Catalogen: Generating Catalogs of Code Examples Collected from OSS

Daiki Takata*, Abdulaziz Alhefdhi†, Maipradit Rungroj*,
Hideaki Hata*, Hoa Khanh Dam†, Takashi Ishio* and Kenichi Matsumoto*
*Nara Institute of Science and Technology, Japan
Email: {takata.daiki.ta3, maipradit.rungroj.mm6, hata, ishio, matumoto}@is.naist.jp
†University of Wollongong, Australia
Email: {aa043,hoa}@uow.edu.au

***Abstract*—Given a Java class name as a query, Catalogen generates a catalog of code examples collected from open source software projects. A set of code examples are categorized based on the similarity of N-gram features in code, and automatically generated comments are attached to all code examples.**

1. Catalogs of Code Examples

Code examples are considered to be an important knowledge source of developers [1]. To create reference documentation for a Java class, we develop Catalogen, a system that generates a catalog of code examples collected from open source software (OSS) projects. A generated catalog contains categorized code examples and corresponding code comments, which are automatically generated. Code examples are actual implementations in OSS source code, and include an instantiation of the class and sequences of instance method calls.

Techniques. Catalogen consists of three techniques. (i) Code example collection. Source code related to a query Java class is searched from OSS projects. From the search results, appropriate scope of code snippets are extracted. (ii) Code comment generation. Similar to pseudo-code generation using sequence-to-sequence (seq2seq) models for statistical machine translation [2], code comments in natural language are generated by translating from the collected code examples. (iii) Code example categorization. Clustering based on N-gram features of code is applied for the collected code examples.

Data Sources. To collect source code related to a query Java class, we employ `searchcode`[1], a web service for code search. It takes query words as input and provides source code fragments related to the query from OSS projects hosted in GitHub, Bitbucket, Google Code, SourceForge and GitLab. For code example search, we do not target Stack Overflow, because it is reported that there exist API misuses even in the accepted posts [3]. Instead, we target actual usages in OSS projects, as they are considered to be reliable in practice. Users can learn frequent practical patterns of API usages, as it is reported that the usage of APIs obeys Zipf distribution [4].

For training seq2seq models, we processed the source code of the Apache POI project and built a corpus of 13,000 pairs of Java methods and their corresponding Javadoc comments.

2. Techniques

2.1. Code Example Collection

Given a Java class name as a query, the web API of `searchcode` returns a result in a JSON format; it includes source file names, their line numbers that match the query word (i.e. the class name of interest), and file IDs to access the file contents on the service.

Code examples considering dataflow are beneficial to learn the usage of API. To extract appropriate scope of code examples from `searchcode`, Catalogen collect the following statements in the same scope.

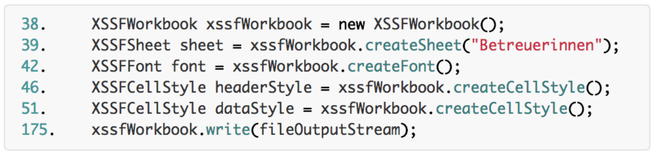
1) A statement defining a variable v of the given class C.
2) Statements calling an instance method of the class C using the variable v.

A code example satisfying the above conditions likely shows how to get an instance of the class and call an instance method.

A strict check on the conditions requires a semantic analysis on source code. It is impossible, since the web service provides a source file alone. The definitions of external classes used in the source file are unavailable. Instead, Catalogen checks the conditions using patterns of tokens as follows.

- An occurrence of the class name C followed by an identifier v is regarded as a definition of the variable v. The tool recognizes the tokens as the start of a code example.
- If another occurrence of v followed by a dot appears in the same code block, the tokens are regarded as an instance method call for v. The last line of such instance method calls is recognized as the end of the code example.

1. https://searchcode.com/



```
XSSFWorkbook xssfWorkbook = new XSSFWorkbook();
XSSFSheet sheet = xssfWorkbook.createSheet("Betreuerinnen");
XSSFFont font = xssfWorkbook.createFont();
XSSFCellStyle headerStyle = xssfWorkbook.createCellStyle();
XSSFCellStyle dataStyle = xssfWorkbook.createCellStyle();
xssfWorkbook.write(fileOutputStream);
```

Figure 1. A code example of `XSSFWorkbook`

The patterns approximate dataflow with respect to a variable v whose type is C. The constraint of "same code block" checks the scope of the variable and also limits the size of a code example.

Figure 1 shows an instance of code example for class `XSSFWorkbook`. The instance includes the definition of a variable *xssfWorkbook* followed by the lines of method calls using the variable. To reduce the size of a code example, Catalogen shows only the lines including the defined variable.

2.2. Code Comment Generation

The previous step returns a number of code examples showing how a given API class has been used in practice. Those code examples however often do not have comments, which make it difficult for developers to understand them. Hence, in this step we automatically generate a high level description for each code snippet. While the code snippet is written in a programming language (e.g. Java), its description is expressed in natural language. Our machinery is thus built upon the notions and ideas in Neural Machine Translation. Specifically, we employ the deep learning-baseed sequence-to-sequence (seq2seq) model to automatically learn both syntactic and semantic features representing a code snippet, and the relation between them and words in a comment.

This seq2seq model has two important components: an Encoder and a Decoder, each of which consists of a Long Short-Term Memory (LSTM). LSTM [5] is a recurrent neural network, which maps a sequence of input vectors into a sequence of output vectors. In our model, a code snippet is parsed into a sequence of code tokens, which are input into the Encoder component. The output from the Encoder is then fed into the Decoder to generate a sequence of word tokens which represent the code's comment.

2.3. Code Example Categorization

Inverse Document Frequency (IDF) has been widely used in many applications because of its simplicity and robustness; however, IDF cannot handle phrases that are composed of more than one term. Because IDF gives more weight to terms occurring in fewer documents, rare phrases are assigned more weight than good phrases that would be useful in text classification. N-gram IDF is a theoretical extension of IDF for handing multiple terms and phrases by bridging the theoretical gap between term weighting and multi-word expression extraction [6].

Terdchanakul et al. reported that for classifying bug reports into bugs or non-bugs, classification models using features from N-gram IDF outperform models using topic modeling features [7]. Similary, we use N-gram features of code detected with N-gram IDF, for categorizing similar code examples.

By using the N-gram IDF, we vectorize token sequence of a code example into a feature vector. We use the group average method to consider outliers and classification sensitivities as a hierarchical clustering method. The number of clusters is determined by assuming an appropriate threshold value of the boundary where the characteristics of the classification greatly change is determined from the generated tree diagram.

3. Catalogen

A preliminary result is available on the following URL.

https://takata-daiki.github.io/catalogen/preliminary/

Catalogen is available on the following URL.

https://takata-daiki.github.io/catalogen/

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers JP15H02683, 18H03221, and 16H05857.

References

[1] C. Treude and M. P. Robillard, "Understanding stack overflow code fragments," in *Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sept 2017, pp. 509–513.

[2] Y. Oda, H. Fudaba, G. Neubig, H. Hata, S. Sakti, T. Toda, and S. Nakamura, "Learning to generate pseudo-code from source code using statistical machine translation (t)," in *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 574–584.

[3] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online Q&A forum reliable?: A study of API misuse on stack overflow," in *Proceedings of the 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 886–896.

[4] D. Qiu, B. Li, and H. Leung, "Understanding the API usage in java," *Information and Software Technology*, vol. 73, pp. 81–100, may 2016.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] M. Shirakawa, T. Hara, and S. Nishio, "N-gram IDF: A global term weighting scheme based on information distance," in *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 2015, pp. 960–970.

[7] P. Terdchanakul, H. Hata, P. Phannachitta, and K. Matsumoto, "Bug or not? bug report classification using N-gram IDF," in *Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sept 2017, pp. 534–538.