) is a participant, and \(P(S|t)\) is the probability of term \(t\) used by a participant in the discussion thread. \(S_{pr}\) is the maximum probability \(P(S|t)\) of term \(t\) as its score. The purpose of this score is to find terms that are related to the topic of conversation because participants usually use specific terms frequently according to their interests and expertise.

Another term score \(T_{pr}\) is calculated as follows:

\[
T_{pr}(t) = \max_T P(T|t),
\]

where \(t\) is the given term and \(T\) is a turn. The purpose of the \(T_{pr}\) score is to find the shifted topic terms which appear in a number of turns as the discussion evolves. The \(S_{pr}\) and \(T_{pr}\) scores are used to represent the importance of words in the discussion.

The word-based entropy features like THISSENT and PENT are derived according to a heuristic rule: if a sentence contains words of many different part-of-speech (POS) types like *Noun* and *Verb*, it is informative. The entropy \(E_w(s)\) of a string \(s\) is calculated as follows:

\[
E_w(s) = \frac{\sum_{i=1}^{N} p(x_i) \times -\log(p(x_i))}{(\frac{1}{2} \times -\log(M)) \times M}
\]

where \(x_i\) is a POS type in the string \(s\), \(p(x_i)\) is the probability of \(x_i\) appearing in \(s\), \(N\) is the number of POS types in \(s\), and \(M\) is the string length. The denominator represents the entropy of the ideal case. It is used to normalize the actual entropy of the string. The value of the entropy \(E_w(s)\) is between 0 and 1. A larger \(E_w(s)\) means that the string contains richer information. If each POS type of the string \(s\) just appears once, \(E_w(s) = 1\).

As mentioned in [1], [4], the CWS score is used to measure the cohesion of the conversation. CWS is calculated as the number of terms that occur in other turns besides the current turn.

### Table 4 24 textual features used in BRC.

| Features               | Class Description |
|------------------------|-------------------|
| TLOC                   | Str. Position of the sentence in turn |
| CLOC                   | Str. Position of the sentence in thread |
| TPOS1                  | Str. Time from beginning to current turn |
| TPOS2                  | Str. Time from current turn to end |
| SPAU                   | Str. Time btw. current and next turn |
| PPAU                   | Str. Time btw. prev. and current turn |
| BIGAUTH                | Prt. BOOLEAN for first participant |
| DOM                    | Prt. Word # used by the participant |
| SLEN                   | Len. Normalized global sentence length |
| SLEN2                  | Len. Normalized local sentence length |
| MXT                    | Lex. Max \(T_{pr}\) of words |
| MNT                    | Lex. Mean \(T_{pr}\) of words |
| SMT                    | Lex. Sum of \(T_{pr}\) of words |
| MXS                    | Lex. Max \(S_{pr}\) of words |
| MNS                    | Lex. Mean \(S_{pr}\) of words |
| SMS                    | Lex. Sum of \(S_{pr}\) of words |
| COS1                   | Lex. Sim. of prev. and next sentences using \(S_{pr}\) |
| COS2                   | Lex. Sim. of prev. and next sentences using \(T_{pr}\) |
| CENT1                  | Lex. Sim. of current and rest sentences using \(S_{pr}\) |
| CENT2                  | Lex. Sim. of current and rest sentences using \(T_{pr}\) |
| THISSENT               | Lex. Entropy of the current sentence |
| PENT                   | Lex. Entropy from beginning to current sentence |
| SENT                   | Lex. Entropy from current sentence to end |
| CWS                    | Lex. Approximated ClueWordScore [1] |

---

\(^1\)http://nlp.stanford.edu/software/tokenizer.shtml

\(^1\)http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewisu04a11-smart-stop-list/english.stop

\(^1\)http://tartarus.org/martin/PorterStemmer/index-old.html

---

The TSM summarizer uses a logistic regression model to train the classifier as BRC [1]. The logistic regression model [16] is a nonlinear regression model commonly used to estimate the probability of a binary response based on the categorical data. For a sentence \(s\) from the training set, it can be represented as a feature vector \(X = \{x_1, x_2, \cdots, x_n\}\),
which is the vector of the predictor variables. In this work, each sentence is represented with the 24 BRC features ($n = 24$). The probability $P(y = 1 | X)$ of the sentence assigned to the class of summary can be calculated as follows:

$$P(y = 1 | X) = \frac{e^{w_0 + w_1 x_1 + \cdots + w_n x_n}}{1 + e^{w_0 + w_1 x_1 + \cdots + w_n x_n}},$$

where $W = \{w_0, w_1, \cdots, w_n\}$ is the vector of the corresponding model parameters which are estimated in the logistic regression model to maximize the probability accuracy based on the training data $X$, and $y$ denotes whether the sentence is included in the summary. The logit transformation of $P(y = 1 | X)$ is calculated as follows:

$$g(P(y = 1 | X)) = \log \left( \frac{P(y = 1 | X)}{1 - P(y = 1 | X)} \right) = w_0 + WX.$$

With the trained logistic regression model, each testing sentence has its estimated $P(y = 1 | x)$ score. The TSM summarizer uses these scores to generate the final summary result.

3.5 Summary Generation

Before summary generation, the size of the summary needs to be decided first. The summary size is controlled according to a compression rate, which is a ratio of the size of the summary to the size of the original documents. If the compression rate is too low, the information provided by the summary is incomplete. According to the past study [17], the compression rate of 5–30% is an acceptable range. Because this work takes BRC [1] and AUSUM [3] as the baseline for performance comparison, the compression rate is 25%, which is also used in these schemes.

Because the logistic regression model gives each testing sentence a probability score, all testing sentences are sorted in a descending order according to their probability scores. Then, the summary generation process is started from the beginning of the list to select the sentences having high scores until the size of the summary meets the compression rate requirement.

4. Experiments and Discussion

4.1 Dataset

We use the BRC dataset [1] for four open source projects: Eclipse, Gnome, KDE, and Mozilla. This dataset and its variants are widely used in the related research [3], [5], [7]. In the dataset, 9 bug reports are selected from each software project, and there are 36 bug reports in total. The BRC dataset does not contain short bug reports and bug reports consisting mostly of long stack traces and code segments. The main reason is that they are usually not read by programmers [1]. Table 5 is the characteristics of the dataset.

4.2 Evaluation Metrics

To evaluate the effectiveness of the proposed TSM summarizer, we use Precision and Recall as the evaluation metrics. In the dataset, gold summaries (GS) are annotated for all bug reports. Precision represents the ratio of the extracted GS sentences to the all extracted sentences:

$$\text{Precision} = \frac{\text{# of extracted GS sentences}}{\text{# of all extracted sentences}} = \frac{tp}{tp + fp},$$

where $tp$ (true positives) is the number of extracted GS sentences, and $fp$ (false positives) is the number of extracted sentences which are not in GS. Recall is the ratio of the extracted GS sentences to the total GS sentences:

$$\text{Recall} = \frac{\text{# of extracted GS sentences}}{\text{# of all GS sentences}} = \frac{tp}{tp + fn},$$

where $fn$ (false negatives) is the number of GS sentences which are not extracted.

Because Precision and Recall represent two aspects of extraction performance, we calculate their F1 score which is a harmonic mean of Precision and Recall:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

4.3 Experimental Results

In the experiments, we implemented the TSM semantic model and discussed its performance with different combinations of the following semantic features: the basic NR model (N), the Anthropogenic features (A), and the Procedural features (P). We also implemented the BRC-based classifier (BRC) to learn the textual information using a logistic regression model. For the sake of performance comparison, we used $L_T = 5$ in the experiments as [3]. In this work, the compression rate was 25%. The experiments used 36-fold cross validation to evaluate the performance of the summarizer. Each fold contained one bug report. For each fold, the classifier was trained with other 35 bug reports and tested with the bug report in the fold. This step is repeated 36 times with a different fold as the testing set.

The experiment results are shown in Table 6. The performance data of the original BRC (Orig_BRC) [1], [4] and

| # of bug reports | 36
| Min sentence # in a turn | 1
| Max sentence # in a turn | 151
| Total sentences | 2360
| # of summaries | 454
| # of non-summaries | 1906

Table 5 The information of the dataset.

| Summarizer | Precision | Recall | F1 |
| --- | --- | --- | --- |
| Orig_BRC [1], [4] | 0.57 | 0.35 | 0.40 |
| AUSUM [3] | 0.49 | 0.51 | 0.50 |
| TSM (BRC+N) | 0.509 | 0.543 | 0.525 |
| TSM (BRC+N+A) | 0.519 | 0.538 | 0.528 |
| TSM (BRC+N+F) | 0.507 | 0.550 | 0.528 |
| TSM (BRC+N+A+F) | 0.520 | 0.541 | 0.530 |

Table 6 The performance of the baseline summarizers and the TSM summarizer.
AUSUM [3] are directly used from their publications as the baselines for comparison.

From Table 6, we can find that the simplest TSM model (BRC+N) outperforms Orig_BRC and AUSUM in Recall and F1. Although Orig_BRC achieves the best Precision performance, it has a relative low Recall score. It shows that Orig_BRC cannot provide many informative sentences in the summaries. From Table 6, we can also find that three TSM models (BRC+N+A, BRC+N+P, BRC+N+A+P) have better F1 scores.

Among four TSM models, BRC+N+P has the lowest Precision score because some less informative Procedural sentences are extracted to the summaries. However, BRC+N+P has the highest Recall score because the Procedural sentences are usually short such that more sentences can be extracted to the summaries. On the contrary, the summaries in BRC+N+A usually have fewer sentences under the restriction of the compression rate because the Anthropogenic sentences are usually longer. Therefore, BRC+N+A has a higher Precision score than BRC+N+P. Considering both Anthropogenic features and Procedural features, BRC+N+A+P achieves the highest F1 score of 0.530.

In the experiments, we further discussed the impacts of the length threshold \( L_T \) because it may influence the filtering output of the basic NR model. Therefore, the performance of the TSM summarizer is also influenced. In the experiments, the length threshold \( L_T \) was set to 4, 5, and 6 to investigate its impacts.

Table 7 shows the results. When \( L_T = 4 \), the first ENR layer preserves many short sentences. Therefore, more noise information is included and the performance is decreased. When \( L_T = 6 \), many moderate sentences are also filtered out. However, the F1 scores of BRC+N, BRC+N+A, and BRC+N+P are hindered because some removed sentences are informative. On the contrary, BRC+N+A+P still achieves the best F1 score of 0.537. This is because many Anthropogenic and Procedural sentences are preserved. However, the Precision scores of all TSM models are decreased when \( L_T = 6 \). This is because some informative moderate Investigative sentences are removed. Compared with Orig_BRC and AUSUM, TSM demonstrates the enhanced performance in achieving relative improvements of 34.3% and 7.4% in the F1 measure, respectively.

### 5. Conclusion

In a process of bug fixing, developers may need to read previous bug reports for bug resolution. However, this comprehension process is usually time-consuming and tedious. Bug report summarization is an approach to help developers quickly capture important information.

In this paper, we propose a two-layer semantic model (TSM) considering both the semantic information and the shallow textual information. Moreover, two new semantic classes, Anthropogenic and Procedural, are proposed to identify informative sentences. We have conducted empirical experiments to evaluate the performance. The results show that the proposed TSM (BRC+N+A+P) model can effectively achieve the best 0.537 F1 score with relative improvements of 34.3% and 7.4% in comparison with Orig_BRC and AUSUM.

Several issues need to be investigated further. First, some informative sentences are still classified into Others. A more accurate semantic model should be devised. Second, bug reports have many jargon words. A dictionary is thus needed to help word preprocessing.

### References

[1] S. Rastkar, G.C. Murphy, and G. Murray, “Summarizing Software Artifacts: A Case Study of Bug Reports,” Proc. 32nd ACM/IEEE International Conference on Software Engineering (ICSE ’10), pp.505–514, 2010.

[2] R. Lotufo, Z. Malik, and K. Czarnecki, “Modelling the ‘Hurried’ Bug Report Reading Process to Summarize Bug Reports,” Proc. 2012 IEEE International Conference on Software Maintenance (ICSM ’12), pp.430–439, 2012.

[3] S. Mani, R. Catherine, V.S. Sinha, and A. Dubey, “AUSUM: An Approach for Unsupervised Bug Report Summarization,” Proc. ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE ’12), pp.11–11:11, 2012.

[4] S. Rastkar, G.C. Murphy, and G. Murray, “Automatic Summarization of Bug Reports,” IEEE Trans. Software Eng., vol.40, no.4, pp.366–380, April 2014.

[5] R. Lotufo, Z. Malik, and K. Czarnecki, “Modelling the ‘Hurried’ Bug Report Reading Process to Summarize Bug Reports,” Empir. Softw. Eng., vol.20, no.2, pp.516–548, April 2015.

[6] N. Nazar, Y. Hu, and H. Jiang, “Summarizing Software Artifacts: A Literature Review,” J. Comput. Sci. Technol., vol.31, no.5, pp.883–909, Sept. 2016.

[7] H. Jiang, N. Nazar, J. Zhang, T. Zhang, and Z. Ren, “PRST: A PageRank-Based Summarization Technique for Summarizing Bug Reports with Duplicates,” Int. J. Softw. Eng. Know., vol.27, no.6, pp.869–896, 2017.

[8] J. Anvik, L. Hiew, and G.C. Murphy, “Coping with an Open Bug Repository,” Proc. 2005 OOPSLA Workshop on Eclipse Technology eXchange (ETX ’05), pp.35–39, 2005.

[9] N. Bettenburg, S. Just, A. Schröter, C. Weiß, R. Premraj, and T. Zimmermann, “Quality of Bug Reports in Eclipse,” Proc. 2007 OOPSLA Workshop on Eclipse Technology eXchange (eclipse ’07), pp.21–25, 2007.

[10] U. Hahn and I. Mani, “The Challenges of Automatic Summarization,” Computer, vol.33, no.11, pp.29–36, Nov. 2000.

[11] G. Murray and G. Carenini, “Summarizing Spoken and Written Conversations,” Proc. 2008 Conference on Empirical Methods in
Natural Language Processing (EMNLP ’08), pp.773–782, 2008.

[12] D.R. Radev, H. Jing, M. Styš, and D. Tam, “Centroid-based Summarization of Multiple Documents,” Inform. Process. Manage., vol.40, no.6, pp.919–938, Nov. 2004.

[13] J. Carbonell and J. Goldstein, “The use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries,” Proc. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’98), pp.335–336, 1998.

[14] Q. Mei, J. Guo, and D. Radev, “DivRank: The Interplay of Prestige and Diversity in Information Networks,” Proc. 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’10), pp.1009–1018, 2010.

[15] X. Zhu, A.B. Goldberg, J. Van Gael, and D. Andrzejewski, “Improving Diversity in Ranking using Absorbing Random Walks,” Proc. 2007 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2007), pp.97–104, 2007.

[16] D. Hosmer and S. Lemeshow, Applied Logistic Regression, New York: Wiley, 1989.

[17] J.-Y. Yeh, H.-R. Ke, W.-P. Yang, and I.-H. Meng, “Text Summarization using a Trainable Summarizer and Latent Semantic Analysis,” Inform. Process. Manage., vol.41, no.1, pp.75–95, Jan. 2005.

Cheng-Zen Yang received the B.S. and M.S. degrees from Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan, in 1988 and 1990, respectively. Then he received his Ph.D. degree from Department of Computer Science and Information Engineering, National Taiwan University in 1996. Currently, he is an associate professor in Yuan Ze University. His research interests include software testing, machine learning, information retrieval, and high-speed networking.

Cheng-Min Ao received the M.S. degree from Department of Computer Science and Engineering, Yuan Ze University, Taiwan, in 2015. His research interests include software testing, information retrieval, and text mining.

Yu-Han Chung received the B.S. degree from Department of Computer Science and Engineering, Yuan Ze University, Taiwan, in 2017. Currently, she is persuading her M.S. degree in Department of Computer Science and Engineering, Yuan Ze University. Her research interests include information retrieval, document summarization, and text mining.