International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-10, Issue-6, November 2022
https://doi.org/10.55524/ijircst.2022.10.6.18
Article ID IRP1331, Pages 101-109
www.ijircst.org

Software Bug Reports: Automatic Keyword and Sentence-Based Text Summarization Using Artificial Intelligence

Zaid Altaf[1], and Ashish Oberoi[2]

[1]M. Tech Scholar, Department of Computer Science & Engineering, RIMT University, Mandi Gobindgarh, Punjab, India
[2]Assistant Professor, Department of Computer Science & Engineering, RIMT University, Mandi Gobindgarh, Punjab, India

Correspondence should be addressed to Shakir Fayaz Reshi; abrarrashid57@gmail.com

ABSTRACT- The purpose of text summarization is to quickly and accurately extract the most important data from papers. The proposed unsupervised method seeks to synthesise complete and informative bug reports (software artefacts). The suggested approach employs Rapid Automatic Keyword Extraction and the term frequency-inverse document frequency method to identify applicable keywords and phrases. During the sentence extraction procedure, fuzzy C-means clustering is used to prioritise sentences that have a high degree of membership in each cluster (beyond a predefined threshold). The selection of sentences is performed by a rule-engine. Information is extracted using keywords and sentences chosen by the clustering process, and the rules are developed using domain knowledge. The proposed method produces a logical and well-organized summary of apache bug reports. The retrieval summary is improved with the help of hierarchical clustering by removing unnecessary details and rearranging them. The Apache Project Bug Report Corpus (APBRC) and the original Bug Report Corpus are used to evaluate the effectiveness of the proposed method. Measures of performance such as precision, recall, pyramid precision, and F-score are used to evaluate the results. Experiment results demonstrate that our proposed method significantly outperforms the state-of-the-art baseline methods like BRC and LRCA. In addition, it achieves substantial gains compared to prior art unsupervised methods as Hurried and centroid. It extracts the most relevant keyword phrases and sentences from each cluster to offer comprehensive coverage and a coherent summary. The average values for precision, recall, f-score, and pyramid precision on the APBRC corpus are 78.22%, 82.18%, 80.10%, and 81.66%, respectively.

KEYWORDS- Rapid Automatic Keyword Extraction, Text Summarization, Fuzzy C-Means, Bug Reports, Hierarchical Clustering, Rule Engine.

I. INTRODUCTION

Many domains' information is now online. Reading complete text documents and finding essential information is difficult and time-consuming with so much material. Text summary automatically extracts relevant information. Summarizing a text document takes human intelligence to extract useful information. Document, essay, news, and email summarization have all employed automatic text summarization [1–6]. Many open source projects have their bug reports handled by Jira, Bugzilla, or another software repository [7-9]. Therefore, many procedures, such as the triage of reports [13–15], the resolution of bugs [10, 11], and the detection of duplicate bug reports [10, 11], are affected. have been automated in order to handle the large number of reports. Software testers and developers must read hundreds-sentence issue reports to complete their duties. Since bug report history is not a generic text summary, testers and developers need subject knowledge. This research summarises bug reports, the most valuable software project artefacts. Name, brief description, BugID, detailed description, contributor comments, and more are all included. The information gathered from this helps in the search for and resolution of software defects. In order to aid engineers in fixing bugs, there is new study focussing on bug summaries. Reading and understanding a bug report is tiresome.

Abstractive and extractive algorithms exist in literature. Abstract summarization modifies text semantics, word order, and natural language with the same context. Deep learning advances this field. Researchers have found that CNNs, RNNs, RL, and GANs all perform very well when it comes to predicting outcomes. Since deep learning is supervised and conventional golden summaries are not available in all domains, training data is the key drawback. Extractive summarization condenses text by extracting sentences in the same order and language. Both supervised [19–21] and unsupervised [22, 23] methods have been presented in the literature to summarise bug reports. To create a Bug Report Corpus, Rastkar et al. [19] supervised compilation of 36 bug reports from open source projects. Each report of a bug is distinct in 24 different ways, such as its vocabulary, contributors, length, and organization. Annotator-created "golden summaries" were used to train a logistic regression classifier. Statistical analysis revealed a 57% accuracy rate, 35% recall, 40% f-score, and 66% accuracy rate in the pyramid. Attempting to enhance [19], Jiang et al. presented PRST [20]. The authors created Modified Bug Report Corpus using 36 BRC bug reports and their duplicates [19]. (MBRC). Page-rank algorithm calculates textual similarity, and logistic regression classifier calculates sentence probability. Merged results were summarised using high-probability phrases. There was a modest uptick in accuracy, pyramid accuracy, recall, and f-score.

In contrast, an unsupervised method ranks sentences according to some metric and uses the highest-scoring ones to compile a summary. Lotufo et al. [23] devised an

unsupervised method for generating summaries by observing how developers interacted with a lengthy report. Multi-model resampling (MMR), centroid sampling, diverse rank, and grasshopper sampling were all used by Mani et al. Keyword analysis, lexical matching, and sentence weight were used in the aforementioned studies [19, 20].

We use an unsupervised strategy to summarise bug reports using keyword- and sentence-based characteristics. Extracting features from a set of keywords is done with the help of tf-idf and RAKE. Statistical methods geared toward corpora have been used in keyword extraction methods [24, 25]. Using natural language processing to identify speech chunks and then combining that information with statistical, machine learning, or supervised methods, document-oriented approaches [26] tackled this issue. RAKE, a language-independent, domain-independent, unsupervised algorithm, avoids these issues. It provides reasonable precision, simplicity, and computing efficiency [27]. Fuzzy C-means clustering is utilised for sentence-based feature extraction instead of length, location, title word, thematic word, and others [19], [28], [29].

First, BRCS extracts bug reports from five Apache Software Foundation projects [30]. APBRC contains one-line descriptions, extended descriptions, and contributor comments from 21 bug reports (Apache Project Bug Report Corpus). Standard preparation processes separate bug report content into sentences.

Keyword extraction precedes feature extraction. Keyword features are tf-idf and RAKE keywords. RAKE scores concepts/words by assessing their sentence content. Text content words are used to calculate each keyword or keyword phrase's score. RAKE extracts longer phrases with more meaning than tf-idf. Authors use tf-idf to identify unigram terms not taken by RAKE.

The best method for classifying sentences into groups based on their shared characteristics was found using a combination of Gap Statistics, K-means, Silhouette, and with-in-squares. Bug reports use fuzzy C-means clustering based on optimal cluster. Each word or phrase is placed in the group that has the smallest Euclidean distance to its center. Select high-membership sentences from each cluster.

The third step is selecting sentences to use in the extractive summary once all features have been culled. Reduced by 20%, the bug report. Results showed that the method was superior to state-of-the-art algorithms in terms of accuracy, recall, f-score, and pyramid accuracy.

Hierarchical clustering generates a brief summary after the summary. Dendrograms rearrange sentences and choose the superior of two identical ones to create a streamlined summary free of superfluous information.

This document is structured. Section II describes the study's motivation. Section IV discusses research methodology, and Part III presents preliminary ideas for the proposed approach. Step-by-step instructions for the proposed method are presented in Section V, and the results are discussed in Section VI. Possibilities that the validity claims may not hold up are detailed in Section VII. The following section, Eight, discusses relevant research. The final section of the paper concludes the discussion.

II. LITERATURE REVIEW

Software systems value bug reporting most. It includes developer comments, predefined fields, an id, a description, and a title. In the past, 200 bug reports in the Mozilla open-source project meant 275 bug reports. Many sentences are repeated in each bug report, making it difficult to read and understand for both testers and developers. Time spent by software testers has been reduced thanks to a new technique for summarising bug reports.

Our goal is to create a unique, revolutionary automatic text summarization system that detects bug report domain knowledge and generates high-quality bug summaries. Sentences that start with ", ", "tmp field," "sql," ", ", "public static," and "=" are examples of summary sentences. The most important parts of a bug summary, the description and comments, are inaccessible to text mining techniques like tf-idf, unigram, bigram, and centroid. The author employed RAKE to extract code snippets from bug report textual data, which has not been done before [19–21].

Instead of optimization, centroid-based fuzzy clustering is used to identify key sentences. Fuzzy clustering outperforms unsupervised methods [22, 23]. A rule engine creates an unsupervised bug report summary by combining keywords and sentences. Hierarchical clustering removes unnecessary sentences based on dendograms and re-ranks them to create a compact summary.

III. PRELIMINARY CONCEPTS

This section discusses bug report summarising concepts. It has automatic keyword extraction from bug reports, fuzzy clustering, and hierarchical clustering.

A. Text Pre-Processing

The raw bug reports must be processed before a summary can be created.Segmentation, tokenization, stop word removal, punctuation removal, and stemming are pre-processed.

Segmentation: Sentences from bug reports are parsed using delimiters. The sentences that were retrieved are then sorted according to the reported bug.

Tokenization: Segmenting sentences into meaningful words, symbols, and phrases.

Stopwords are deleted from textual material without semantic information in this step.

Punctuation and special characters like questioning and exclamation are eliminated.

Stemming removes suffixes and prefixes to return words to their roots. Presentation becomes "present."

After these stages, text data becomes a Document term matrix (DTM). It shows a document's phrase frequency.

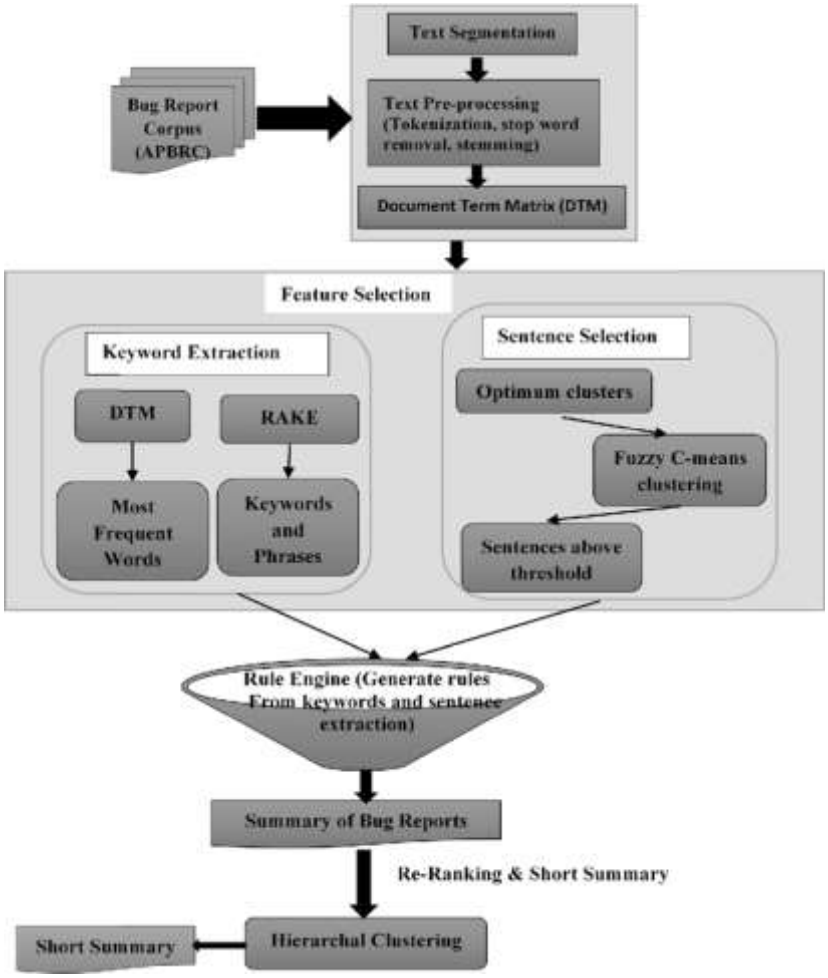


Figure 1: Proposed Process Flow Diagram

B. Rapid Automatic Keyword Extraction (Rake)

Rake can pull keywords out of text regardless of the domain. It breaks down texts into potential keywords, which are word strings that describe the text's subject matter. By analysing these content words' co-occurrence in candidate keywords, it extracts them. Rake divides text into words for keyword extraction. Then, the phrases are separated from one another by using stop words and phrase delimiters. The potential keywords are evenly spaced throughout the text. The frequency of individual content words within a potential keyword is displayed using a word co-occurrence matrix. Words that are up for consideration as part of a text are called "candidate words." Each keyword is scored after identification. Each candidate keyword's score is the sum of its content words [30, 31]. This is how keywords are scored:

- The freq value of a given textual document is first used to calculate the freq value of each content word.
- After the frequency calculation is complete, the value denoted by deg is the word's degree. To quantify, we count how many times the content word appears in potential keywords.
- Finally, the ratio of a word's degree to its frequency is calculated and is shown as.

$$Deg(CW)/Freq(CW) \quad (1)$$

Here, using the description of bug ID HDFS-7707, we can see how the scores of candidate keywords are calculated (below) and in Table 1.

$$\begin{aligned} \text{Score (edit log corruption)} &= \text{score (edit)} + \text{score (log)} \\ &\quad + \text{score (corruption)} \\ &= 2.4 + 2.4 + 3 = 7.8 \quad (2) \\ \text{Score (edit log)} &= \text{score (edit)} + \text{score (log)} \\ &= 2.4 + 2.4 = 4.8 \quad (3) \end{aligned}$$

The frequency of edits is greater than that of corruption, and the degree of edits is greater than that of corruption, but the degree of corruption to the frequency of edits is greater than the degree of edits to the frequency of edits (edit).

Therefore, a fast automatic keyword extraction method is used to choose commonly used words and lengthier candidate keywords. RAKE is superior to tf-idf, text rank, ngram with tag, and other keyword extraction methods in terms of accuracy and recall. [30], [31].

C. Fuzzy Clustering

Unsupervised machine learning clusters data. Data points are clustered by distance or similarity. Hard and soft (fuzzy) clustering are the main methods. Hard clustering assigns one value to each data point, 0 or 1.

Each data point may belong to more than one cluster depending on the results of the soft clustering method. In the field of fuzzy clustering, the most well-known method is called "fuzzy C-means" [32]. Fuzzy c-means reduces the

Euclidean distance between data points and cluster centers. FCM randomly chooses cluster centres and assigns membership values to each data point. It updates the cluster centre and degree of membership after each iteration using equation (4). (5).

$$M_{ij} = 1 \Big/ \sum_{K=1}^{C} (E_{ij}/E_{ik})^{(\frac{2}{f}-1)} \quad (4)$$

$$C_j = (\sum_{i=1}^{n} \frac{(M_{ij})^f x_i)}{(M_{ij})^f} \quad (5)$$

The objective function, or the separation of the ith data point from the jth cluster center, is minimised over the course of a set number of iterations.

$$O_f = \sum_{i=1}^{n} \sum_{j=1}^{c} (M_{ij})^f | \left|x_i - c_j\right| |^2 \quad (6)$$

D. Hierarchical Clustering

Like other unsupervised clustering methods, hierarchical clustering [36] groups similar data sets together. This phenomenon can either bring people together or drive them apart. Different from the bottom-up method of agglomerative clustering, which assigns each data point to its own cluster, the top-down method of divisive clustering divides a larger cluster into several smaller, more manageable clusters. This strategy uses agglomerative hierarchical clustering. By doing so, the approach treats each data point as its own cluster and calculates the distance between them. Several neighbouring clusters are combined into one larger cluster if their characteristics are sufficiently similar. After each new cluster is generated, its proximity to the others is calculated and combined until a single cluster is formed. Hierarchical clustering yields a tree-like structure called a dendogram, which documents numerous sequences of mergers. Furthermore, a wide variety of approaches are used to calculate the closeness or similarity of two clusters. In this paper, we use a linking strategy that is statistically average. Every single point in one cluster is measured against every single point in another cluster to determine their average distance. To determine this value, we use the equation Where, DiE C1 and DjE C2 [36].

$$Sim\,(C1, C2) = \sum_{i=1}^{n} \sum_{j=1}^{n} Sim(D_i, D_j)/|C1|^*|C2|^* \quad (7)$$

IV. METHODOLOGY

Research concepts are defined here. Section IV includes the pseudocode of the proposed method and then the framework and its modules.
This method centres on four overarching ideas:

- Detailed coverage of the primary topic area.
- Ratio of compressed data to the uncompressed source.
- To reorder the summary so that the most important and relevant information appears first.
- Spread out data sets with as little overlap as possible.

A. Corpus Creation and Text Pre-Processing

The APBRC has received 21 bug reports concerning Apache. A bug report is parsed for one-liners, detailed descriptions, and comments, then the sentences are extracted and organised into a corpus. Tokenization, stopword elimination, and stemming pre-process the sentences. DTM construction follows pre-processing. "tm" and "NLP" R packages implement the process.

B. Feature Extraction

Features are taken from preprocessed text. Everything of interest in the text is taken out. All features of the text exist at the word or sentence level. Various combinations of text features have been extracted to improve bug report relevancy and coverage. Explained are recommended approach features.

V. KEYWORD FEATURES

Extractive summaries use sentences. To choose important sentences, extract important words/keywords. You can use one of two techniques to extract keywords:
Prevalence of Terms in Section 1.1 Documentation Frequency, Inverse (Tf-Idf) Each word's term frequency and inverse document frequency in a given document are computed.
1.2 A Fully Automated, Rapid Extraction Procedure (RAKE)
RAKE may pull keywords from documents. It uses commas and other delimiters to separate phrases. To determine which words and phrases within a text are most relevant, it uses a frequency and co-occurrence analysis. The total score for a keyword phrase is the average score of its constituent content words. Section III illustrates scoring (B).

A. Sentence Level Features

Once sentences containing the most relevant keywords have been selected, the attributes of those sentences can be analyzed. Some characteristics of complete sentences are listed below.
2.1 Sentence position: It stipulates that summary sentences should always include document leading sentences. Calculated:

$$P\,(S_i) = 1 - (^{i\,-\,1}/_{N}) \quad (8)$$

2.2 Sentence Length: Summaries should include both short and long sentences, but longer ones are more crucial. Calculated:

$$(S_i) = \frac{number\ of\ words\ in\ S_i}{Total\ numnber\ of\ words\ in\ longest\ sentence} \quad (9)$$

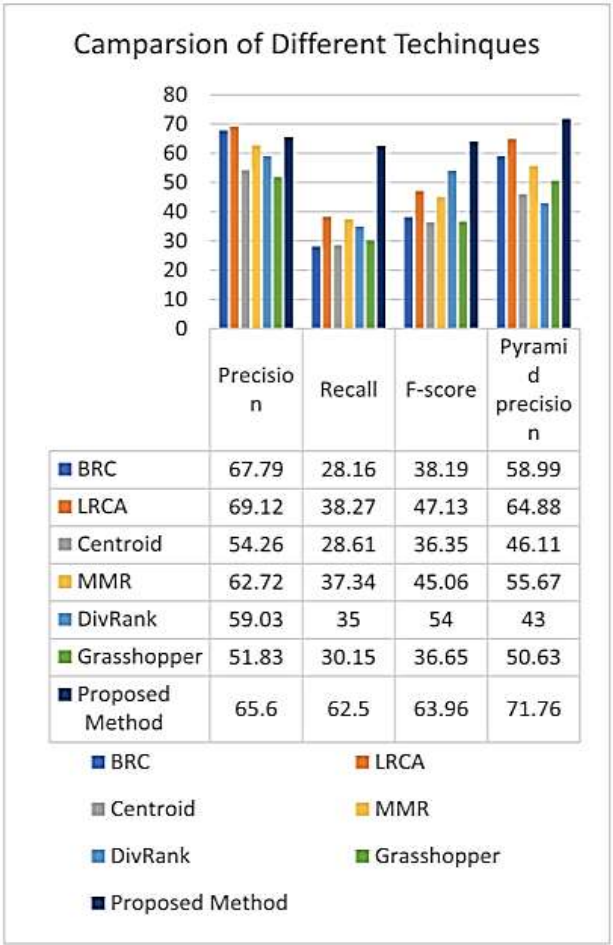
B. Fuzzy C-Means

Fuzzy C-means extracts sentences. Sentence selection is shown below.

- First, the sentences in the dataset are clustered.
- Clustering can only be used after the optimum number of clusters has been determined. This method employs four distinct techniques: Gap Statistics (GSS), K-means, Within sum of squares (WSS), and Silhouette. The best number of clusters is chosen if it is found to be optimal by at least two of the techniques [37].
- Fuzzy C-means clustering is used to determine the optimal number of clusters. The extent to which a given sentence fits into each category can be measured.
- Sentences from each cluster are culled to provide a summary; those with a higher degree of membership

(DOM) or proximity to the cluster centroids are preferred.

- A summary that does justice to the source material and its context will use at least two and no more than four sentences from each cluster.



	Precision	Recall	F-score	Pyramid precision
BRC	67.79	28.16	38.19	58.99
LRCA	69.12	38.27	47.13	64.88
Centroid	54.26	28.61	36.35	46.11
MMR	62.72	37.34	45.06	55.67
DivRank	59.03	35	54	43
Grasshopper	51.83	30.15	36.65	50.63
Proposed Method	65.6	62.5	63.96	71.76

Figure 2: Analyzing and contrasting several algorithms on the BRC corpus

C. Rule Engine

A well-designed set of rules is essential for producing a summary that is both exhaustive and comprehensible. All rules were manually created by domain experts. All rules prioritise features. Notations include:

- Degree of Membership – D.O.M
- Threshold Value – □
- Length of a sentence – L(Si)

Here are the parameters that were set up:

- In the event that the D.O.M. is greater than for each cluster,
- The condition is met if the keyword score is lower than 6.
- For two or more keywords, IF (Keyword(score)1) THEN
- If (function() exists) and (bugId exists), then
- If (Keyword(score) 1) AND (L(Si))

These rules determine sentence selection for an extractive summary. Summary sentences are taken from the source document. Bug report compression is 20%.

D. Hierarchal Clustering

The bug report extraction summary is processed to eliminate repetition and re-rank sentences. Hierarchical clustering removes redundancy and re-ranks the summary. Dendograms show clusters in average connecting. Dendograms show phrase hierarchies as tree-like graphs. In a tree-like structure, sentence height indicates relevance. Top sentences are the shorter ones because they are more important. Selecting one sentence among similar-height sentences eliminates repetition.

Thus, hierarchical clustering rearranges summary sentences. Redundant summary sentences were eliminated. Reranking sentences shows that the original document's less important sentences are more relevant. Hierarchical grouping produces a clearer, more relevant summary. This brief summary will assist developers and contributors understand the original problem report.

Section V illustrates the recommended methodology using Apache project HDFS-13112 bug report (E).

This is the initial step to selecting sentences for optimal clusters. Four methods—GSS, K-means, WSS, and silhouette—compute clusters. Fig 6 shows the graphs.

Table 1: Score computation of various content words

Content words	Edit	Log	Corrup-tion	delayed	Block	Remov-al	Fix
Deg(CW)	12	12	6	3	3	3	2
Freq(CW)	5	5	2	1	1	1	2
Ratio (Deg/freq)	2.4	2.4	3	3	3	3	1
Content words	Recur-sively	Op_close	Corrup-ting	Filey	Restart-ing	Dirx	fail
Deg(CW)	2	1	1	1	1	2	1
Freq(CW)	1	1	1	1	1	2	1
Ratio (Deg/freq)	2	1	1	1	1	1	1

VI. RESULTS AND ANALYSIS

RQ1 compares the suggested bug report summarising methodology to supervised and unsupervised methods. Table and bar graphs in fig 2 show the experimental outcomes for four evaluation measures.

The novel strategy routinely outperforms comparable methods, as seen above. Thus, it may summarise bug reports better, as seen in figure 3 to 6.

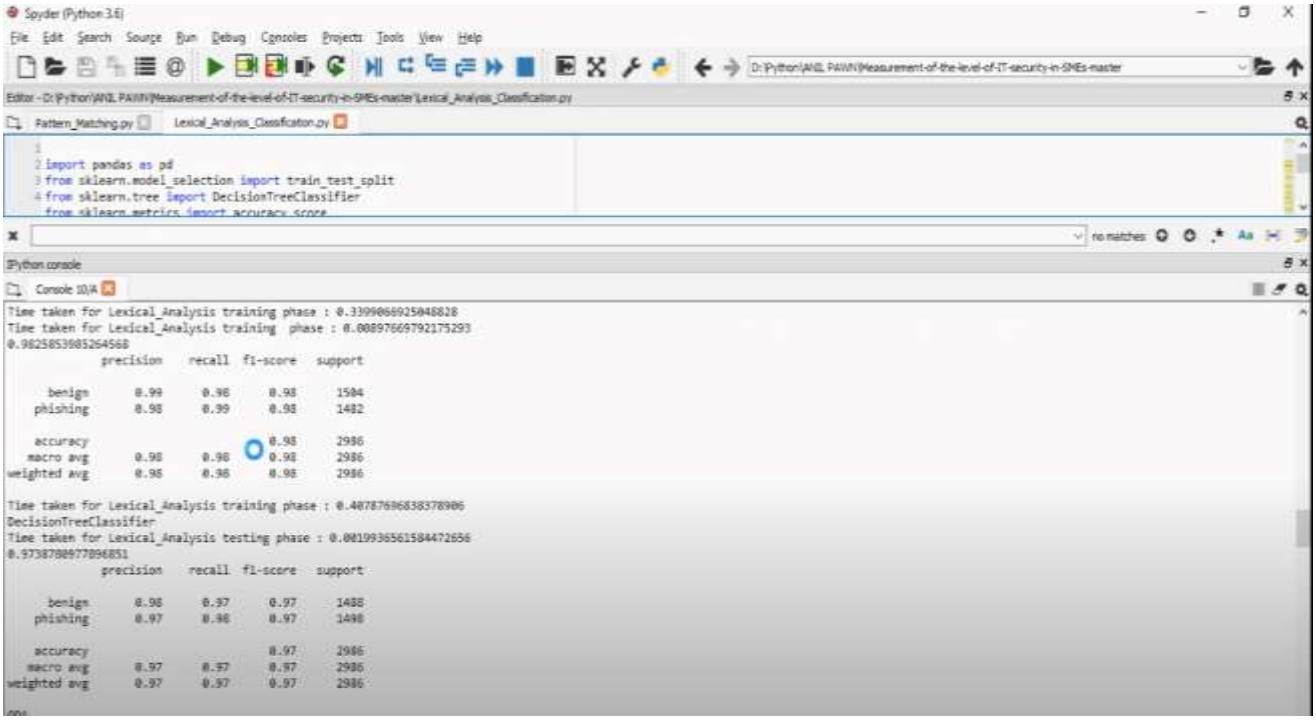


Figure 3: Accuracy and time taken for training

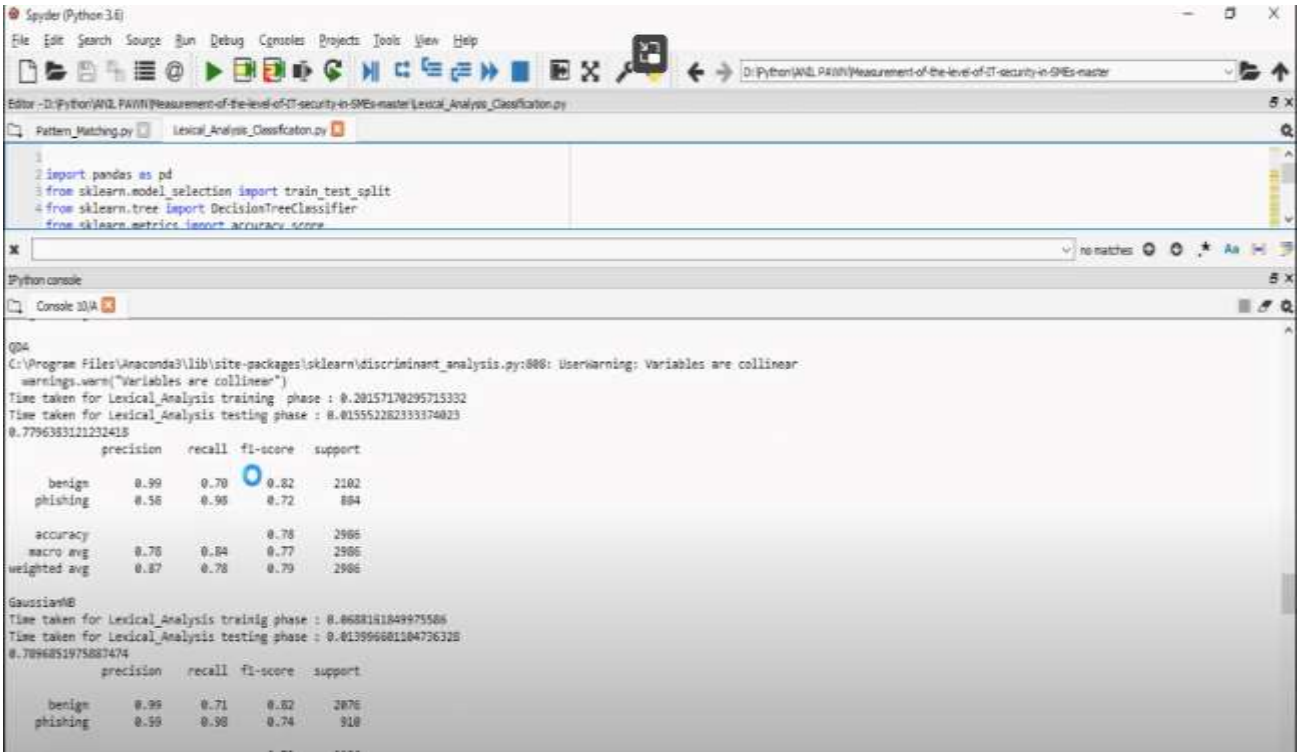


Figure 4: Accuracy

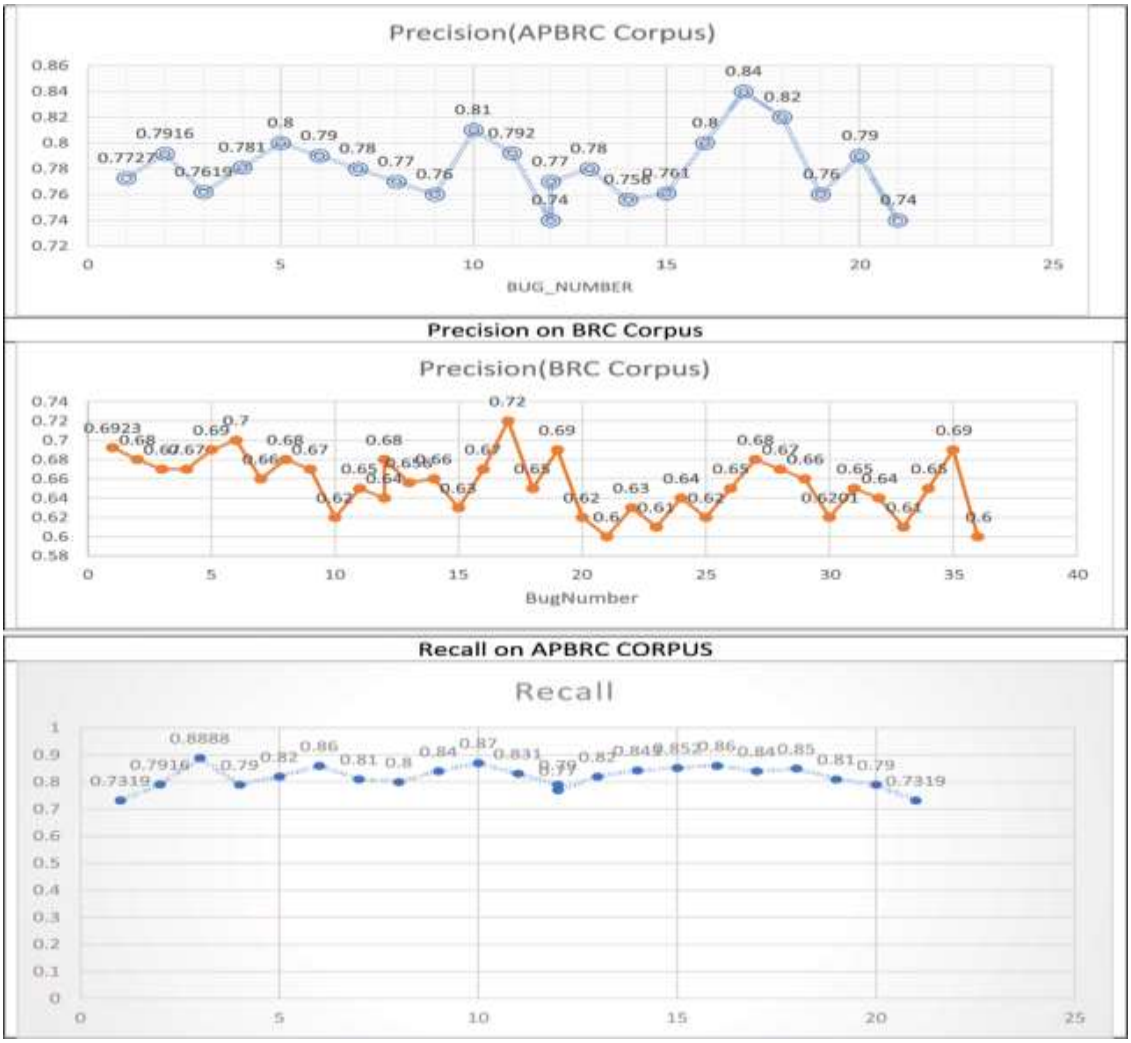


Figure 5: Precision, Recall, F-Score, and Pyramid Precision on The Apbrc and Brc Corpora, among other Metrics

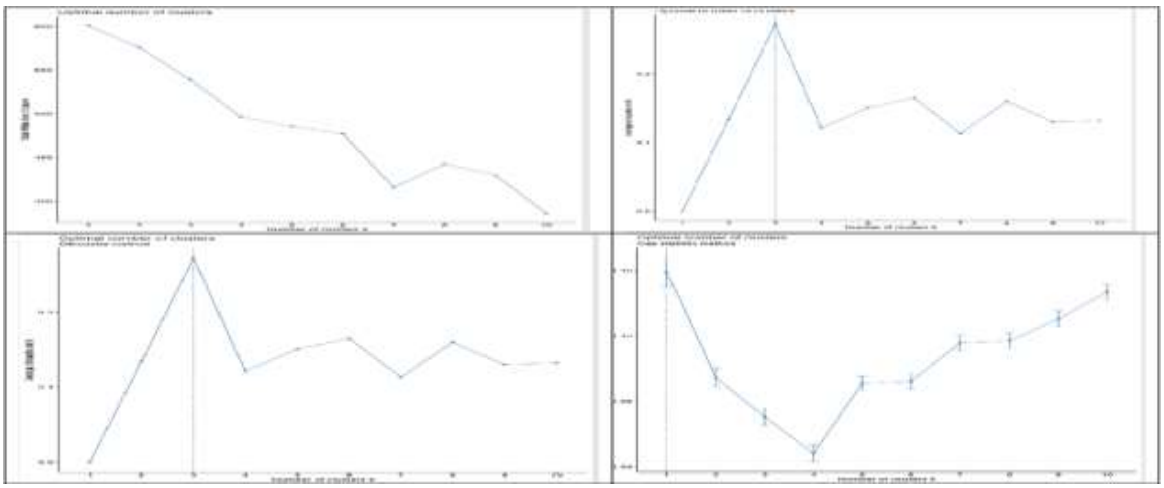


Figure 6: Computation of Optimum Number of Clusters

VII. CONCLUSION

In this work, we introduce an unsupervised method for automatically summarising software bug reports using keywords and full sentences. Two feature extraction strategies are employed to overcome literature's corpus-oriented and document-oriented drawbacks: Quickly and automatically extract relevant keywords by inverting the frequency with which they appear in documents. RAKE is unsupervised and works across languages and domains. Sentence clusters from bug reports are extracted. Optimal clustering is determined by K-means, GSS, Silhouette, and WSS. Words with high membership value are chosen from each cluster using fuzzy c-means clustering to handle ambiguous information in bug reports. Rule-based method combines keyword and sentence elements into an unified summary. The resulting summary may contain a single instructive sentence or several sentences with comparable

meaning. Hierarchical clustering reduces redundancy and re-ranks summary phrases. The bug report is compressed 20%. It summarises bug reports. Due to the limited availability of training data outside of the BRC, an unsupervised learning approach is necessary to generate useful summaries from any bug report corpus. Fuzzy c-means is used to make predictions with less room for error and more information. The problem report is parsed by fuzzy C-means clustering to find the sentences that best describe each cluster.

The proposed unsupervised method can be tried out on any dataset to generate a short but complete summary without the use of training data. The method is tested on the APBRC and BRC corpora. The proposed method is contrasted with both supervised (BRC and LRCA) and unsupervised (MMR, Centroid, DivRank, Grasshopper, Hurried) alternatives. Recall, F-score, pyramid precision, and precision are all improved by 34.3%, 25.77%, 12.77%, 24.23 %, 16.83 %, and 6.88%, respectively, compared to the results obtained by using BRC and LRCA. The F-score indicates a productivity increase of 25.77% and a growth of 16.83% showon in Fig 3 and fig 4. Methods that don't require human intervention have been enhanced. The mean values of precision, recall, f-score, and pyramid precision for the APBRC corpus are 78.22 percent, 82.1 eight percent, 80. one percent, and 81.6 six percent, respectively. As a bug summarying method, automation is superior.

Future work will involve evaluating different clustering algorithms. There are a variety of metrics that can be used to evaluate the success of the proposed approach. In the field of literature, an evaluation method known as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is used to assess works based on their recall. Inaccurate bug report evaluations have been made. It could be used in upcoming bug reports.

REFERENCES

[1] K. Zechner, "Automatic summarization of open-domain multiparty dialogues in diverse genres", Comput. Linguistics, vol. 28, pp. 447-485, Dec. 2002.

[2] L. Zhou, E. Hovy and M. Rey, "A Web-trained extraction summarization system", Proc. HLT-NAACL Conf., pp. 205-211, May 2003.

[3] X. Zhu and G. Penn, "Summarization of spontaneous conversations", Proc. 9th Int. Conf. Spoken Lang. Process., pp. 1531-1534, 2006.

[4] G. Murray and G. Carenini, "Summarizing spoken and written conversations", Proc. Conf. Empirical Methods Natural Lang. Process. EMNLP, pp. 773-782, Oct. 2008.

[5] O. Rambow and J. Chen, "Summarizing email threads", 2004.

[6] S. Wan and K. McKeown, "Generating overview summaries of ongoing email thread discussions", Proc. 20th Int. Conf. Comput. Linguistics COLING, pp. 549, 2004.

[7] X. Xia, D. Lo, E. Shihab and X. Wang, "Automated bug report field reassignment and refinement prediction", IEEE Trans. Rel., vol. 65, no. 3, pp. 1094-1113, Sep. 2016.

[8] E. Hassan and T. Xie, "Software intelligence: The future of mining software engineering data", Proc. FSE/SDP workshop Future Softw. Eng. Res. FoSER, pp. 161-165, 2010.

[9] T. Xie, S. Thummalapenta, D. Lo and C. Liu, "Data mining for software engineering", Computer, vol. 42, no. 8, pp. 55-62, Aug. 2009.

[10] T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling", Proc. 27th IEEE/ACM Int. Conf. Automated Softw. Eng. ASE, pp. 70-79, 2012.

[11] Sun, D. Lo, X. Wang, J. Jiang and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval", Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. ICSE, pp. 45-54, 2010.

[12] H. Mei and L. Zhang, "Can big data bring a breakthrough for software automation?", Sci. China Inf. Sci., vol. 61, no. 5, pp. 1-3, May 2018.

[13] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen and X. Wang, "Improving automated bug triaging with specialized topic model", IEEE Trans. Softw. Eng., vol. 43, no. 3, pp. 272-297, Mar. 2017.

[14] T. Zhang, G. Yang, B. Lee and E. K. Lua, "A novel developer ranking algorithm for automatic bug triage using topic model and developer relations", Proc. 21st Asia–Pacific Softw. Eng. Conf., pp. 246-253, Dec. 2014.

[15] J. Xuan, H. Jiang, H. Zhang and Z. Ren, "Developer recommendation on bug commenting: A ranking approach for the developer crowd", Sci. China Inf. Sci., vol. 60, Jul. 2017.

[16] M. Rush, S. Chopra and J. Weston, "A neural attention model for abstractive sentence summarization", Proc. Conf. Empirical Methods Natural Lang. Process., pp. 379-389, 2015.

[17] S. Chopra, M. Auli and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks", Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol., pp. 93-98, 2016.

[18] M. Mohd, R. Jan and M. Shah, "Text document summarization using word embedding", Expert Syst. Appl., vol. 143, Apr. 2020.

[19] S. Rastkar, G. C. Murphy and G. Murray, "Automatic summarization of bug reports", IEEE Trans. Softw. Eng., vol. 40, no. 4, pp. 366-380, Apr. 2014.

[20] 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. Knowl. Eng., vol. 27, no. 6, pp. 869-896, Aug. 2017.

[21] H. Jiang, X. Li, Z. Ren, J. Xuan and Z. Jin, "Toward better summarizing bug reports with crowdsourcing elicited attributes", IEEE Trans. Rel., vol. 68, no. 1, pp. 2-22, Mar. 2019.

[22] S. Mani, R. Catherine, V. S. Sinha and A. Dubey, "AUSUM?: Approach for unsupervised bug report summarization", Proc. ACM SIGSOFT 20th Int. Symp. Found. Softw. Eng., pp. 1-11, Nov. 2012.

[23] R. Lotufo, Z. Malik and K. Czarnecki, "Modelling the 'hurried' bug report reading process to summarize bug reports", Empirical Softw. Eng., vol. 20, no. 2, pp. 516-548, Apr. 2015.

[24] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval", J. Document., vol. 28, no. 1, pp. 11-21, Jan. 1972.

[25] G. Salton, A. Wong and C. S. Yang, "A vector space model for automatic indexing", Commun. ACM, vol. 18, no. 11, pp. 613-620, Nov. 1975.

[26] D. Engel, "Mining for Emerging Technologies Within Text Streams and Documents", Proc. Int. Conf. Data Mining. Soc. Ind. Appl. Math., pp. 1-18, Feb. 2009.

[27] S. Rose, D. Engel, N. Cramer and W. Cowley, "CO RI automatic keyword extraction", Text Mining Appl. Theory, vol. 1, pp. 1-20, 2010.

[28] D. Patel, S. Shah and H. Chhinkaniwala, "Fuzzy logic based multi document summarization with improved sentence scoring and redundancy removal technique", Expert Syst. Appl., vol. 134, pp. 167-177, Nov. 2019.

[29] F. B. Goularte, S. M. Nassar, R. Fileto and H. Saggion, "A text summarization method based on fuzzy rules and

applicable to automated assessment", Expert Syst. Appl., vol. 115, pp. 264-275, Jan. 2019.
[30] Kaur and S. G. Jindal, "Bug report collection system (BRCS)", Proc. 7th Int. Conf. Cloud Comput. Data Sci. Eng. Confluence, pp. 697-701, Jan. 2017.
[31] S. Rose, D. Engel and N. Cramer, "Automatic keyword extraction from individual documents", Text Mining Appl. Theory, vol. 1, pp. 1-20, Mar. 2010.
[32] F. Lobo, "Fuzzy c-means algorithm Fuzzy c-means algorithm".
[33] S. K. Lakshmanaprabu, K. Shankar, D. Gupta, A. Khanna, J. J. P. C. Rodrigues, P. R. Pinheiro, et al., "Ranking analysis for online customer reviews of products using opinion mining with clustering", Complexity, vol. 2018, pp. 1-9, Sep. 2018.
[34] A. Karami, A. Gangopadhyay, B. Zhou and H. Kharrazi, "Fuzzy approach topic discovery in health and medical corpora", Int. J. Fuzzy Syst., vol. 20, no. 4, pp. 1334-1345, Apr. 2018.
[35] N. Statistical, S. Ncss and A. R. Reserved, "Fuzzy clustering".
[36] N. Statistical, S. Ncss and A. R. Reserved, Hierarchical clustering /dendrograms.
[37] C. Malika, N. Ghazzali, V. Boiteau and A. Niknafs, "NbClust: An R package for determining the relevant number of clusters in a data Set", . Stat. Softw., vol. 61, no. 6, pp. 1-36, 2014.
[38] V. K. Gupta and T. J. Siddiqui, "Multi-document summarization using sentence clustering", Proc. 4th Int. Conf. Intell. Human Comput. Interact. (IHCI), pp. 1-5, Dec. 2012.