Enabling Log Recommendation Through Machine Learning on Source Code

Liudas Mikalauskas lmikalauskas@tudelft.nl

TU Delft

1. Logging

- Is used in software development.
- Is a common practice.
- Reveals important runtime information.
- Helps with debugging and maintenance.
- Should only reveal necessary information and not hurt performance

It is a challenging task [1-3].

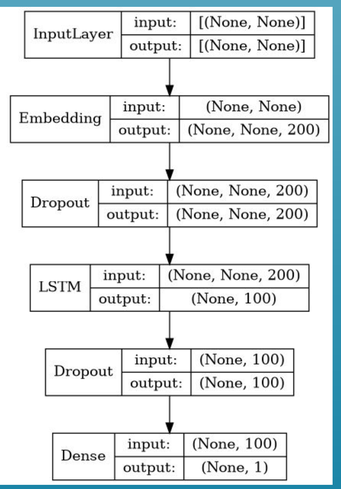
2. Research question

What is the performance of a log recommendation model developed following the methods of Li et al. [1], using CloudStack® source code as training data?

3. Methodology

- Creating a dataset (Figure 1)
  - Extract Java files
  - Extract methods from each file
  - Build Abstract Syntax Trees
  - Identify blocks
  - Label blocks
  - Remove all statements related to logging
  - For each block extract features (structural token sequence)
- Deep learning (Figure 2)
  - Embed each token using language processing (NLP)
  - Feed sequences of embeddings to a Recurrent Neural Network (RNN)
  - Convert output to a binary prediction
- Fine tuning
  - Adjust feature extraction, NLP, RNN algorithms
  - Adjust number of epochs, internal RNN states
- Evaluation
  - Calculate the performance of the model

Figure 2: Neural network model



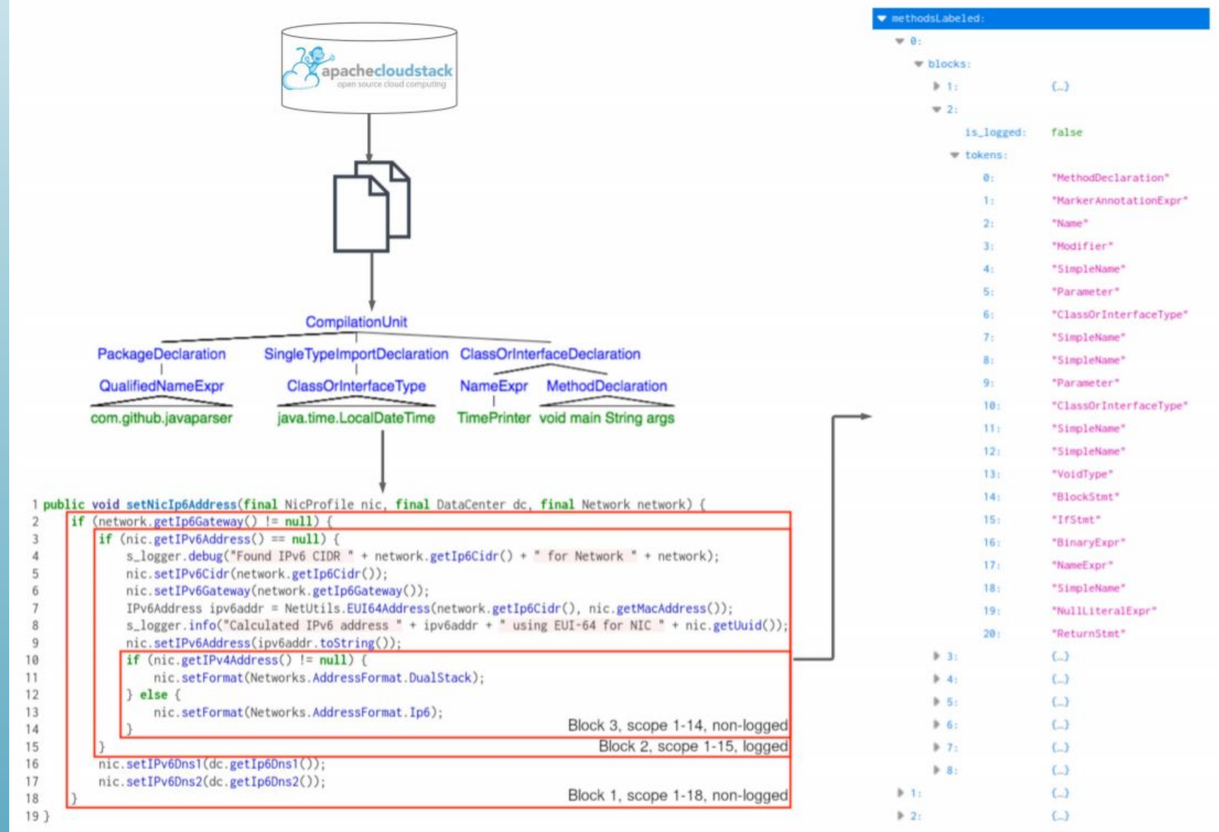


Figure 1: Flowchart of dataset creation method. This shows the most general steps

4. Results (Figure 3)

- F-Measure (FM) peaks at epoch 9 with a value of **0.57** (precision - 0.73, recall - 0.47)
- Average FM is **0.53** (average precision - 0.72, recall - 0.43)
- In 15 epochs the model showed potential for learning with 13% increase in FM and a positive sum of FM differences between every two neighbours

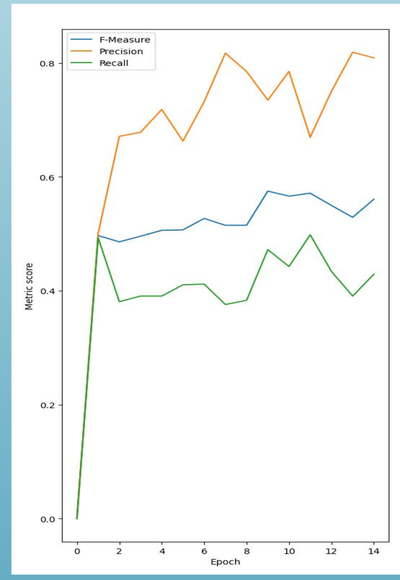


Figure 3: FM, precision and recall over 15 epochs

5. Conclusion

- Methods of Li et al. are reproducible
- Model trained on this specific dataset showed good performance predicting logs within project
- The performance is similar to that of Li et al. (they gained FM of 0.55)
- The gap between precision and recall is bigger than that of Li et al.
- A study on feature filtering was made and it revealed that not filtering features results in an increase of all tested metrics

6. Future recommendations

- Study computationally expensive configurations (more epochs, more hidden nodes, larger word vectors)
- Extend model to predict log level
- Study cross-project performance
- Investigate what causes a bigger gap between precision and recall

7. References

- [1] Li, Heng, et al. “Studying Software Logging Using Topic Models.” Empirical Software Engineering, vol. 23, no. 5, Oct. 2018, pp. 2655–94. DOI.org (Crossref), doi:10.1007/s10664-018-9595-8.
- [2] Zhu, Jieming, et al. “Learning to Log: Helping Developers Make Informed Logging Decisions.” 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, IEEE, 2015, pp. 415–25. DOI.org (Crossref), doi:10.1109/ICSE.2015.60.
- [3] Li, Zhenhao, et al. “Where Shall We Log?: Studying and Suggesting Logging Locations in Code Blocks.” Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ACM, 2020, pp. 361–72. DOI.org (Crossref), doi:10.1145/3324884.3416636.
- [4] Sabharwal, Navin and Ravi Shankar. Apache CloudStack Cloud Computing. Packt Publishing, 2103. Open WorldCat, http://www.totalboox.com/book/id-7445847629213878005.
- [5] Danny van Bruggen, Federico Tomassetti, Roger Howell, Malte Langkabel, Nicholas Smith, Artur Bosch, … Bernhard Haumacher. (2020, May 25). javaparser/javaparser: Release javaparser-parent-3.16.1 (Version javaparser-parent-3.16.1). Zenodo. http://doi.org/10.5281/zenodo.3842713