CS230

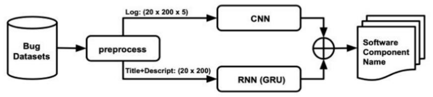
DeepBug: A hybrid of CNN and RNN approach for software bugs triage

Yuanliang Lu (lofus@stanford.edu), Lucy Gao (lgao@ibm.com), Dawny Liu (dawny@motorola.com)

https://youtu.be/za1P0DAajYA

Introduction

To save the manual effort spent on software bugs triage, we we built a deep neural network approach named as DeepBug which takes the bug report title, description and logs, triage that which software component the bug belongs to. Deepbug is designed as a hybrid architecture with a CNN and a RNN running in parallel. The CNN path keeps learning from the bug log file in a way mimicking manual log analysis by layers, and the RNN path works on learning the text sequences from bug title and description. The stream of the two paths are aggregated as the final output as a classifier.



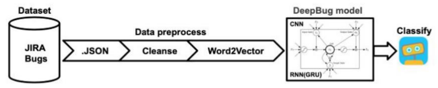
Data

With public dataset of 240K bug reports and internal data from JIRA system for 200K samples. Each sample comes with

- **Title**-- A summary for the bug, less than 30 words.
- **Description**-- A description of what are the scenarios and behaviors, less than 300 words.
- **Logs**-- Logs printed from the software stack.
- **Component** -- the ***data label***, software component

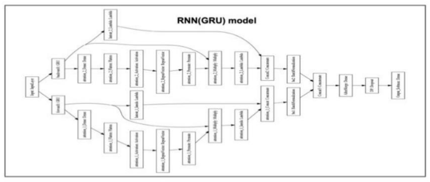
Features

Convert datasets into .json files, then we wrap up the bug title, description and component with a JSON object. The log trace are preprocessed with a separate JSON object. The .json files are cleansed, such as removing the tabs, the hex code and URLs. The title and description are embedded and vectorized as (20×200) dimensional, Log fragments are vectorized as $20 \times 200 \times 5$ dimensional matrices.



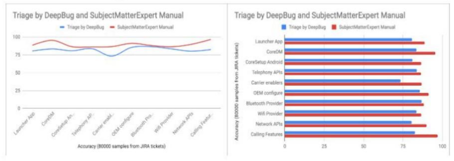
Model

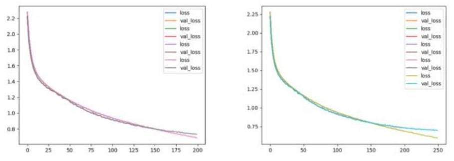
With Keras framework we built and trained two models. 1) the CNN path for logs learning and 2) the RNN with GRU path. We prototyped the CNN model with an AlexNet design, constructed with three convolution layers, get concatenated with a flatten layer, and lastly a softmax for output. In this model the inputs are $20 \times 200 \times 5$ matrices and we applied a padding initially and then conv2D(). For the RNN path we compared the performance for RNN(GRU) and RNN(LSTM) design and RNN(GRU) reached a better performance for cross validation and the average accuracy on test set.



Result

Validated by the industry datasets and our JIRA dataset, DeepBug has achieved an accuracy of avg 86.25% for the top 20 software components classification.



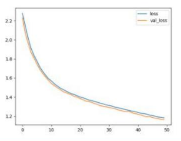


Discussion

1. Our project shows it is more promising to classify for software component instead of developer. In nature there is no distinct correlation in between a developer and the bug fixed by him, instead there's correlation between the software component name and the bugs fixed against it. Base on this intuition we changed the dataset and DeepBug model got an accuracy of 86.25% triaging the top 20 components.

2. Learning from log files. A CNN can learn the bug's log stack in the way mimicking manual log analysis by people, the process works similar to image recognition, such as to extract the features from the logs per layers of the software stack, and abstract at a high level (later cnn layers) to classify to which software component (labeled) it has happened.

3. Dropout. Excessive dropouts might lead to validation loss running below the training loss. We found such an interesting issue and resolved it by decreasing keep-prop and have dropout only for the last layer before softmax dense.



Future Directions

We plan to scale Deepbug for all 300 components in our JIRA system, it could save a huge effort. DeepBug can be improved, especially for the CNN path for log file learnings, and it will be able to generalize for more datasets from industry.

References

[1] Ali Badashian el. al., Crowdsourced bug triaging, 2015
[2] Senthil Mani, el. al,, Exploring the Effectiveness of Deep Learning for Bug Triaging, 2016

Acknowledgements

Huge thanks to Suvadip Paul, and CS230 instructors, Andrew Ng and Kian Katanforoosh, for such a great course.

https://youtu.be/za1P0DAajYA