Complete Bug Report Summarization using Task-Based Evaluation: A Survey

[1]Miss. R. K.Taware, [2]Prof. S. A. Shinde

[1]ME II Computer,VPCOE,Baramati. Pune University(MH),India.

E-mail: rututaware11@gmail .com

[2]Assistant Professor, VPCOE, Baramati. Pune University(MH), India.

E-mail. Meetsan_shinde@yahoo.com

Abstract— Automatic text summarization is based on numerical, linguistical and empirical methods where the summarization system calculates how often certain key words presents. The key words belong to the so named as open class words. The summarization system computes the frequency of the key words in the text, which sentences they are existing in, and where these sentences are in the text. In other words, summaries save time in our daily work. To write a summary of a text is a non-trivial process where one, on one hand has to extract the most central information from the original text, and on the other has to consider the reader of the text and her previous knowledge and possible special interests. Automatic Text Summarization is a technique where a computer summarizes a text. A text is given to the computer and the computer returns a shorter less redundant extract of the original text. So far automatic text summarization has not yet reached the quality possible with manual summarization, where a human interprets the text and writes a completely new shorter text with new lexical and syntactic choices. However, automatic text summarization is untiring, consistent and always available. Evaluating summaries and automatic text summarization systems is not a straightforward process.

Generally speaking there is one can also perform task-based evaluations where one tries to discern to what degree the resulting summaries are beneficent for the completion of a specific task. It focuses on different aspects of creating an environment for evaluating information extraction systems, with a center of interest in automatic text summarization.

Keywords— Empirical Software Engineering, Summarization of Software Artifacts, Bug Report Duplicate Detection, Extractive System , Abstractive System , Text Summarization ,Email threats ,state- of – threat system.

INTRODUCTION

Those from outside the profession of software development at times mistakenly trust that the profession is all about programming. Those involved in software development know that the profession has a strong factor of information management. Any successful large and complex software system needs the creation and management of many artifacts: requirements, designs, bug reports, and source code with surrounded documentation to name just a few. To perform work on the system, a software developer must often read and understand artifacts related with the system development. For example a developer trying to fix a performance bug on a system may be told that a similar bug was solved before several days.

Finding the bug report that seized the knowledge about what was fixed will likely involve the developer to perform examines and read several bug reports in search of the report of interest. Each report read may cover several sentences of description as well as tens of sentences representing discussion amongst team members. Sometimes, the amount of information may be overwhelming, causing searches to be out of control and duplicate or non-optimized work to be performed, all because the previous history of the project has been unnoticed. One way to reduce the time a developer spends getting to the right artifacts to perform their work is to provide a summary of each artifact.

Generally, there are two approaches to automatic summarization: extraction and abstraction.
In this approach the possibility of automatic summary generation, focusing on one kind of project artifact, bug reports, to make the investigation manageable and to focus on these reports as there are a number of cases in which developers may make use of existing bug reports, such as when triaging bugs or when performing change tasks and these reports can often be lengthy, involving discussions between multiple team membersYou can copy and past here and format accordingly to the default front. It will be easy and time consuming for you.

Need OF Automatic Summarization

A. Key word extraction:

a. Task description **:** The task description is the method in which by taking a part of text, such as some textual part of any document , and create a list of keywords, but most text shows absence of pre-existing key words. In that extractor might select key words from th document. These are pulled directly from the text. As compare to an abstractive system, it somehow internalize the content and generate key words that might be more descriptive and more like what a human would produce. Key words have many applications, such as to improve document browsing by providing a short summary.

b. Key word extraction as supervised learning :There are known key words available for a set of training documents. Using the known key words, assign positive or negative tags to the part of text. Then train a classifier that can distinguish among positive and negative values like features..After training a classifier , select keywords for text documents.

c. Unsupervised key word extraction (TextRank) :In this, consider key words available for a set of training documents. Using the known key words, assign positive or negative values to the given part of text. Then train a classifier that can distinguish amongst positive and negative values. After that, select key words for text documents. In this method probabilities are given,so a threshold value is used to select the key words. Key words extractors are generally weighed using precision and recall. Example is TextRank method.

B. Document summarization

Like keyword extraction, document summarization hopes to identify the core of a text. Summarization systems are typically estimated using summarization approaches. The most common way is using the so-called ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure. This is a recall-based measure that concludes how well a system-generated summary covers the content present in one or more human-generated model summaries known as references.

a. Supervised learning approaches: This technique is like supervised key words extraction. Basically, if we have a collection of documents and human-generated summaries for them, we can learn features of sentences that make them good candidates for inclusion in the summary. Features may include the position in the document and the number of words in the sentence, etc. Disadvantage of supervised extractive summarization is that the known summaries must be manually created by extracting sentences so the sentences in an original training document can be labeled as in summary or not in summary.

b. Unsupervised approaches :TextRank and LexRank :This approach is quite similar to unsupervised key words extraction. It gets around the issue of costly training data. Certain unsupervised summarization methods are based on finding a centroid sentence, which is the mean word vector of all the sentences in the text. LexRank is an algorithm basically identical to TextRank, and both use this approach for document summarization.It is used for key words extraction or any other NLP methods.

C. Multi-document summarization method

It is an automatic technique directed at extraction of information from multiple characters written about the same topic. And summary report permits individual users, such as professional’s data, to quickly make clear themselves with information kept in check in a large cluster of documents.
This summarization makes information reports that are both concise and comprehensive. The aim of a brief summary is to simplify information search and cut the time by pointing to the most relevant source documents. So there is limitation to the need for accessing original files

METHOD 1 : EXTRACTION AND ABSTRACTION APPROACH

Nenkova and K. McKeown,2011[2]: Automatic Text Summarization Technique is divided into two categories as follows:

1) Extraction technique:
Extraction technique is simply extracting important sentences into final summary, where importance of sentence is calculated based on weights assigned to sentences using statistical and linguistic features of text. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. The main challenge is to identify important parts of document and extract them for final summary. Here most work presented on single-document summarization using extraction method.

2) Abstraction technique :
In contrast, abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might generate. Such a summary might contain words not explicitly present in the original. Abstraction technique involves semantic based text summarization. Abstraction technique uses purely linguistic feature, which are very hard to calculate.

Advantages :In order to make summary more reliable, accurate, complete and less redundant following process is applied:

· Final weight of every sentence using weight-ranking equation is computed.
· Final weights are sorted in reverse order.
· Top weighted sentences are selected for summary according to compression ratio required.

Disadvantages :There are a lot of challenges which needs to be resolved such as: -

· A corpus for different language's stop words is not available.
· Features for different language are diverse and are very difficult to process.
· Pronoun level ambiguity is very difficult to remove.

METHOD 2: SUMMARIZING SPOKEN AND WRITTEN CONVERSATIONS

This method G. Murray and G. Carenini, ,2008[3] refer to research on summarizing conversations in the meetings and emails domains. It present a conversation summarization system that works in multiple domains utilizing general conversational features, and compare our results with domain-dependent systems for meeting and email data.In this method uses an extractive approach to summarization, presenting a novel set of conversational features for locating the most significant sentences in meeting speech and emails. So this approach demonstrate that using these conversational features in a machine-learning sentence. classification framework yields performance that is competitive or superior to more restricted domain-specific systems, while having the advantage of being portable across conversational modalities. This robust the performance of the conversation-based system is attested via several summarization evaluation techniques, and this give an in-depth analysis of the effectiveness of the individual features and feature subclasses used.

Conversation Summarization System

Conversation summarization approach, it treat emails and meetings as conversations comprised of turns between multiple participants. This method follow Carenini et al. (2007) in working at the finer granularity of email fragments, so that for an email thread, a turn consists of a single email fragment in the exchange. The features it derive for summarization are based on this view of the conversational structure.

- Length features - For each sentence, it derive a word-count feature normalized by the longest sentence in the conversation (SLEN) and a word-count feature normalized by the longest sentence in the turn (SLEN2).
- Structural features - Including position of the sentence in the turn (TLOC) and position of the sentence in the conversation (CLOC). It also include the time from the beginning of the conversation to the current turn (TPOS1) and from the current turn to the end of the conversation (TPOS2).
- Pause-style features: Include the time between the following turn and the current turn (SPAU), and the time between the current turn and previous turn (PPAU), both normalized by the overall length of the conversation. These features are based on the email and meeting transcript timestamps.
- Conversation participants features : Include two types of features. One measures how dominant the current participant is in terms of words in the conversation (DOM), and the other is a binary feature indicating whether the current participant initiated the conversation (BEGAUTH), based simply on whether they were the first contributor.
- Several lexical features : Used in these experiments. For each unique word, determine two conditional probabilities. For each conversation participant, system compute the probability of the participant given the word, estimating the probability from the actual term counts, and take the maximum of these conditional probabilities as the first term score, which will call as Sprob.

Comparison Summarization Systems : In order to compare the Conversation Summary system with state-of-the-art systems for meeting and email summarization, respectively, this method also present results using the features described by Murray and Renals (2008) for meetings and the features described by Rambow (2004) for email. Because the work by Murray and Renals used the same dataset. However, Rambow carried out summarization work on a different, unavailable email corpus, and so in this method re-implemented their summarization system for current email data.

Advantages:

- Here the conversation feature set that is similarly effective in both the meetings and emails domains.
- A general conversation summarization approach can achieve results on par with state-of-the-art systems that rely on features specific to more focused domains.
- A general conversation summarization system is valuable in that it may save time and effort required to implement unique systems in a variety of conversational domains.

Disadvantages :

- It is only for particular domain so by extending this system to other conversation domains such as chats, blogs and telephone speech, it will give better results.

METHOD 3: COPING WITH AN OPEN BUG REPOSITORY

J. Anvik, L. Hiew, and G. C. Murphy,2005 provides the method of an initial characterization of two open bug repositories from the Eclipse and Firefox projects, describe the duplicate bug and bug triage problems that arise with these open bug repositories, and discuss how applying machine learning technology to help automate these processes. This method present data to fill this gap, providing a characterization of the data in and the use of parts of the bug repositories for two open source projects: Eclipse (V3.0) and Firefox (V1.0). This data confirms two problems that arise with open bug repositories that open source developers have communicated to us previously . The difficulty of detecting which bug reports are duplicates of those already in the repository and the difficulty of assigning new bug reports to the appropriate developer. At this time, the approaches taken to these problems are human-oriented; humans must read the bugs and decide upon whether they are duplicates, and to whom they should be assigned. So these processes can, at least in part, be automated by using the historical information about the bug processes stored in the bug repository. The data and ideas presented in this method provide a basis for considering the kind of support that should be integrated into a development environment to support better bug reporting and tracking.

Examples of heuristics that used by systems are as follows:

- If a report is resolved as FIXED, it was fixed by whoever submitted the last patch that was approved. (Firefox)
- If a report is resolved as FIXED, it was fixed by whoever marked the report as resolved. (Eclipse)
- If a report is resolved as a DUPLICATE, it was resolved by whoever resolved the report of which this is a duplicate of. (Eclipse and Firefox)
- If a report is resolved as WORKSFORME, it was marked as such by the person doing the bug assignment, so it is unclear who the developer would have been. The the report is labeled as unclassifiable. (Firefox) Similar to the analysis we performed for who submits bugs , we determine the top five domains of developers who resolved bugs.
- Advantages :
- This duplicate detection uses a statistical model built from the knowledge of past reports using machine learning techniques.
- An incremental approach allows to detect duplicates of new bug reports and adapt to the changing composition of bugs in the repository.
- By using cosine similarity [9], the model classifies new bug reports as either being unique or duplicate.
-

METHOD 4: SUMMARIZING EMAIL THREADS

O. Rambow, L. Shrestha, J. Chen, and C. Lauridsen,introduced summarizing email threads, i.e., coherent exchanges of email messages among several participants. Summarizing threads of email is different from summarizing other types of written communication as it has an inherent dialog structure. So in this method initial research which shows that sentence extraction techniques can work for email threads as well, but profit from email-specific features. In addition, the presentation of the summary should take into account the dialogic structure of email communication. This system shows Email is a written medium of asynchronous multi-party communication. This means that, unlike for example news stories but as in face-to-face spoken dialog, the email thread as a whole is a collaborative effort with interaction among the discourse participants.

However, unlike spoken dialog, the discourse participants are not physically co-present, so that the written word is the only channel of communication. But some time replies do not happen immediately, so that responses need to take special precautions to identify relevant elements of the discourse context . Thus, email is a distinct linguistic genre that poses its own challenges to summarization. In the approach the paradigm used for other genres of summarization, namely sentence extraction: important sentences are extracted from the thread and are composed into a summary. Given the special characteristics of email, here predict that certain email-specific features can help in identifying relevant sentences for extraction. In addition, in presenting the extracted summary, special "wrappers" ensure that the reader can reconstruct the interactional aspect of the thread, which assume the part is crucial for understanding the su

Disadvantages:

- Need to perform a qualitative error analysis and investigate in more detail .
- Need to improve the automatic extraction. mmary. We acknowledge that other techniques should also be explored for email summarization, but leave that to separate work

METHOD 5 : IDENTIFYING DUPLICATE DEFECT REPORTS WITH NATURAL LANGUAGE PROCESSING

Various processes that are currently being used in software engineering require or produce reports that are written in natural language. Defect reports that are generated by a number of testing and development activities is such an object that is structured in natural language, which makes it very hard to compare two reports for similarity. In this document, a method that makes use of Natural Language Processing (NLP) techniques to identify duplicate defect reports will be explained. A situational example that shows the different steps within the method is given and related literature is also described.

When complex software products are being developed, it is very common that defects slip into the product. These defects can lead to software failures or unexpected behaviour. Testing procedures will likely find these defects and report them in so called defect reports that are put in a defect management system. If the same architecture is being used in different products or development is parallel, the same defects may be reported by testing activities of different products. The same error might be reported several times in the defect management system. The method that will be described can be used to identify these duplicates in the system. It is largely based on Natural Language Processing (NLP).

There are five stages in the NLP (Manning and Schütze, 1999):

- Tokenization
- Stemming
- Stop words removal
- Vector space representation
- Similarity calculation

ACKNOWLEDGMENT

Thanks to prof. Santosh. A. Shinde, for useful discussion on summarization and for his great efforts of supervising and leading me to accomplish this satisfactory work. To college and department staff , they were a great source of support and encouragement. To my friends and family, for their warm wishes and loves. Thanks to every person who gave something too light my pathway.

CONCLUSION

In this review paper ,we discussed the three methods for bug report summarization based on text summarization concept i.e. Keyword extraction , Document summarization etc. and their advantages and disadvantages. Performance of extraction based summarization is better than abstractive approach. According to results of different bug report summarization and duplicate detection task. We conclude that there is no single method to generate summaries as well as duplicate detection task of documents. There are several methods are available for these task.We have discussed the need and challenges of document summarization and duplicate detection task and their application. We have tried to present almost all possible techniques of bug report summarization.

REFERENCES:

[1] Sarah Rastkar, Gail C. Murphy and Gabriel Murray, "Automatic Summarization of Bug Reports,"IEEE Transactions on Software Engineering,2013.

[2] Nenkova and K. McKeown, "Automatic summarization," Foundations and Trends in Information Retrieval, vol. 5, no. 2-3, pp. 103–233, 2011.

[3] G. Murray and G. Carenini, "Summarizing spoken and written conversations," in EMNLP'08: Proc. of the 2008 Conference on Empirical Methods on Natural Language Processing, 2008.

[4] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," in Proc. of the 2005 OOPSLA Workshop on Eclipse Technology eXchange, 2005, pp. 35–39.

[5] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in ICSE'06: Proc. of the 28th International Conference on Software Engineering, 2006, pp. 361–370.

[6] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in ICSE'06: Proc. of the 28th International Conference on Software Engineering, 2006, pp. 361–370.

[7] J. Davidson, N. Mohan, and C. Jensen, "Coping with duplicate bug reports in free/open source software projects," in VL/HCC'11: Proc. of the 2011 IEEE Symposium on Visual Languages and Human-Centric Computing, 2011, pp. 101 108.

[8] O. Rambow, L. Shrestha, J. Chen, and C. Lauridsen, "Summarizing email threads," in HLT-NAACL'04: Proc. of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2004.

[9] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," in ICSE'08: Proc. of the 30th International Conference on Software Engineering, 2008, pp. 461–470

[10] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in ICSE'07: Proc. Of the 29th International Conference on Software Engineering, 2007, pp. 499–510