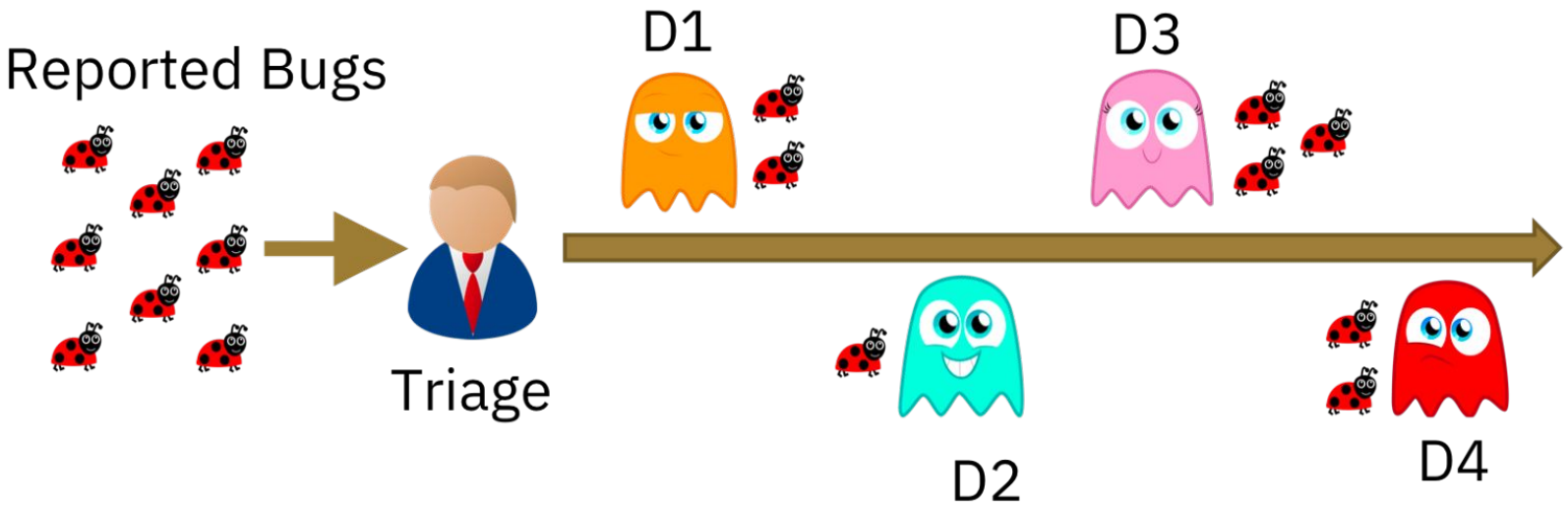
CS559 Deep Learning Project
Final Presentation

Automated Bug Triaging by Using DBRNN-A

Emre Doğan
H. Alperen Çetin

Project Description: Bug Triaging Process



A Sample Bug Report

Labeled bug report: 599892
Fixed by: brettw@chromium.org
Title: GN should only load each import once
Description: In GN mutliple BUILD files can load the same import. GN caches the results of imports so we don't have to load them more than once. But if two BUILD files load the same import at the same time, there is a race. Rather than lock, the code allows each to load the file and the first one finished "wins". This is based on the theory that the race is rare and processing imports is relatively fast. On Windows, many build files end up with the visual_studio_version.gni file which ends up calling build/vs_toolchain.py. This script can be quite slow (slower than the rest of the entire GN run in some cases). The result is that the race is guaranteed to happen for basically every BUILD file that references the .gni file, and we end up running the script many times in parallel (which only slows it down more). We should add the extra locking to resolve the race before loading rather than after.

Dataset

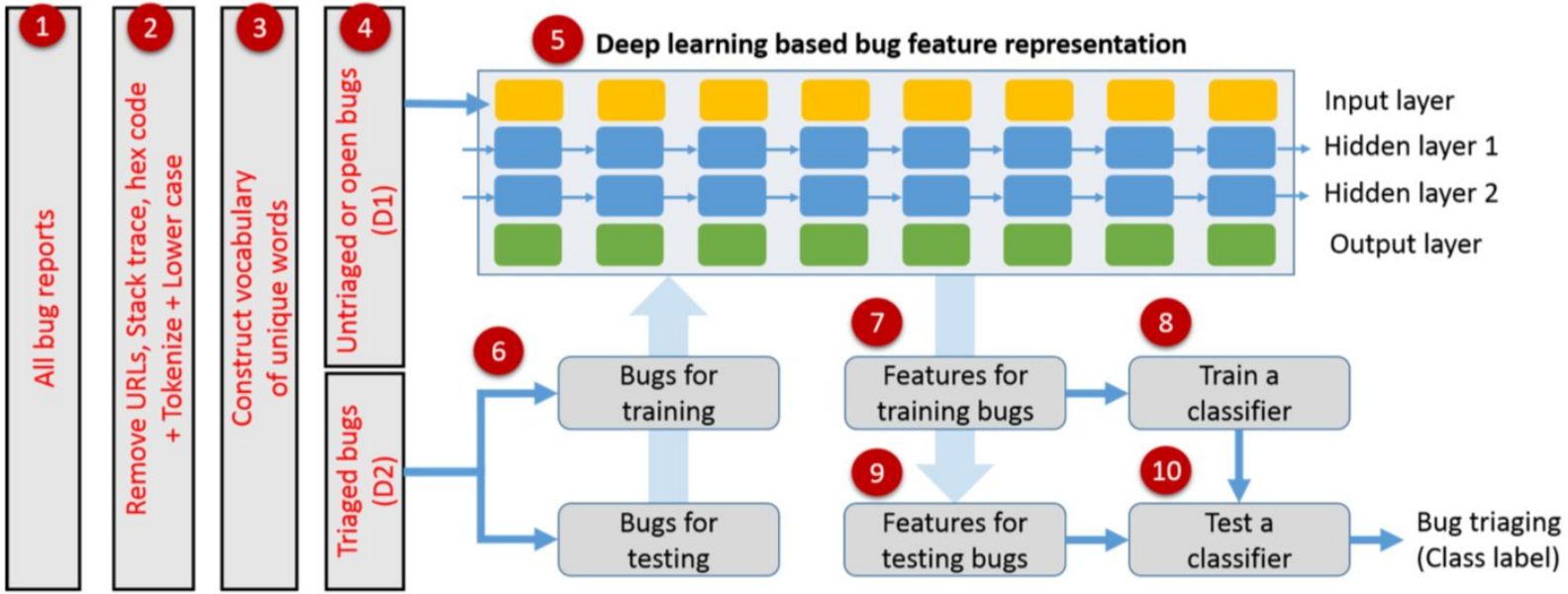
Google Chromium:


```
{
                "id" : 1,
                "issue_id" : 2,
                "issue_title" : "Testing if chromium id works",
                "reported_time" : "2008-08-30 16:00:21",
                "owner" : "",
                "description" : "\nWhat steps will reproduce t
                "status" : "Invalid",
                "type" : "Bug"
}
```

Steps of DeepTriage [1]:

1. Preprocess
2. Word embedding using open bugs
3. Train a Deep Bidirectional RNN with Attention Mechanism Model (DBRNN-A) by using closed bugs and the embedding from 2nd step.
4. Predicting the best developer for the given bug report by using *Softmax Classifier with Categorical Cross Entropy Loss.*
    - Number of class labels ≈ 1000

The overall process [1]



Challenges Faced Up To Now

- Soft Attention Layer Implementation:
  - Problematic implementation.
  - Adapted an existing attention mechanism for our study.

- Uncertain parameters in th paper:
  - Number of nodes in the LSTM layers

Contributions

- Attention layer implementation in Keras
- Using GRU instead of LSTM
- Using embedding of open bugs from all datasets
- Adding an extra dense layer

All of our work is shared in GitHub:

https://github.com/hacetin/deep-triage

Chronological cross validation

Cross-validation for time series

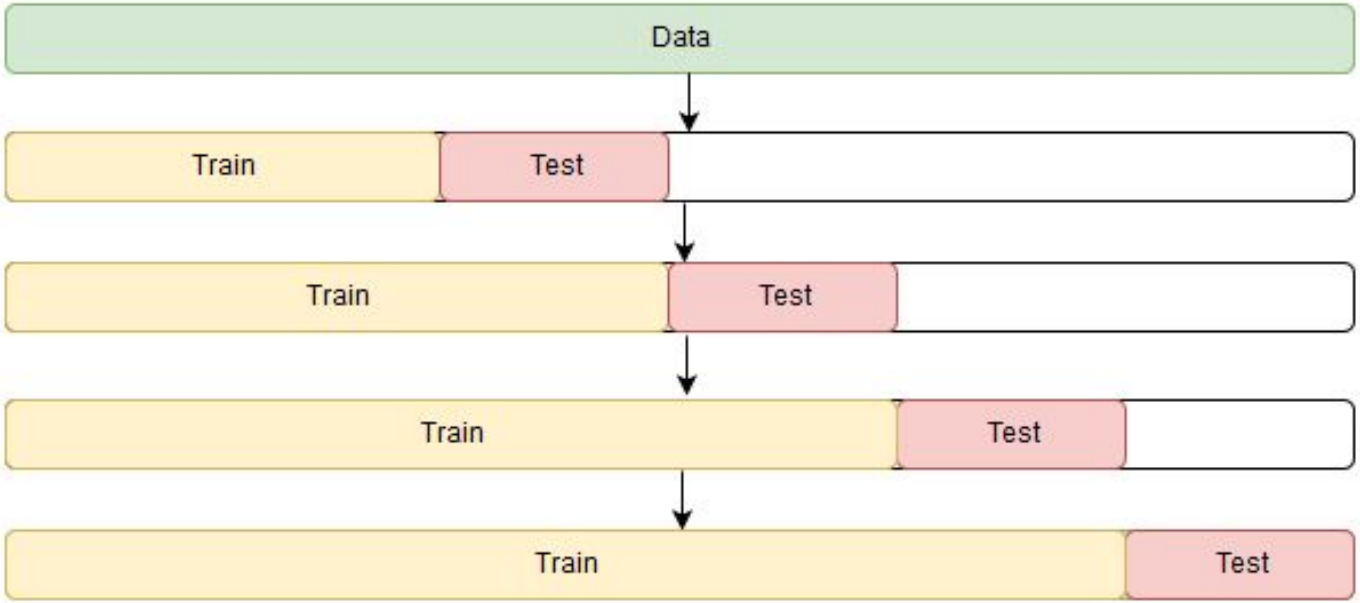


Image: https://stats.stackexchange.com/questions/14099/using-k-fold-cross-validation-for-time-series-model-selection

Using GRU instead of LSTM

Chronological Cross Validation Top-10 Accuracy of Google Chromium

	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10	Average
Paper	36.7	37.4	41.1	42.5	41.8	42.6	44.7	46.8	46.5	47.0	42.7
LSTM	33.4	40.0	41.9	37.3	38.8	41.0	41.3	44.5	47.0	51.0	41.6
GRU	32.4	40.5	41.9	38	38.8	41.6	42.6	44.3	45.8	50.0	41.6

Training time for one epoch reduced up to 60%.

Combining 3 Different Datasets for Corpus

```
"google_chromium", "mozilla_core", "mozilla_firefox"
```

- Not possible to apply transfer learning between different datasets.

Chronological Cross Validation Top-10 Accuracy of Google Chromium

	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10	Average
Paper	36.7	37.4	41.1	42.5	41.8	42.6	44.7	46.8	46.5	47.0	42.7
LSTM	33.4	40.0	41.9	37.3	38.8	41.0	41.3	44.5	47.0	51.0	41.6
LSTM with merged corpus	33.7	41.3	44.3	39.1	40.3	43.1	43.2	47.0	48.9	53.0	43.4

Adding Extra Dense Layer

- Depending on the number of class labels (different for each dataset)

Chronological Cross Validation Top-10 Accuracy of Mozilla Firefox

	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10	Average
Paper	38.9	37.4	39.5	43.9	45.0	47.1	50.5	53.3	54.3	55.8	46.6
1000	38.7	37.6	45.0	43.4	43.0	39.2	50.5	49.0	48.2	46.4	44.1
20y + 10y	37.5	39.6	48.1	43.9	44.6	40.0	52.3	49.8	47.1	46.7	44.9
1000+1000	38.7	36.3	44.4	43.5	44.9	37.3	51.7	51.5	49.6	49.7	44.8

Results

Results for our model for Mozilla Firefox (LSTM, 1000+1000 dense layers and combined corpus)

CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8	CV9	CV10	Average
37.9	32.0	47.5	47.4	45.6	38.8	54.1	51.8	50.8	51.0	45.7

Chronological Cross Validation Top-10 Accuracy of Mozilla Firefox from the paper [1]

BOW + MNB	22.0	22.8	23.6	26.3	29.2	32.3	34.4	36.4	38.6	38.4	30.4 ± 6.2
BOW + Cosine	18.4	21.9	25.1	27.5	29.1	31.4	33.8	35.9	36.7	38.3	29.8 ± 6.3
BOW + SVM	18.7	16.9	15.4	18.2	20.6	19.1	20.3	21.8	22.7	21.9	19.6 ± 2.2
BOW + Softmax	16.5	13.3	13.2	13.8	11.6	12.1	12.3	12.3	12.5	12.9	13.1 ± 1.3
DBRNN-A + Softmax	38.9	37.4	39.5	43.9	45.0	47.1	50.5	53.3	54.3	55.8	**46.6 ± 6.4**

Conclusion

- Working implementation in Keras.
- Using GRU instead of LSTM.
- Using Embedding of open bugs from all datasets.
- Adding extra Dense Layer.
- All work is available on GitHub.

Thank you

References

1. Mani, S., Sankaran, A., & Aralikatte, R. (2019, January). Deeptriage: Exploring the effectiveness of deep learning for bug triaging. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (pp. 171-179). ACM.