Characterizing Common and Domain-Specific Package Bugs: A Case Study on Ubuntu

Xiaoxue Ren*, Qiao Huang*, Xin Xia†, Zhenchang Xing‡, Lingfeng Bao*, and David Lo§
*College of Computer Science and Technology, Zhejiang University, China
†Faculty of Information Technology, Monash University, Australia
‡College of Engineering and Computer Science, Australian National University, Australia
§School of Information Systems, Singapore Management University, Singapore
{xxren, tkdsheep, lingfengbao}@zju.edu.cn, xin.xia@monash.edu, Zhenchang.Xing@anu.edu.au, davidlo@smu.edu.sg

***Abstract*—Ubuntu is an open source software platform that runs everywhere from the smartphone, the tablet and the PC to the server and the cloud. In Ubuntu, there are many self-contained or third-party software packages for different use, and a bug report in Ubuntu could affect one or more packages simultaneously. Identifying the common package bugs in Ubuntu can help both developers and users better understand the packages they are developing or using, and also provide further guidelines to developers of similar packages in the future. In this paper, we perform a large-scale empirical study of common package bugs on Ubuntu by leveraging topic modeling. By analyzing a total of 240,097 bug reports, we identify 3 general bugs that are common to all Ubuntu packages, i.e., Graphical User Interface (GUI), Maintenance, and Runtime bugs. Moreover, we categorize top-100 packages with most number of bug reports into 6 categories (i.e., graphics, internet, office, sound and video, system management, and kernel), and identify domain-specific bugs for each category.**

***Index Terms*—Empirical Study, Bug Report Summarization, Topic Model**

I. Introduction

Due to the complexity of software development, bugs are inevitable. Bug fixing is one of the most important activities in the whole life cycle of software development and maintenance. In this case, understanding bugs in a software system could help developers and users understand the system further, and also provide further guidelines to developers of similar systems in the future [1].

Ubuntu, one of the world's most popular Linux distributions for desktop and laptop PCs, is a large project with a huge number of bug reports. Although Ubuntu has its own bug tracking system (i.e., Launchpad[1]), it cannot tell developers what are the common problems that are mentioned by a large number of bug reports. We notice that Launchpad allows developers to assign "tags" for each bug report. However, most tags are in a coarse granularity, which only contain limited information (e.g., the mostly used tags are about release version, type of CPU and etc). Thus, in this paper, we are interested to summarize bug reports in Ubuntu and identify common problems for different packages. Identifying these problems through bug reports can help developers better understand the packages they are developing, and also provide further guidelines to developers of similar packages in the future [2]–[5]. Besides, the package users can also be benefited when they need to choose the most suitable package from several comparable packages [6].

Previous studies (e.g., [7]) have proposed techniques to automatically summarize bug reports. However, these work mainly focus on summarizing an extractive abstract of the problem in an individual bug report. Still there lacks of techniques to identify high-level problems that are mentioned by a large number of bug reports [8]. In this paper, we propose a semi-automated approach to summarize bug reports in Ubuntu and identify common problems for different packages. First, we preprocess the textual fields of all bug reports and apply topic model [9] on that whole dataset to extract *n* topics. Topic model is a widely used generative model, which is a modeling method for text implicit topics. Here we treat each topic as a candidate problem. Then for each candidate problem, we manually summarize a high-level abstract by reading bug reports related to this topic. Finally, we manually select typical problems among all candidate problems.

By leveraging topic models, we conduct an empirical study on a total of 240,097 bug reports collected from Ubuntu. First, we manually identify 3 common problems that are common to see in different Ubuntu packages, i.e., Graphical User Interface (GUI), Maintenance, and Runtime. GUI problem mainly concerns the quality of user interface in a package and the user experience of interactions. Maintenance problem mainly relates to package installation, upgrade, and configuration. Runtime problem mainly concerns the availability of a package. Then, we categorize the top-100 packages with most number of bug reports into six categories, and further investigate them to identify some domain-specific problems.

The main contributions of this paper are:

- We perform an empirical study on a total of 240,097 bug reports collected from Ubuntu. We identify 3 common problems that are common to see in different Ubuntu packages, namely: Graphical User Interface (GUI), Maintenance, and Runtime.
- We further investigate six categories that contain the top-100 packages with most number of bug reports and identify domain-specific problems for each category.

[1]https://bugs.launchpad.net/ubuntu

TABLE I
CATEGORIES THAT CONTAINS TOP-100 PACKAGES AND THEIR BRIEF DESCRIPTIONS.

Category Name	Bug Reports Numbers	Description
Graphics	24745	The category of Graphics include the packages that are related to the user interface, such as unity.
Internet	13667	The packages in the category of Internet are mainly used to help users to get resource from network with the used of Internet, such as browsers or some e-mail tools.
Office	5215	It mainly refers to the office tool, including word processing, table processing, slide processing and so on.
Sound and Video	8648	The category of Sound and Video is a vital part of Multi-Media in computer system that deliver sound and animation to users, such as some players.
System Management	43251	Systems management includes the packages that help users to config and manage the computer systems, just like the file and plug-in management tool and so on.
Kernel	16507	Kernel (operating system) refers to the central component of Ubuntu.

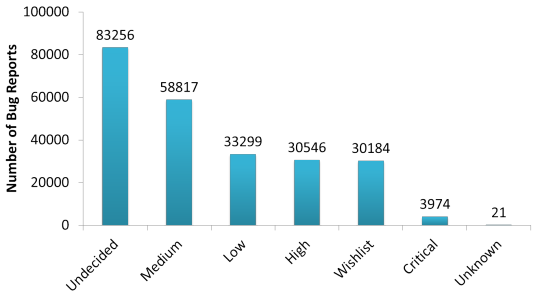

Fig. 1. Number of bug reports with different importance

II. CASE STUDY SETUP

In this section, we describe the details of how we collect the data, and present basic statistics of our dataset. Then we propose our semi-automated common package bug detection approach, and the two research questions of our study.

A. Data Collection

Ubuntu uses Launchpad to report, record, and manage its bugs. We write a web crawler to traverse and store the HTML file of each bug report. For each bug report, we parse the HTML file to extract useful fields, including ID, title, description, affects, status and importance.

Since a bug can simultaneously affect multiple packages with different status and importance. For such bug report, we divide it into multiple bug reports with corresponding packages. In this way, we end up with 240,097 bug reports in total. We also extract the name of each affected packages. Two packages are considered different if they have different names (e.g., *unity* and *unity-2d* are different packages). In total, we extract 16,098 packages with different names.

B. Data Statistics

Calculating the basic statistics of the whole dataset can help us better understand Ubuntu bug reports. We would like to know the number of bug reports with different importance and status. We also want to know the distribution of bug reports among different packages. More specifically, we want to find out the top-100 packages that have the biggest number of bug reports for further investigation.

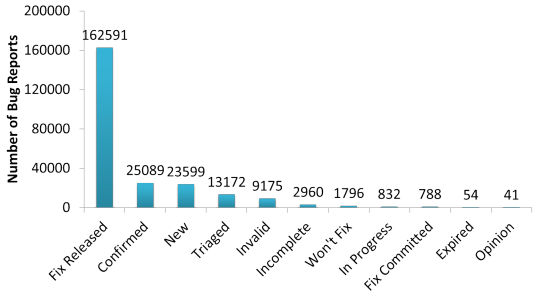

Fig. 2. Number of bug reports with different status

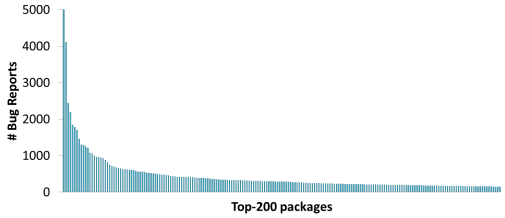

Fig. 3. The distribution of bug reports among top-200 packages

We first calculate the number of bug reports with different importance and status, as shown in Figure 1 and Figure 2, respectively. The number of "undecided" bug reports ranks the first, followed by "medium" bug reports. The number of "critical" bug reports is quite low,about 1.65% of all bug reports. The top-5 status of bug reports are "Fixed Released", "Confirmed", "New", "Triaged" and "Invalid", respectively. Other status of bug reports account for only about 2.7% in total. When summarizing bug reports, we only consider "Fixed Released", "Confirmed" or "Triaged" status.

Then we calculate the number of bug reports for each package. We sort all packages in descending order according to the number of bug reports affecting each package. Figure 3 shows the distribution of bug reports among top-200 packages. Other packages are omitted in Figure 3 due to space limitation. The number of bug reports in top-200 packages (1.24% of all packages) accounted for 38.07% of all bug reports. The distribution is very similar to the long-tail distribution. Additionally, there are 13,625 packages having no more than 10 bug reports. In our experiment, for simplicity reasons (i.e., to reduce manual analysis efforts and execution time), we manually select the top-100 packages with most number of bug reports. Table II presents the name of categories that these top-100 packages belong to[2], and the number of bug reports.

C. Semi-automated Common Package Bugs Detection

[2]The category of these 100 packages can be found in the Ubuntu software-center.

TABLE II
CATEGORIES OF TOP-100 PACKAGES.

Category Name	#Packages
Graphics	unity, compiz, unity8, xorg, ubuntu-ui-toolkit, gnome-panel, kdebase, xorg-server, gdm,gtk+2.0, mir, gnome-terminal, unity-greeter, f-spot, kdebase-workspace, gnome-shell, kdepim, mesa
Internet	firefox, network-manager, evolution, empathy, firefox-3.0, thunderbird, webbrowser-app, samba, pidgin, ubuntuone-client, gwibber, network-manager-applet, chromium-browser, eucalyptus
Office	openoffice.org, libreoffice, evince, ubuntu-docs, gedit
Sound and Video	alsa-driver, rhythmbox, xserver-xorg-video-intel, pulseaudio, totem, banshee, xserver-xorg-video-ati, fglrx-installer, vlc
System Management	ubiquity, nautilus, update-manager, software-center, gnome-control-center, apport, ubuntu-system-settings, gnome-settings-daemon, gvfs, apt, grub2, gnome-power-manager, synaptic, lightdm, ubuntu-release-upgrader, brasero, debian-installer, udev, gnome-system-tools, apparmor, update-notifier, libvirt, initramfs-tools, acpi-support, software-properties, cups, lxc, file-roller, usb-creator
Kernal	linux, linux-source-2.6.15, linux-source-2.6.20, linux-ti-omap4, linux-source-2.6.22, linux-armadaxp, linux-lts-utopic, linux-mvl-dove (Ubuntu) 532, linux-lts-raring (Ubuntu) 529, linux-lts-trusty (Ubuntu) 526, linux-fsl-imx51 (Ubuntu) 526, linux-ec2 (Ubuntu) 512, linux-mako (Ubuntu) 511, linux-lts-saucy (Ubuntu) 508, linux-lts-quantal (Ubuntu) 502, linux-goldfish (Ubuntu) 502, linux-manta (Ubuntu) 499, linux-lts-vivid, linux-flo, linux-raspi2, linux-lts-backport-maverick, linux-lts-backport-natty, linux-lts-wily, linux-lts-xenial

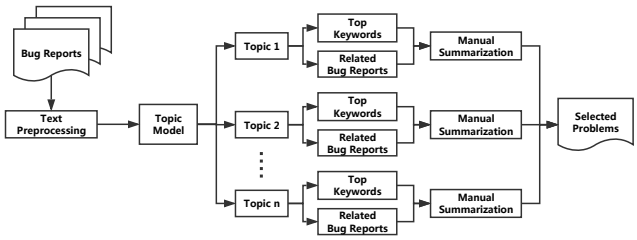

Fig. 4. Overall framework of our approach

TABLE III
AN EXAMPLE OF TOPIC MODEL RESULT

Keywords	crash debug sigsegv happen gnome report startup signal disappear
#21775	nautilus fails to load on gnome-session startup
#49605	nautilus crashes sometime
#27758	nautilus closed while accessing one ftp server
#63350	since last updates of edgy on 1st october, i can't start nautilus anymore
#36225	nautilus crashes
#75669	nautilus crash upon create folder from file menu
#66368	nautilus crashed with no particular reason
#29315	crash on copy operation

1) Overall Framework: Figure 4 presents the overall framework of our approach, which takes a set of bug reports as input. Our framework first preprocesses the title and description to extract the words in each bug report. Then we build topic model on the preprocessed dataset and we end up with *n* topics (i.e., candidate problems). For each topic, we extract the top-k keywords that are most likely to appear in that topic. Next, we manually summarize a short abstract of each topic and eliminate topics that do not contain enough bug reports or useful information for summarization. Finally, we select typical topics to represent significant problems. Notice that some similar topics may be merged to form a higher-level problem. Table III presents an example of topic model result.

D. Text Preprocessing

Topic model requires extracting words appearing in a bug report's title and description. We concatenate title and description to extract words in 3 parts, namely **Tokenization** (only keep tokens that contain English letters), **Stop-word Removal** (use stop word list from Snowball project[3]) and **Stemming** (use Porter stemmer[4])

E. Topic Model

In machine learning and natural language processing, a topic model [10] is a typical statistical model for discovering the hidden semantic structures (i.e., abstract "topics") that occur in a collection of documents. Intuitively, given that a bug report is about a particular problem (i.e., topic), one would expect particular words to appear in the bug report more or less frequently: "connect" and "wifi" will appear more often in bug reports about network problem.

A document (i.e., bug report) typically concerns multiple topics (i.e., problems) in different proportions [11]–[16]. In this paper, we apply Latent Dirichlet allocation (LDA) [9], [17] to extract topics from bug reports.

F. Manual Summarization

Using topic model, we end up with *n* topics (n = 200 in our experiment). We treat each topic as a candidate problem, which needs manual checking. The manual summarization process is performed by the first author and another two graduate students. In this case, we use **card sorting** and the specific steps are as follows: **Opening card sorting**, **Labeling problems** and **Summarizing typical problems**.

G. Research Questions

With the collected dataset, in this paper, we are interested in answering the following 2 research questions:

RQ1: What are the common package bugs?

Extracting and understanding the common package bugs could help developers understand Ubuntu better, and also help developers to avoid these common problems when designing a new package for a Linux operating system [18]. To answer RQ1, we use the semi-automated approach to summarize bug reports and identify the common problems.

[3]http://snowball.tartarus.org/algorithms/english/stop.txt
[4]http://tartarus.org/martin/PorterStemmer/

TABLE IV
COMMON PACKAGE BUGS AND THEIR CORRESPONDING KEYWORDS.

Problem	Keywords
GUI	menu dialog button panel bar scroll cursor view display layout desktop keyboard mouse monitor click drag
Maintenance	install upgrade remove build config version boot patch dpkg apt depend
Runtime	crash hang freeze suspend memory leak cpu slow segment segfault exception

RQ2: What are the domain-specific package bugs for different package categories?

Summarizing domain-specific bugs for packages of each category can help developers avoid these domain-specific bugs when developing a new package. In RQ2, we select top-100 packages in Ubuntu, and the 100 packages are categorized into six categories. Similar to RQ1, we use the semi-automated approach proposed in the previous section to summarize bug reports, and identify the domain-specific problems.

III. CASE STUDY RESULTS

A. *RQ1: What are the common problems?*

After manual summarization, we identify 3 common problems: Graphical User Interface (GUI), Maintenance and Runtime. Since a common problem covers a wide scope, we divide each common problem into several sub-problems for better understanding. For each common problem, the titles of some representative bug reports for different sub-problems below:

*1) **GUI**:* We divide GUI problem into 2 sub-problems:

- ***Unexpected results of interactions***
  - *screen fades out twice when log out dialog is displayed*
- ***Quality of GUI widget***
  - *navigation button in package list view is not visible enough*

From the above examples, we find that these bugs are about interaction and the performance between the combination of widgets (e.g., button, list view, etc.). This kind of problem is closely related to the user experience, and it will directly affect the quality of packages.

*2) **Maintenance**:* We divide maintenance problem into 2 sub-problems:

- ***Problem during maintenance***
  - *oem-config-firstboot runs on every reboot*
- ***Problem after maintenance***
  - *oem-config isn't removed after completion*

From the above examples, we notice that problems may occur during maintenance phase (e.g., package installation, upgrade, configuration, etc.). These problems can make a package unable to work.

*3) **Runtime**:* We divide runtime problem into 3 sub-problems:

- ***Crash problem***
  - *subprocess pre-installation script killed by signal (segmentation fault), core dumped*
- ***System response problem***
  - *response too slow when clicking "provided by ubuntu" subitem of "installed software"*
- ***Performance problem***
  - *xserver sometimes hangs (100% cpu) when closing a window on r500/video-ati*

TABLE V
DISTRIBUTION OF COMMON PACKAGE BUGS OF THE SIX CATEGORIES.

Top-100 Packages Categories	#GUI	#Maintenance	#Runtime	#All
Graphics	9188 (37.13%)	1079 (4.36%)	3113 (12.58%)	13219 (53.42%)
Internet	3258 (23.84%)	731 (5.35%)	3517 (25.73%)	8031 (58.76%)
Office	789 (15.13%)	429 (8.22%)	1474 (28.26%)	2519 (48.31%)
Sound and Video	839 (9.70%)	1286 (14.87%)	2364 (27.34%)	4442 (51.36%)
System Management	6591 (15.24%)	21954 (50.76%)	8373 (19.36%)	26310 (60.83%)
Kernal	5216 (3.16%)	2052 (12.43%)	2841 (17.21%)	4531 (27.45%)

From the above examples, we notice that a package may crash in different situations. Similar to the crash problem, sometimes the package just becomes very slow or even loses response if it is operated in a specific way. Finally, a package may take up too much system resources (e.g., 100% CPU usage, memory leak, etc).

*4) **Common package bugs in the six category of Top-100 packages**:* Table IV shows the keywords we selected for the three types of common package bugs. Table V presents the distribution of common package bugs of the six categories. The last column shows the total number and percentages of bug reports when considering all 3 common package bugs together. Notice that a bug report may assigned to multiple kinds of common package bugs.

For GUI problem, the percentage ranges between 3.16% and 37.13%. Specifically, *Graphics* has the highest percentage of bug reports on GUI problem, while *Kernel* has only 3.16% bug reports on GUI problems. One possible reason is that the packages in the category of Graphics are mainly related to graphical shells. On the other hand, the packages under the category of Kernel are most working in the background. Besides, *Internet* also have more than 20% bug reports belonging to GUI problem, it dues to the fact that in the category, there are also many problems about the UI performance and so on. For maintenance problem, the percentage ranges between 4.36% and 50.76%. *System Management* and *Sound and Video* have relatively high percentage (50.76 % and 14.87%) of bug reports belonging to maintenance problem. The main reason for the high percentage in System Management is that there are many cases of installation and upgrading. In the category of Sound and Video, the problems often belong to the second type of problem in maintenance, namely the so-called problem after maintenance. However, in contrast, the category of Graphics only has 4.36% bug reports about maintenance. One possible reason is that Graphics includes many packages that are the official file manager for the GNOME desktop, which are stable to use. Comparing to GUI and maintenance problem, the runtime problem has a relatively uniform distribution among the top-100 packages. The percentage ranges between 12.58% and 28.26%. By manual investigation, we find that all packages have a substantial number of bug reports about crash and response problems. Thus, we believe that runtime problem is common to see in different packages. Finally, we notice that, in Kernel, only 27.45% bug reports belong to all

3 common problems, which indicates this package may have more domain-specific problems, which are as follows.

B. RQ2: What are the domain-specific package bugs for different package categories?

The top-100 packages are categorized into six categories, i.e., graphics, Internet, office, sound and video, system management, and kernel. In RQ2, we analyze the domain-specific package bugs in these six categories, and we exclude the three types of common package bugs as shown in RQ1.

1) ***Internet****:* we summarized three domain-specific problems, including network problem, message problem and printer problem. These problems mainly happen at the time of using the Internet to get resources through the network.

- ***Network problem*** *(It refers to the problems that often happen when connecting the network, which occupy about 52.65% of all the problems in the category.)*
  - *IPv6 Information confusing with multiple IPv6 addresses*
- ***Message problem*** *(It refers to the problems about the response or warnings when using the Internet which occupy about 15.92% of all the problems in the category.)*
  - *Messages lost when too long*
- ***Printer problem*** *(It refers to the problem that happen when using the printer, which occupy about 6.43% of all the problems in the category.)*
  - *Cannot print from print preview even when print.whileinPrintPreview is set to true*

From the above, it can be seen that the bugs on Printer are happening in some specific browsers, e.g., Firefox.

2) ***Graphics****:* we summarized five domain-specific problems, including File problem, Lock Screen and Workspace problem.

- ***File problem*** *(It refers to the problems about file operations, it takes about 22.43% among all the problems in this category.)*
  - *The "open file" dialog doesn't treat saved searches as folders*
- ***Lock Screen*** *(It refers to the problems that happen when locking screen, which takes about 9.70% among all the problems in this category.)*
  - *Lockscreen capslock detection does'nt work well with remapped capslocks*
- ***Workspace problem*** *(It refers to the UI problems happen in the workspace, which takes about 12.36% among all the problems in this category.)*
  - *Workspaces on Gutsy switch incorrectly using Compiz with left over Feisty settings*

From the domain-specific problems listed above, it can be seen that some of them are closely related to user experience, e.g., the window UI bugs. These kinds of bugs should be paid attention to achieve a better user experience.

3) ***System Management****:* we summarized five domain-specific problems, including Trash, USB, Coding problem, Rating, as well as Users and groups problem.

- ***Trash*** *(It is obvious about problems that happen when managing the trash, which takes about 15.36% among all the problems in the category of System management.)*
  - *nautilus does not ask for confirmation before emptying trash*
- ***USB*** *(It refers to the problems about the management of USB, which takes about 13.97% among all the problem.)*
  - *Cannot see files on external USB drive, and Cannot unmount drive*
- ***Coding problem*** *(It refers to the problems about different types of coding methods, which takes about 18.16% among all the problems.)*
  - *Cannot un-stretch icon in a Nautilus window*

4) ***Kernel****:* it has been concluded three domain-specific problems as follows:

- ***Suspend/resume failure*** *(It refers to the problems about a certain part of the computer fails to suspend or resume, which takes about 14.59% among all the problems.)*
  - *System fails to resume from suspend ONLY when suspended by closing lid*
- ***Disk problem*** *(It refers to the problems about disk, such as read errors and so on, which takes about 10.22% among all the problems.)*
  - *Disk Read Errors during boot-time caused by probe of invalid partitions*
- ***Card work problem*** *(It refers to the problems about card work, such as SD card and SDHC card, which takes about 12.52% among all the problems.)*
  - *Linux doesn't support ENE CB-712 SD card reader*

From the problems listed above, these bugs are mainly caused due to different computer configurations.

5) ***Office****:* apart from the common problems in RQ1, it has been concluded three domain-specific problems.

- ***Language and spelling problem****It means that in Office category, there may be no spelling check problems and problems about different kinds of languages. It takes about 12.47% among all the problems.)*
  - *spell check uses wrong English dictionary*
- ***Font problem*** *(It refers to the problems about the font of words in the office category, which takes about 10.21%.)*
  - *[Upstream] Impress Font fuzzy in presentation mode when Use hardware acceleration enabled*
- ***File problem*** *(It refers to the problem about opening and saving of files in the office category, which takes about 11.83%.)*
  - *[Upstream] I/O Error while opening file from WebDAV over GVFS from Nautilus*

6) ***Sound and Video****:* there are three domain-specific problems, including Radio problem, Playlist problem, Sound card Problem as well. In addition, playlist problem may also belong to the second kind of runtime problem, namely performance problem. While for the sound and video category, playlist problem should be highlight as a domain-specific problem.

- ***Radio problem*** *(It refers to the problems about radio, such as radio server and so on, which takes about 8.25% among all the problems.)*

  - *Broken internet radio playlist handling*
- ***Playlist problem*** *(It refers to the problems about playlist, such as the not updating and so on, which takes about 23.41%.)*
  - *banshee recursive smart playlist does not update*
- ***Sound card Problem*** *(It refers to the bad performance of sound devices, which takes about 18.56%.)*
  - *USB sound card always reports as hw:1 (even if it is the only one)*

C. Threats to Validity

Internal Validity. We perform a semi-automated process to summarize common problems in Ubuntu packages. The process may miss some typical problems that are not identified by manual reading the result of topic model. Besides, the keywords of each common problem are also manually selected, which may miss some important keywords or include keywords that are "too general". To reduce this threat to internal validity, three people perform this manual summarization independently, and any conflict is resolved by three people.

External Validity. Although we summarize the common problems by analyzing more than 200K bug reports, we only use the top-100 packages to further investigate these problems. In the future, we plan to further investigate more Ubuntu packages and apply our approach to other projects.

IV. Related Work

The work conducted by Rastkar et al. [7] is the most similar work to ours. They designed a conversation-based extractive summary generators to produce summaries for bug reports. Their approach focuses on extract important sentences to summarize an individual bug report, while our approach apply topic model to aggregate similar bug reports to identify high-level problems that are mentioned by many bug reports.

There have been a number of studies on software artifacts summarization [19]–[24]. Guerrouj et al. used the context that surrounds code elements in StackOverflow posts to summarize the use and purpose of code elements [19], to help developers to read the code more smoothly. Moreno et al. proposed a technique to automatically generate human readable summaries for Java classes [20]. Gu et al. proposed a pattern-based parsing technique which can parse complex app review sentences to extract aspects and corresponding opinions [21]. However, our work is different from these works, we focus on characteristics of common and domain-specific package bugs in Ubuntu, while the previous studies focus on summarization techniques.

V. Conclusion and Future Work

In this paper, we propose a semi-automated approach to summarize bug reports in Ubuntu and identify common problems for different packages. We conduct an empirical study on a total of 240,097 bug reports collected from Ubuntu. By manually summarization, we identify 3 common problems that are common to see in different Ubuntu packages, namely: Graphical User Interface (GUI), Maintenance, and Runtime. Then, we categorize the top-100 packages with most number of bug reports into six categories, and further investigate them to identify some domain-specific problems for each category. In future work, we plan to further investigate more packages in Ubuntu and apply our approach to other projects.

References

[1] D. Bertram, A. Voida, S. Greenberg, and R. Walker, "Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams," in *CSCW*. ACM, 2010, pp. 291–300.

[2] Q. Huang, D. Lo, X. Xia, Q. Wang, and S. Li, "Which packages would be affected by this bug report?" in *ISSRE*. IEEE, 2017, pp. 124–135.

[3] G. Yang, T. Zhang, and B. Lee, "Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports," in *COMPSAC*, 2014, pp. 97–106.

[4] 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," in *APSEC*, 2014, pp. 223–230.

[5] X. Sun, H. Yang, X. Xia, and B. Li, "Enhancing developer recommendation with supplementary information via mining historical commits," *JSS*, vol. 134, pp. 355–368, 2017.

[6] N. K. Nagwani and S. Verma, "Generating intelligent summary terms for improving knowledge discovery in software bug repositories," in *SEKE*, 2016, pp. 827–844.

[7] S. Rastkar, G. C. Murphy, and G. Murray, "Automatic summarization of bug reports," *TSE*, vol. 40, no. 4, pp. 366–380, 2014.

[8] R. Lotufo, Z. Malik, and K. Czarnecki, "Modelling the 'hurried' bug report reading process to summarize bug reports," in *Empirical Software Engineering*, vol. 20, no. 2, 2015, pp. 516–548.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[10] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[11] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," *TSE*, vol. 43, no. 3, pp. 272–297, 2017.

[12] R. P. Gopalan and A. Krishna, "Duplicate bug report detection using clustering," in *australian software engineering conference*. IEEE, 2014, pp. 104–109.

[13] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, and A. Schroter, "What makes a good bug report," in *TSE*, vol. 36, no. 5, 2010, pp. 618–643.

[14] Y. Tian, D. Lo, X. Xia, and C. Sun, "Automated prediction of bug report priority using multi-factor analysis," in *EMSE*, vol. 20, no. 5. IEEE, 2015, pp. 1354–1383.

[15] X. Yang, D. Lo, L. Li, X. Xia, T. F. Bissyandé, and J. Klein, "Characterizing malicious android apps by mining topic-specific data flow signatures," *IST*, vol. 90, pp. 27–39, 2017.

[16] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[17] X. Wei and W. B. Croft, "Lda-based document models for ad-hoc retrieval," in *SIGIR*, 2006, pp. 178–185.

[18] A. Sureka and K. V. Indukuri, "Linguistic analysis of bug report titles with respect to the dimension of bug importance," in *bangalore annual compute conference*. IEEE, 2010.

[19] L. Guerrouj, D. Bourque, and P. C. Rigby, "Leveraging informal documentation to summarize classes and methods in context," in *ICSE*, vol. 2. IEEE, 2015, pp. 639–642.

[20] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *ICPC*. IEEE, 2013, pp. 23–32.

[21] X. Gu and S. Kim, "" what parts of your apps are loved by users?"(t)," in *ASE*. IEEE, 2015, pp. 760–770.

[22] B. Xu, Z. Xing, X. Xia, and D. Lo, "Answerbot: automated generation of answer summary to developersź technical questions," in *ASE*. IEEE Press, 2017, pp. 706–716.

[23] X. Hu, G. Li, X. Xia, D. Lo, S. Lu, and Z. Jin, "Summarizing source code with transferred api knowledge," in *IJCAI*, 2018.

[24] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation," in *ICPC*, 2018.