COMMENT OR NOT TO COMMENT: THE EFFECTS OF COMMENTS ON METHOD NAME PREDICTION

Author

Nada Mouman
n.mouman@student.tudelft.nl

Supervisors

Annibale Panichella, Leonhard Applis

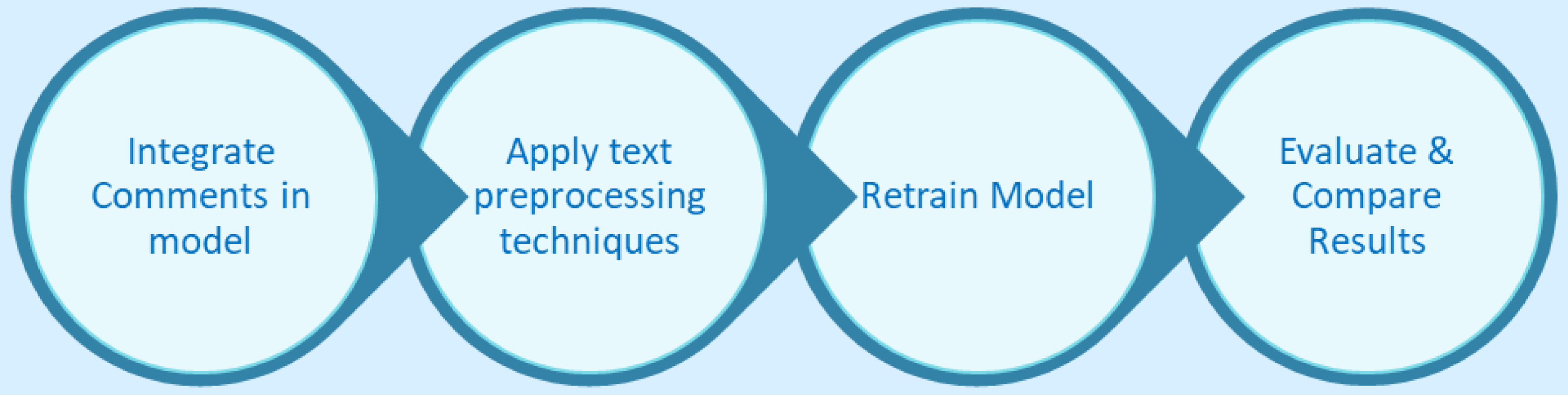
1 Introduction

- In recent years, there has been a growth in the usage of Machine Learning techniques in ***Software Engineering*** tasks [1]
- ***Method Name prediction:*** generates identifier given a method's code snippet
- Meaningful and conventional method identifiers are crucial to the comprehensibility of the software [2]
- ***Code2Seq*** is a model which can predict method names [3]
- ***Comments*** are ***not included*** during the preprocessing and training steps
- Studies have shown that comments improve the readability of the programs [4]

2 Research Questions

- What is the ***impact of comment***s on the performance of Code2Seq for method name prediction?
- How does ***"including Javadoc comments"*** impact the performance of code2seq for method name prediction?
- How does ***"including inline comments"*** impact the performance of code2seq for method name prediction?
- How does ***"filtering the content of the comments"*** impact the performance of code2seq for method name prediction?

3 Methodology



4 Comment Encoding

- Code2Seq uses ***Abstract Syntax Trees*** (AST) to represent the source code
- Comments are included in the AST during preprocessing
- Each comment is associated with at most one parent node
- ***Orphaned Comments:*** comments not associated with any node [5]
- ***Keywords*** extracted from comments using ***TF-IDF*** [6]
- Comments preprocessed with ***Stopwords removal***

```
/** This method gets the first character of a string.
 */
char getFirstLetter(String a){
    return a.charAt(0);
}
```

Figure 1: Code snippet with comment

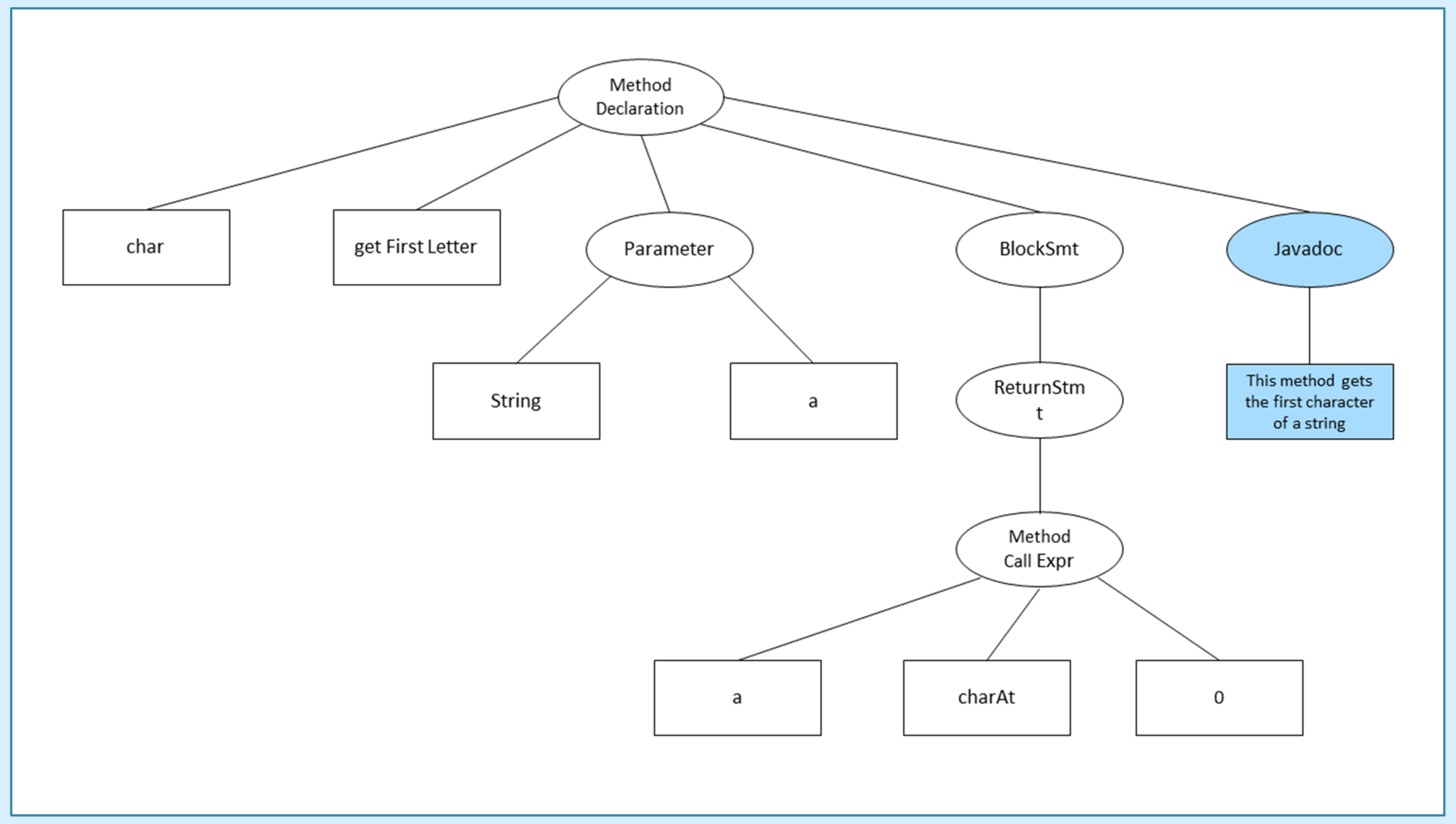


Figure 2: Abstract Syntax Tree with comment

5 Results

Model	Precision	Recall	F1
Original Code2Seq	47.30	36.92	41.47
Code2Seq + comments	**49.01**	37.44	**42.45**
Code2Seq + Javadoc	44.44	35.59	39.53
Code2Seq + inline comments	47.75	37.30	41.88
Code2Seq without stopwords	47.52	**39.14**	**42.93**
Code2Seq + TFIDF	**48.36**	35.88	41.19

6 Conclusion

- Improvement of ***2.4%*** in F1 score for the model with raw comments
- Gain of ***6%*** and ***3.5%*** in recall and F1 score respectively for model without stopwords
- ***Reduction*** in performance for model with javadoc
- ***Minimal improvement*** for TFIDF model and model with inline comments
- Extend models with orphaned comments
- Experiment with different amounts of keywords extracted from comments using TF-IDF

Github Repository

Related Literature

[1] X. Li, H. Jiang, Z. Ren, G. Li, and J. Zhang, "Deep learning in software engineering," 2018. DOI: 10.48550/ARXIV.1805.04825.
[2] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, "Suggesting accurate method and class names," ESEC/FSE 2015, pp. 38–49, 2015. DOI: 10 . 1145 / 2786805 .2786849.
[3] U. Alon, S. Brody, O. Levy, and E. Yahav, Code2seq: Generating sequences from structured representations of code, 2018. DOI: 10. 48550/ARXIV.1808.01400.
[4] T. Tenny, "Program readability: Procedures versus comments," IEEE Transactions on Software Engineering,vol. 14, no. 9, pp. 1271–1279, 1988. DOI: 10.1109/32.6171.
[5] Smith, N., Bruggen, D. V., & Tomassetti, F. JavaParser: Visited analyse, transform and generate your Java code base. 2017
[6] L.-P. Jing, H.-K. Huang, and H.-B. Shi, "Improved feature selection approach tfidf in text mining," vol. 2, 944–946 vol.2, 2002. DOI: 10 . 1109 / ICMLC . 2002 .1174522.