Natural Language Processing (NLP) Applied on Issue Trackers

Mathias Ellmann
University of Hamburg
Hamburg, Germany
mathias.ellmann@uni-hamburg.de

ABSTRACT

In the domain of software engineering NLP techniques are needed to use and find duplicate or similar development knowledge which are stored in development documentation as development tasks. To understand duplicate and similar development documentations we will discuss different NLP techniques as descriptive statistics, topic analysis and similarity algorithms as N-grams, the Jaccard or LSI algorithm as well as machine learning algorithms as Decision trees or support vector machines (SVM). Those techniques are used to reach a better understanding of the characteristics, the lexical relations (syntactical and semantical) and the classification and prediction of duplicate development tasks. We found that duplicate tasks share conceptual information and are rather created by inexperienced developers. By tuning different features to predict development tasks with a gradient or a Fidelity loss function a system can identify a duplicate tasks with a 100% accuracy.

CCS CONCEPTS

• **General and reference** → **General conference proceedings**; **Computing standards, RFCs and guidelines**; **Empirical studies**; **Evaluation**; **Experimentation**; *Reliability*; *Metrics*;

KEYWORDS

Natural Language Processing (NLP), Duplicates, Document Analysis

ACM Reference Format:
Mathias Ellmann. 2018. Natural Language Processing (NLP) Applied on Issue Trackers. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering (NL4SE '18), November 4, 2018, Lake Buena Vista, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3283812.3283825

1 INTRODUCTION

Nowadays developers do not only use software development documentations such as wikis, project work logs [16] or API reference documentations [17] to develop software. Developers also use documentations and their embedded media files that are created and maintained by the software community active on websites as Stack Overflow, Eclipse Bugzilla or YouTube [18, 19, 30, 31]. Software developers create over 3,000 [1] software documents as development posts every day. This might lead to software development documents that can be similar to each other by their characteristics or knowledge included. The software documents might be synthetically or even semantically similar to each other as code clones [11] are and can be distinguished into different similarity types such as a duplicate or similar development documentation.

Software developers spent 20% of their time on searching for information [10, 26]. They take a look on similar software documentations that include similar and additional information as an API reference document and/or Stack Overflow post to get new inspirations on how to solve their current development task. A linkage of development tasks that of the same type that have the same software solution with a similar approach might safe time that can be used to evolve new software solutions. This might increase the productivity of the developer's work day [15].

The software community is often unsure how and when to declare a software document as similar. Bettenburg et al. or Rakha et al. [4, 22] discusses several criteria when a software document as bug report or a task can be considered as a duplicate. This uncertainty might lead to a point where similar software documents will be deleted in which it can be very useful for software developers [4]. To understand the similarity of software documents and to predict them there is a need for empirical studies and several NLP techniques [12, 20].

2 RESEARCH DATA AND METHOD

Software developers organize their work in issue trackers as Eclipse Bugzilla, Mozilla or Open Office [25]. They set different priorities and severities as well as sub-task their development tasks [5, 24, 33]. There are 29 bug reports submitted every day [3] and there exist over 225.000 bug reports in Eclipse as well as 420.000 bug reports in Mozilla [4] already in 2008! Over 20% of the bug reports in Eclipse and over 30% in Mozilla is duplicated.

Every development documentation in Eclipse Bugzilla has a same structure (and is very similar to other issue trackers as Mozilla, Open Office etc.)). It starts with a task summary as well as other information that allocate the task to an Eclipse product and component. A bug report defines different fields in the development documentation that shows the relation between the current task and other tasks as "Duplicates" [2].

To study duplicate bug development tasks we conduct a literature review that especially focus on the five analysis to understand and find duplicate development tasks. We found research papers and extract findings that are relevant for the five types of analyses [8] - characteristic, syntactic, semantic, classification and prediction

[1] https://api.stackexchange.com/2.2/info?site=stackoverflow
[2] https://bugs.eclipse.org/bugs/show_bug.cgi?id=227639

NL4SE '18, November 4, 2018, Lake Buena Vista, FL, USA

ACM ISBN 978-1-4503-6055-5/18/11…$15.00
https://doi.org/10.1145/3283812.3283825

analysis - to understand and find duplicate tasks. For every paper we searched for the first publication which especially focus on the relevant aspects for the analysis. Then we followed the citation list and sorted the papers by the analysis made. Overall we found 35-40 papers and summarized the main findings and the NLP techniques used in this paper.

3 TYPES OF ANALYSIS

The ultimate goal of this paper is to show the application of different NLP techniques to study duplicate development documentation. As recently published [8] to understand similar software development documentation we also differentiate between a characteristic analysis to understand the characteristics of duplicate software development documentation. The syntactic and semantic analysis will show how duplicate development documentation are lexically related to each other. The classification and prediction analysis will show what features can be used to understand and to predict a duplicate software development documentation.

4 CHARACTERISTIC ANALYSIS OF DUPLICATES

4.1 Frequencies and Structure of Duplicates.

In companies as Sony Ericsson [23] there are 10% (defect) duplicate development tasks. In Mozilla [2] there are 30% duplicate development tasks. In Eclipse Bugzilla [4] there are about 20% duplicate development tasks. A duplicate bug report has a bug id, summary, description and comments. Also other informations does exist in a development tasks as the project name [3], the priority as well as the severity of a bug [28] that characterizes a bug as well as other contextual informations that can be derived from the system as the used frameworks or libraries [1].

4.2 Content of Duplicates.

Bettenburg et al. [4] found that some Eclipse duplicates are obvious to identify because they have the same title and/or description for example "test". The ids of duplicate development tasks can be very close to each other as Sureka et al. [29] found (50% of duplicates have a difference of 3138 bug ids). Bettenburg et al. [4] also think that duplicate development tasks can add additional information as components, patches, screenshots and stack traces. Wang et al. [32] found that developers use paraphrases (the edit-field Vs. input line field) in duplicates and show that it is very hard to find paraphrases of technical descriptions. Runeson et al. [23] found in an interview with analysts that duplicates could often not be understood.

4.3 Reasons for Duplicates.

Cavalcanti et al . [5] found that the number of duplicates does not depend on the type of a report as an enhancement or a defect. Davidson et al. [6] found that consumer projects does not have a higher number of duplicates because of contributers that does not have enough knowledge in using the software repository. Cavalcanti et al . [5] found that the software size does not influence the number of duplicates. They also found that developers which working in a private project use more common words and using a template when creating a development task then in an open source project that might influence the number of duplicates too.

[3] https://wiki.eclipse.org/WTP/Conventions_of_bug_priority_and_severity

4.4 Creation and Closing Time of Duplicates.

Interestingly Bettenburg et al. [4] found that duplicate reports were approximately added at the same or after short period of time (0.18 - 0.83 days) of the original or master report as Rakha et al. [22] found. Cavalcanti et al. [5] found that the number of duplicates increase in the first year of a software project. Davidson et al. [6] found that if more bugs exist it takes longer to find duplicates. In comparison to open source tracking systems as Bugzilla the frequency of existing duplicates in Sony Ericsson is around 20 days for more then half of the duplicates [23] which shows a strong correlation between open and close dates.

5 SYNTACTIC ANALYSIS OF DUPLICATES

5.1 Terms & Paraphrases.

Runeson et al. [23] found that developers use synonyms to describe duplicate development tasks. Interestingly developers use also words that are sensefull in the same development context [14] as crash and dump. Sureka [29] found morpholic variations of word when describing duplicate development tasks as switching and changing to a bug which can be partly identified with a n-gram algorithm that evaluates a sequence of characters (here: **ing**). For duplicate bug reports that share more information as the noun as "create getter" or "generate getter" the n-gram measure might be more accurate. Developers describe duplicate development tasks differently maybe because of a different interpretation of the failure source as TextEditor or TextViewer. Sureka [29] found also that developers use short form as "config" or "configuration" as well as "temp" and "temporary". They also hyphenated phrases as "Ctrl-7" and "Ctrl F7".

5.2 Informations & Concepts.

Runeson et al. [23] evaluated duplicate development tasks (in a Sony Ericsson bug triaging system) by using the Jaccard algorithm and others. They mentioned that often information of a development co-occur in the document e.g. in the description of a development tasks which is also mentioned in the severity field. Duplicated text has its origin from the same users e.g. using same phrases. They also think that different problems can be described by using the same vocabulary. Sureka [29] found also that duplicate development tasks contains misspelled words. They also found that duplicates contain similar concepts which can be found by using my using a n-gram mechanism that to measure a sequence of characters. For example a development screencast "viewerContributions" and the duplicate development task has "contributions" in its title. Wang et al. [32] made also similar findings that developer often used paraphrases to describe duplicate development tasks.

5.3 Similarities Between Duplicates.

Sureka [29] et al. could reach a similarity of development tasks of up to 73.62% if both duplicate development tasks are located in the same product and component (15.00%). There is a less syntactical similarity if duplicates are located in another component. Even less

when they are located in another product but are issued in a same component (4.48%). Interestingly duplicate development tasks have a higher syntactical similarity if they are not located in the same product or component (6.90%).

6 SEMANTIC ANALYSIS OF DUPLICATES

6.1 Term Dependencies of Duplicates.

Sureka [29] found similar concepts in duplicate development tasks especially in the system messages. For example they found StringIndexOutOfBoundsException and StingIndexOutOfBounds in development tasks. Wang et al. [32] found paraphrases in duplicate development tasks especially in their summaries. We think a semantic similarity between duplicate development tasks does depend on the chosen terms (class names, method names etc.) when programming software or when describing an issue in an issue tracker.

6.2 Context Dependencies of Duplicates.

Nguyen et al. [21] made several observations regarding to the semantic similarity of duplicate bugs as developers using similar terms used that can be understood in the same development context. They observed that terms were mentioned in the same development context as RemoteFileAction and org.eclipse.ui.DefaultTextEditor which shows also that a semantic relation has not only be focused on its text then also on other context information that can be associated with the e.g. project context. Alipour et al. [1] found also several topics (by using the LDA algorithm) in a Android development task that are typical for the software development tasks as 3G, alarm, audio, SD-card.

6.3 Identifying a Semantic Similarity Between Duplicates.

Sureka et al. [29] found that measuring the similarity from a character level might be needed because of compound words as Out of memory Vs. OutOfMemoryError example which is a typical writing pattern for coding [4]. A semantic comparison of duplicates is also possible when considering the context information [1] of duplicate development task. Dit et al. found [7] in their semantic comparison of the comments of the developments tasks descriptions to detect duplicate development tasks.

7 CLASSIFICATION AND PREDICTION OF DUPLICATES

7.1 Feature Selection.

Researchers use textual features that are extracted from the summary or the description of a development task. They also use categorical features as a priority of a development task, the component name, type or version. They also use contextual features [1] that can be extracted from the architecture of a system of several development tasks e.g. their topics as 3G, alarm, android_market or others. Duplicate development tasks provide several information to classify them. For example Alipour et al. [1] use technical terms derived from the android system (especially its architecture) to classify a duplicate. Sureka et al. [29] calculated character-based n-grams between the text mentioned in duplicates (summary and description). Runeson et al. [23] calculated the cosine similarity between dupicates. Sun et al. [28] extracted 54 different features (27 features based on single words and 27 features that were based on bigrams) that were calculated on the similarity (e.g. idf of different input parameters) of the text (summary and description) of duplicate and non-duplicate pairs. Sun et al. [27] extended their method by a parameter tuning based on a stochastic gradient descent that enables to tune the free parameters in the used algorithm (BM25Fex). Lazar et al. [13] used up to 25 features (textual as well as categorical (fields as product name, component name) to classify duplicates and non-duplicates. Zhou [34] used nine textual/statistical features to classify duplicates and non-duplicates. The nine textual features were calculated from the text of two development tasks (idf, Jaccard similarity..). The nine textual features were also weighted with a Fidelity loss function and a stochastic gradient descent algorithm.

[4] https://msdn.microsoft.com/en-us/library/ms229043(v=vs.100).aspx

7.2 Feature Extraction.

Zhou et al. [34] used nine different calculation methods to classify a duplicate development task pair. We think that the overall idea of all the calculation methods which they used was to compare the frequency and existence of words that occur in different information entities of a development task as the summary and the descriptions. To decrease the influence of large numbers (many words occur in two descriptions) they also used a logarithmic function to lower the influence of long descriptions. Runeson et al. [23] also focus on the existence of words and similarity of text that are calculated by the Jaccard, Dice and Cosine algorithm. Lazar [13] et al. and others also used limitation techniques to improve the similarity measurement as as word.net to find similar words in duplicate development tasks.

7.3 Feature Optimization.

It is possible to tune parameters to improve the classification of duplicate development tasks. When researchers started to classify and predict duplicates they treated the documents equally. Nguyen et al. [21] uses the BM25F algorithm to weight their features. The overall benefit of using the BM25 then the TF-IDF that it allows to change the free parameter k to get a higher similarity value between duplicate development tasks because its possible to react on the number of terms which gives the chosen textual features a different weights. Nguyen et al. [21] used also the LDA algorithm to find similarities between duplicate development tasks. The found the number of topics K for the LDA algorithm can range from 140-320 for Eclipse or 100-240 for Mozilla. Zhou [34] et al. uses different textual features that compare the similarity of duplicate development tasks. The researchers weight their features by iterating through a training set until they received the best ranking of duplicate development tasks. Kaushik et al. [9] et al. found also that for Eclipse the optimal number of topics by using the LSI or LDA algorithm is 400-550 and for Mozilla they are 400-500 which shows also that the number of topics might depend on the features chosen as well as the source of data.

7.4 Duplicate Prediction.

Sun et al. [28] used a support vector machine (SVM) to identify duplicate development tasks. They used 54 features and could reach

a recall of up to 0.6 when considering a list of 10 items. Lazar [13] used several machine algorithm (Nearest Neighbors, Linear SVM, RBF SVM, Decision Tree, Random Forest or Naive Bayes) by using 25 features to identify duplicates. For all repositories they could reach a 0.99 accuracy, precision and recall. Also Alipouret al. [1] could reach an up to 0.95 accuracy by using contextual information as the android architecture words, non-functional requirement (NFR) or Android topic words as well as different machine learning algorithm (O-R, LogisticRegression, Naive Bayes, C4.5 and K-NN). The C.4.5 algorithm, a decision tree model outperformed best with the used features.

8 SUMMARY OF THE WORK

In this work we have discussed the characteristics of duplicate development tasks. We found that it might often depend on knowledge and expertise to use the right terms in the right development context. Finally, we have discussed that several features are needed to identify a development task. Contextual features might be predominant when developers do not tune the features e.g. by using a gradient descent or combining the most informative features that serve as strong indicators for a duplicate development task.

We found that concepts let appear tasks as semantically similar. A concept in a development context can be a basis class that were extended or a similar method that were used in development tasks. Similar terms in duplicate development tasks might be better understood when working in a development context. There are several systems as Mylyn or others that support developers when resolving issues. We think there should be a supportive system available in issue trackers that aggregates not only system data then also context data. This might extend the development documentation by adding the most relevant terms to a duplicate development task.

REFERENCES

[1] Anahita Alipour, Abram Hindle, and Eleni Stroulia. 2013. A contextual approach towards more accurate duplicate bug report detection. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 183–192.

[2] John Anvik, Lyndon Hiew, and Gail C Murphy. 2005. Coping with an open bug repository. In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*. ACM, 35–39.

[3] John Anvik, Lyndon Hiew, and Gail C Murphy. 2006. Who should fix this bug?. In *Proceedings of the 28th international conference on Software engineering*. ACM, 361–370.

[4] Nicolas Bettenburg, Rahul Premraj, Thomas Zimmermann, and Sunghun Kim. 2008. Duplicate bug reports considered harmfulâĂę really?. In *Software maintenance, 2008. ICSM 2008. IEEE international conference on*. IEEE, 337–345.

[5] Yguaratã Cerqueira Cavalcanti, Paulo Anselmo da Mota Silveira Neto, Daniel Lucrédio, Tassio Vale, Eduardo Santana de Almeida, and Silvio Romero de Lemos Meira. 2013. The bug report duplication problem: an exploratory study. *Software Quality Journal* 21, 1 (2013), 39–66.

[6] Jennifer L Davidson, Nitin Mohan, and Carlos Jensen. 2011. Coping with duplicate bug reports in free/open source software projects. In *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*. IEEE, 101–108.

[7] Bogdan Dit, Denys Poshyvanyk, Andrian Marcus, et al. 2008. Measuring the semantic similarity of comments in bug reports. *Proc. of 1st STSM* 8 (2008), 64.

[8] Mathias Ellmann. 2017. On the similarity of software development documentation. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 1030–1033.

[9] Nilam Kaushik and Ladan Tahvildari. 2012. A comparative study of the performance of IR models on duplicate bug detection. In *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*. IEEE, 159–168.

[10] Andrew J Ko, Robert DeLine, and Gina Venolia. 2007. Information needs in collocated software development teams. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on*. IEEE, 344–353.

[11] E Kodhai, S Kanmani, A Kamatchi, R Radhika, and B Vijaya Saranya. 2010. Detection of type-1 and type-2 code clones using textual analysis and metrics. In *Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on*. IEEE, 241–243.

[12] Klaus Krippendorff. 2012. *Content analysis: An introduction to its methodology*. Sage.

[13] Alina Lazar, Sarah Ritchey, and Bonita Sharif. 2014. Improving the accuracy of duplicate bug report detection using textual similarity measures. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 308–311.

[14] Walid Maalej and Mathias Ellmann. 2015. On the similarity of task contexts. In *Proceedings of the Second International Workshop on Context for Software Development*. IEEE Press, 8–12.

[15] Walid Maalej, Mathias Ellmann, and Romain Robbes. 2016. Using contexts similarity to predict relationships between tasks. *Journal of Systems and Software* (2016).

[16] Walid Maalej and Hans-Jörg Happel. 2010. Can development work describe itself?. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. IEEE, 191–200.

[17] Walid Maalej and Martin P Robillard. 2013. Patterns of knowledge in API reference documentation. *IEEE Transactions on Software Engineering* 39, 9 (2013), 1264–1282.

[18] Walid Maalej, Rebecca Tiarks, Tobias Roehm, and Rainer Koschke. 2014. On the comprehension of program comprehension. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 23, 4 (2014), 31.

[19] Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen. 2015. Code, camera, action: how software developers document and share program knowledge using YouTube. In *Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on*. IEEE, 104–114.

[20] Tim Menzies, Laurie Williams, and Thomas Zimmermann. 2016. *Perspectives on Data Science for Software Engineering*. Morgan Kaufmann.

[21] Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N Nguyen, David Lo, and Chengnian Sun. 2012. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 70–79.

[22] Mohamed Sami Rakha, Weiyi Shang, and Ahmed E Hassan. 2016. Studying the needed effort for identifying duplicate issues. *Empirical Software Engineering* 21, 5 (2016), 1960–1989.

[23] Per Runeson, Magnus Alexandersson, and Oskar Nyholm. 2007. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 499–510.

[24] Serada. 2009. How do you handle related bugs in Bugzilla? https://stackoverflow.com/questions/1486580/how-do-you-handle-related-bugs-in-bugzilla. (Accessed on 28/09/2009).

[25] Nicolas Serrano and Ismael Ciordia. 2005. Bugzilla, ITracker, and other bug trackers. *IEEE software* 22, 2 (2005), 11–13.

[26] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. 2010. An examination of software engineering work practices. In *CASCON First Decade High Impact Papers*. IBM Corp., 174–188.

[27] Chengnian Sun, David Lo, Siau-Cheng Khoo, and Jing Jiang. 2011. Towards more accurate retrieval of duplicate bug reports. In *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*. IEEE, 253–262.

[28] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau-Cheng Khoo. 2010. A discriminative model approach for accurate duplicate bug report retrieval. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 45–54.

[29] Ashish Sureka and Pankaj Jalote. 2010. Detecting duplicate bug report using character n-gram-based features. In *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*. IEEE, 366–374.

[30] Rebecca Tiarks and Walid Maalej. 2014. How does a typical tutorial for mobile development look like?. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 272–281.

[31] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web?: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 804–807.

[32] Xiaoyin Wang, David Lo, Jing Jiang, Lu Zhang, and Hong Mei. 2009. Extracting paraphrases of technical terms from noisy parallel software corpora. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, 197–200.

[33] Jie Zhang, Xiaoyin Wang, Dan Hao, Bing Xie, Lu Zhang, and Hong Mei. 2015. A survey on bug-report analysis. *Science China Information Sciences* 58, 2 (2015), 1–24.

[34] Jian Zhou and Hongyu Zhang. 2012. Learning to rank duplicate bug reports. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 852–861.