Semantic Patterns Extraction of Code Smells: Retrieving the Solutions of Bugs

Boyang Zhang[1][0000−0003−3994−4597]

[1] University of Tampere, Finland
[2] boyang.zhang@tuni.fi

Abstract. The understanding of code smells have exerted profound influence in the quality and the performance of programming codes. There are various type of code smells require various solutions. In order to interpret the solutions available in code smells, this research uses NLP (natural language programming) techniques to comprehend contents of messages from Technical Debt Dataset. Based on phrase structure rules, semantic patterns were extracted from the Dataset to build connection between trigger words and dependency tree. Verb Phrases are considered as the actions taken by programmers encountering code smells.

Keywords: Code smells · NLP · Phrase structure rules · Semantic patterns.

1 Introduction

With the development of software design, the improvement of existing codes have began to attract more and more attentions nowadays. In the perspective of technical debt which reflect limited performance and lower efficiency of code, there is an opportunity for programmers to interpret and update the existing codes by code smells. In computer programming, codes smells were introduced by Martin Fowler on improving the design of existing codes [1]. Codes smells indicates the current poor designing and quality of codes [2] that may trigger future labour and financial costs, such as blob; complex class; function decomposition; spaghetti code and so on [3]. With the emerging of big data, there are growing numbers of discussions on the domains of code smells available in social media. There are multiple code smell detection tools, such as SonarQube, PWD, JSpIRIT which can be used to identify unused variables, empty objection and so on. Moreover, the diffuseness of code smells has triggered different level of severity to developers and organizations [4]. Researches has tended to focus on code smells themselves, rather than the contents of messages of raised by code smells. In spite of these early observations, what are the exact contents of messages attract our research interests. NLP (natural language programming) techniques are used to analyze large scale dataset collections [5]. This present study uses NLP techniques to decompose the contents of codes smells with follow research questions: How the messages are composed of users to make commit_changes? What kind of semantic patterns can be extracted from the dataset?

2 Method

In this subsection, the method for extracting semantic patterns of the COMMIT_CHANGE messages is discussed. The dynamics of extracting key perspectives includes focus; technique; domain. Then we describe about the data collection in the code smells domain.

2.1 Study Design

Regarding code smells, there is a growing tendency of researches to evaluate the quality of codes. NLP (natural language programming) tools can be used to extract semantic patterns to identify the content of codes smells. Bird et al. (2009) suggested the pipeline architecture for an information extraction system (Fig. 1.) to extract structure data from raw text [6]. In the designing of this study, similar method is applied to extract relationships between various entities.

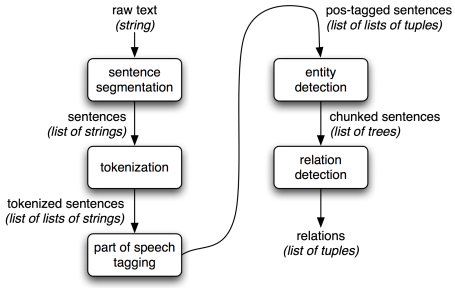

Fig. 1. Pipeline Architecture for Information Extraction System [6]

The messages of COMMIT_CHANGES can explore the content of code smells with in-depth understanding of the fault proneness and valuable responding strategies. Encountering various code smells, COMMIT_CHANGES messages are used for programmers to communicate towards various problems. In order to extract the contents of code smells, we use Phrase Structure Rules to extract semantic patterns as syntactic categories. For example, "You merge the branch in the trunk", Fig. 2. provides the sorting of reservation to claim the language-specific contexts.

In a certain type of research work, Gupta et al. provided a method for characterizing into three aspects as Focus, Domain, and Techniques[7]. Similarly, the above three aspects can be applied in this research with focus on software engineering and validation, domain is in the perspective of code smells; technical debts; software quality. Techniques are bugs detection; faults-fixing; solutions providing and so on. Semantic patterns are extracted from dependency tree of sentences from T the trigger word to (d) the dependency tree which is the T's successor[7], such as T can be 'update', 'edit', d can be 'bugs', 'fault'.

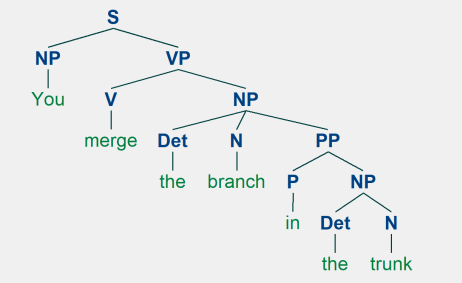

Fig. 2. Example on Phrase Structure Rules

Table 1. Techniques and Domains.

Techniques (to direct object)	plugin - package javadoc - jars check - constraints rejoin - classes
Domains	bugs-detection problems-solving

In the attempt to understand how the semantic patterns are composed in the messages extracted from COMMIT_CHANGES, Feature Request Mining [8] techniques can be used to extract keywords and phrase structure rules in the context. Based on the analysis of feature requests in the domains of codes smells, the sentences can be labelled from keywords to denotes the sentence as a request [8]. For example, "merge the branch in the trunk" is merge <request> in the trunk. Comparatively to the phrase structure rules, the combination of "V" + "NP" can suggests the triggered words to direct-object and subject in the sentence.

2.2 Data Collection

In the domain of codes smells, the data is coming from Technical Debt Dataset which is a curated dataset provides measurement data of 33 Java projects related information, the dataset is from Apache Software Foundation during the period of Oct 2000 to Jun 2018 [9]. This research analyzed the messages from COMMIT_CHANGES which are about the suggestions and recommendations concerning a specific COMMIT, COMMIT_CHANGES contains commitHash; file; projectID; Date; message and so on . COMMITS are the tags that users generate the comments, COMMIT_CHANGES directly connect with COMMITS which are the users generate solutions for the specific COMMIT. The message of COMMIT_CHANGES provides the contents of provided of the programmers. For example: "fixed globbing for zookeeper jar"; "changed all to libthrift".

In the initial screening of the dataset, the commitHash tag represent a single COMMIT object, and the commitHash might be duplicated with the same mes-

sages to the same COMMIT, therefore, the duplicates of messages are removed based on commitHash. The cleaning process of the dataset provides one unique messages from each COMMIT object.

2.3 Analysis of the Data

At beginning, all the raw texts with tags of COMMIT_CHANGES are extracted. After initial screening, the tag of messages are extracted from the raw texts. After duplicate checking, there are 6307 unique messages. Because the research interests are in semantic patterns, the number, stopword, punctuation, plain text, white space are removed before tokenization. Each tokens are labelled by postagged methods to categorize into present tense verb (VBP), noun (NN), preposition (PRP), coordinating conjunction (CC), adjective (JJ) and son on. In order to extract most frequent used verbs, base (VB), 3rd singular present (VBZ), past participle (VBN), gerund (VBG), simple past (VBD) categories are collected to build the keywords for expressing feature requests.

In the next stage, the tag of Verb Phrase (VP in Fig. 2.) was extracted from the sentence, from T trigger word to d which is the dependency connecting with the trigger words, such as direct-object or subject. In feature request mining, it is the <request> or <existing feature>'s contents which represent the phrase structure rules. Such as, "use generic_method for emptyMap instead of EMPTYMAP constant" can be clarified into: use <request> instead of <existing feature>.

3 Results and Findings

This section is to report the results and findings of the methodologies applied in study design. The first subsection illustrates verbs as keywords to express feature requests. The second subsection indicates how the phrase structure rules applied to extract <request> and <existing feature> from the sentences.

3.1 Verbs as Keywords

In order to extract the lexical categories of verbs, the NLTK python packages were used to classify words into POS-tagging which tags words automatically [6]. All types of verbs formats were extracted from the sentences. In Table 2. there are the most frequent used verbs extracted. A threshold boundary was set with verbs >5, verbs less or equal than 5 are not considered in current stage. The threshold boundary can be modified based on the simple size.

3.2 Phrase Structure Rules Extraction

There are various constituent parts to form phrase structure rules. In this research, Verb Phrase (VP in Fig.2.) gives an internal structure of semantic patterns. A verbal phrase might consist V, NP, PP. In order to built the dependency

Table 2. Most Frequent Used Verbs in the Dataset

remov, make, get, build, exist, failur, depend, refer, tri, expect, find, origin, resourc, exclud, take, tracer, expand, know, occur, transit, accumuloconfdir, appear, gitignor, walk, detect, intend, think, span, fri, seem, filenam, mdrobclouderacom, suspend, favor, protect, remain, restor, deepcopi, sensit, accumuloenvsh, save, collect, hold, hope, startsh, accumuloinputformat, david, defend, prefer, referenc

graph which is the Verb Phrase for semantic patterns, we use below Bi-gram formula to test the combinations of verb plus <request> or <existing feature> to form the connection between T trigger word to d direct-object or subject. Bi-gram uses the co-occurrence of two words, to divide the probability of the first words. *P()* represents the conditional probability of the token and next one: W_k, W_{k-1}.

Bi-gram formula for probability on two adjacent elements on tokens.

$$P(W_{k-1}|W_k) = \frac{P(W_{k-1}, W_k)}{P(W_{k-1})} \tag{1}$$

After the calculation of Bi-gram, the semantic patterns were extracted based on the Table 2.'s verbs collection. For example, in Table 3. it is the "remove" verb based patterns extracted from the dataset. It can be seen clearly that "remove stray" takes 68.18 in Bi-gram result, and "remove references" takes 46.49. Based on the phrase structure rules, irrelevant semantic patterns were removed such as "remove the" and so on.

Table 3. Semantic Patterns extracting, Bi-gram.

remove stray, 68.18
remove references, 46.49
remove redundant, 45.93
remove unneeded, 43.59
removed instamo, 35.34
Remove warning, 33.19
removed VFS, 30.17
remove stack 26.29

4 Conclusion

In order to improve the quality and performance of codes, this research uses NLP techniques to interpret contents of messages in Technical Debt Dataset. Phrase structure rules are applied to extract semantic patterns from the dataset to understand techniques and methods used in the domain of codes smells. This

research sheds the lights on addressing the contents of code smells by feature request mining. Moreover, this research can help programmers to understand the contents of code smells and what actions can be taken to improve the quality of codes.

4.1 Limitations and Future Studies

The semantic pattern extraction is based on the phrase structure rules, with various phrase structure rules, the interpretation of the results may vary a lot. This study apples the typical of V + NP, other rules can also be applied, such as NP = Det (determiner) + N (noun) and so on. Deeply, based on the domain of code smells, specific trigger words and dependency tree can be collected to understand the contents of messages. In the current stage, Bi-gram method is used to extract the patterns on two tokens. The future can utilize Tri-gram to interpret 3 tokens patterns. And the multi-word expressions can also be taken into consideration, such as "scrip command" can be interpreted as one word, similarly as "new york" is understand as one city location not "new" and "york". Furthermore, it is possible to utilize RNN (Recurrent Neural Networks) techniques to provide additional advantage of gathering all pre-order vocabulary into consideration for semantic patterns extraction. And also the applying of LSTM (Long Short-Term Memory) method can provide semantic parsing for the task of mapping contents of messages in the domain of code smells.

References

1. Flower, M.: Refactoring. Improving the Design of Existing Code. Addison-Wesley, (1999).
2. Vidal, S., Vazquez, H.,Diaz-Pace, J. A., Marcos, C., Garcia, A., and Oizumi, W.: JSpIRIT: a flexible tool for the analysis of code smells, 34th International Conference of the Chilean Computer Science Society (SCCC), Santiago, pp. 1-6 (2015).
3. Fontana, F. A., Lenarduzzi, V., Roveda, R and Taibi D.: Are Architectural Smells Independent from Code Smells? An Empirical Study. Journal of Systems and Software. 154.10 (2019).
4. Sarimäki, N., Lenarduzzi, V., and Taibi, D.: On the Diffuseness of Code Technical Debt in Java Projects of the Apache Ecosystem. Proceeding of International Conference on TechnicalDebt (TechDebt 2019), Montreal, Canada, (2019).
5. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S and McClosky D.: The Standford CoreNLP Natural Language Processing Toolkit. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations,Baltimore, Maryland. pp. 55-60. (2014).
6. Bird, S., Klein, E., and Loper. E.: Natural Language Processing with Python 1st edn. O'Reilly Media, Inc, (2009).
7. Gupta, S., Manning, C.: Analyzing the Dynamics of Research by Extracting Key Aspects of Scientific Papers, Proceedings of 5th International Joint Conference on Natural Language Processing, pp. 1–9. Asian Federation of Natural Language Processing, (2011).

8. Iacob, C., Harrison, R.: Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. MSR 2013, San Francisco, CA, USA. pp. 41–44, (2013).
9. Lenarduzzi, V., Sarimäki, N., and Taibi, D.: The Technical Debt Dataset. Proceedings of the 15th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE19), Recife, Brazil, (2019).