Co-Training for Commit Classification

Jian Yi David Lee, Hai Leong Chieu

DSO National Laboratories

Abridged Abstract

- Commits are **noisy user-generated** natural language and code patches that have a **scarce labeled data** issue.
- We apply **co-training**, a semi-supervised learning method, to take advantage of the two views available – the commit message (natural language) and the code changes (programming language) – to improve commit classification.

Commits are useful for ...

- Understanding maintenance activities[1;2;3]
- Studying the impact of social factors in developer teams[4;5]
- Detecting bug-fixing patches[6;7;8]

Scarce Labeled Data

- In 2017, largest public commit dataset[2] had 1,151 commits.
- Linux kernel: 82,300 commits in 2019 and now has > 1M commits. [9]
- Rapid commit creation + variety of commits => difficult to create of large-scale, quality commit datasets
  => scarce labeled data.
- Many prior works were conducted under the fully-supervised setting, a setting that can be a stretch for new categories where labeled data is scarce.

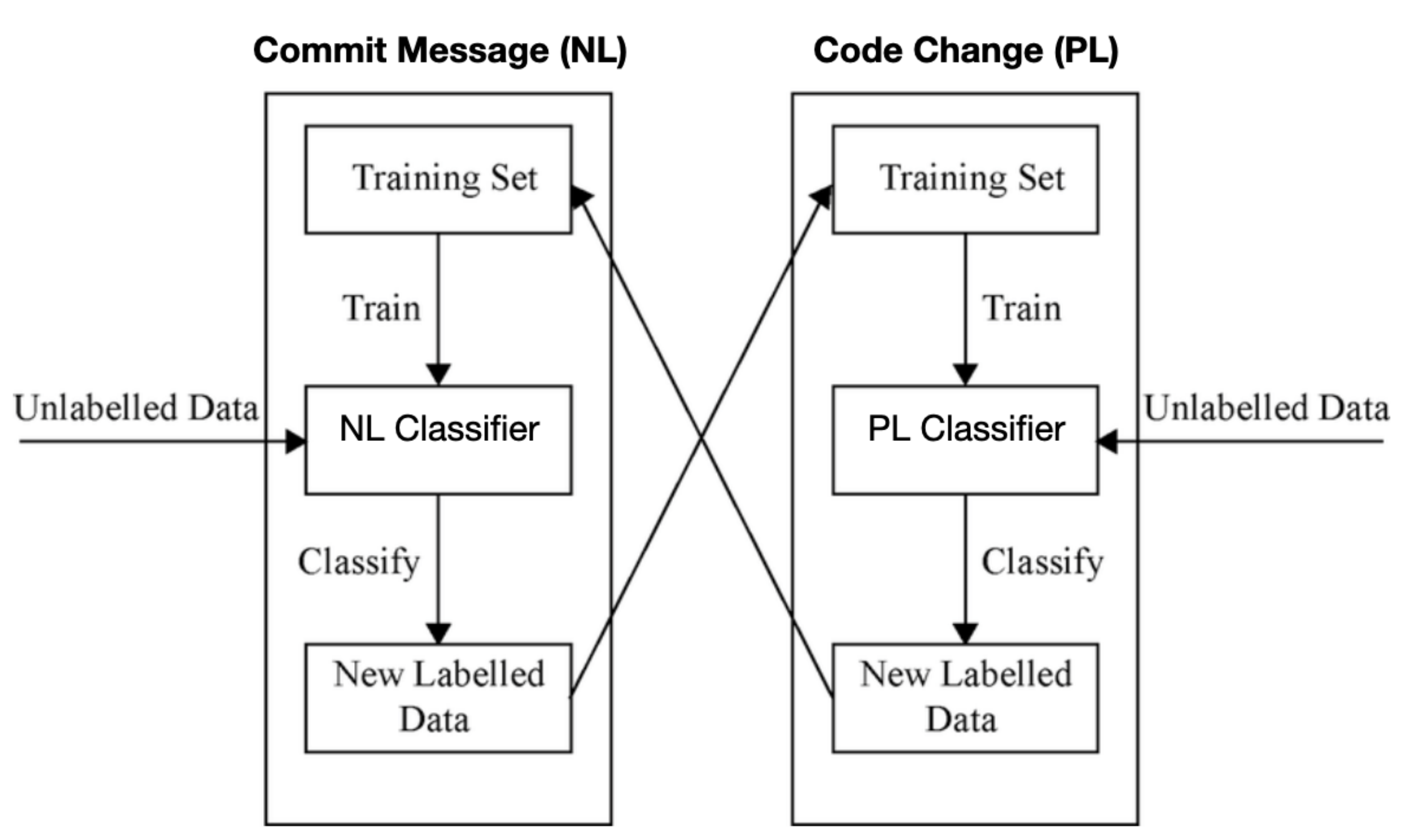
Co-Training's Suitability for Commits

Commit Message (NL view):

```
Improve numerical stability of LayerNorm (#59987)

Summary:
Pull Request resolved: #59987

Similar as GroupNorm, improve numerical stability of
  ↪ LayerNorm by Welford algorithm and pairwise sum.

Test Plan: buck test mode/dev-nosan //caffe2/test:nn -- "
  ↪ LayerNorm"

Reviewed By: ngimel

Differential Revision: D29115235

fbshipit-source-id: 376
  ↪ dac89a4e14bd340aaaf169fef8d0d4ca4a1c4
```

Code Change (PL view, cropped with ellipsis)

```
diff --git a/aten/src/ATen/native/cpu/layer_norm_kernel.
  ↪ cpp b/aten/src/ATen/native/cpu/layer_norm_kernel.cpp
index 95a35571646d..366afe64b72a 100644
--- a/aten/src/ATen/native/cpu/layer_norm_kernel.cpp
+++ b/aten/src/ATen/native/cpu/layer_norm_kernel.cpp
@@ -1,13 +1,14 @@
 #include <cmath>
+#include <tuple>
 ...
 namespace at {
 namespace native {
@@ -29,30 +30,21 @@ void LayerNormKernelImplInternal(
   DCHECK_EQ(X.numel(), M * N);
   DCHECK(!gamma.defined() || gamma.numel() == N);
   DCHECK(!beta.defined() || beta.numel() == N);
-  T* X_data = X.data_ptr<T>();
+  const T* X_data = X.data_ptr<T>();
...
```



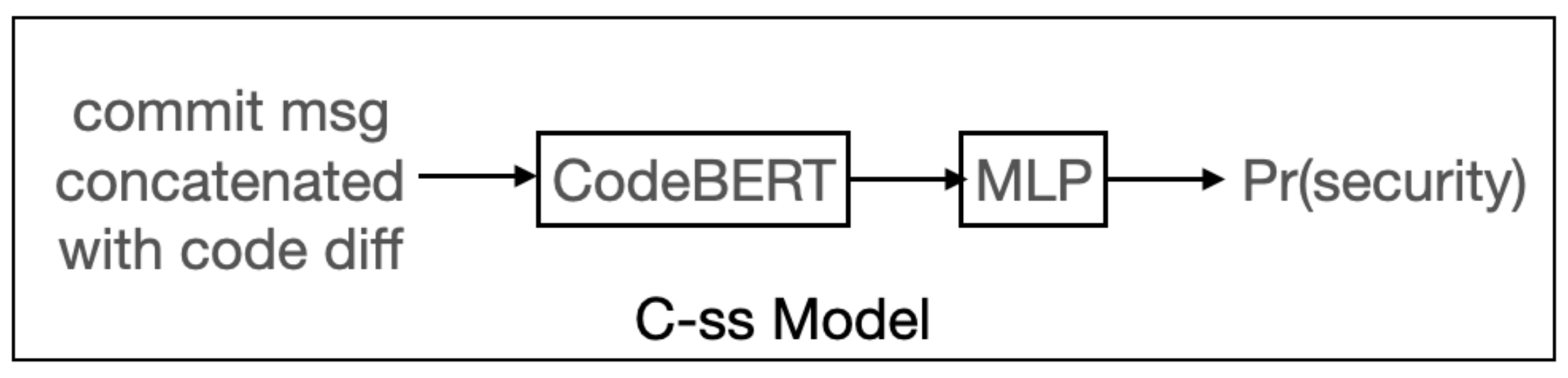
Dataset

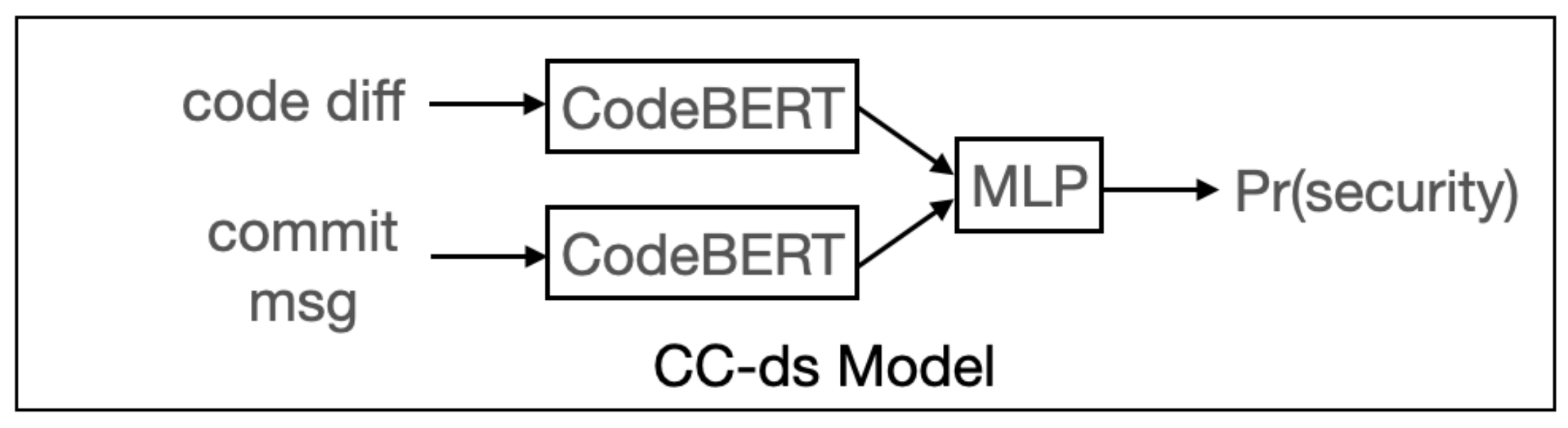
- Create a balanced dataset from Reis and Abreu (2021)[10].
- 3,765 positive samples & randomly sample 2x as many negative samples from the 910 repositories that appear in both positive & negative classes.
  => Total approx. 10,000 commits. (**900Repo** dataset)

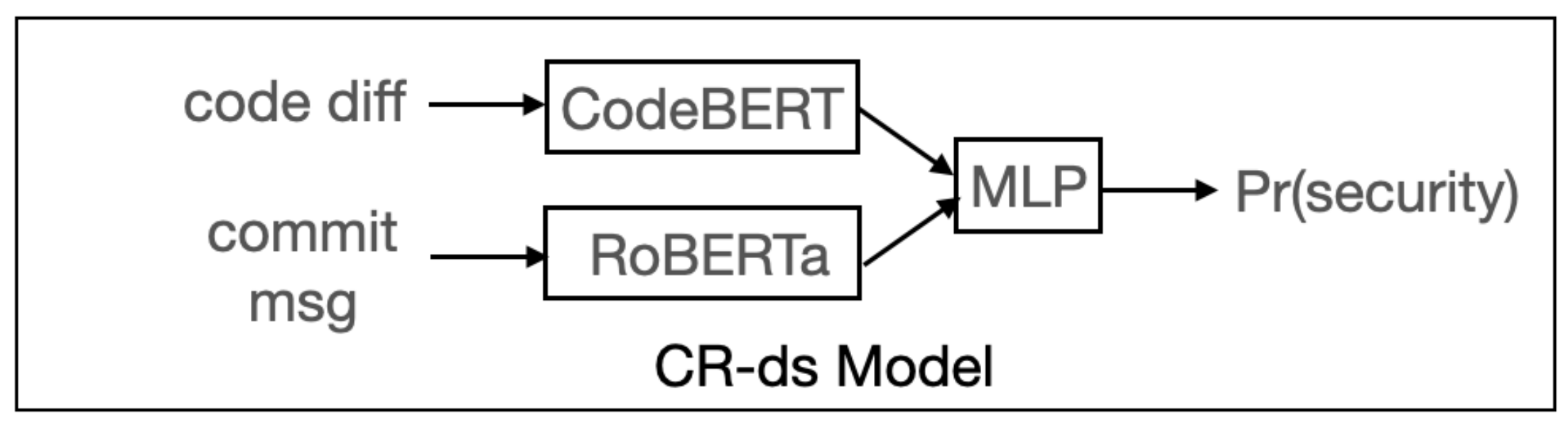
Model Options

RQ1: Can attention between NL and PL improve classification?

RQ2: Is CodeBERT better than RoBERTa at representing commit messages for commit classification?







Model	P	R	F1	AUC	Acc.
C-ss	80.3	80.4	80.4	81.0	80.4
CC-ds	83.0	83.0	83.0	84.9	83.0
CR-ds	**84.0**	**84.1**	**83.9**	**87.4**	**84.1**

Table 1: Test results with different architectures. P/R/F1: average weighted precision, recall and F1 (weighted by class-size), Acc.: accuracy, AUC: area under precision-recall curve.

Co-Training

RQ3: Can co-training improve commit classification?

NL: RoBERTa-MLP classifier on commit messages.
PL: CodeBERT-MLP classifier on code changes.

- To simulate resource-poor conditions, we use either 0.5%, 1%, 2% or 4% of the 900Repo dataset as labeled training data, validation set same size, & 15% as test set. The remaining is unlabeled data for co-training.
- Results in Table 2.
- Improvement across virtually all performance metrics in all scenarios.
- In all scenarios, the PL classifier (CodeBERT-MLP) improved more than the NL classifier (RoBERTa-MLP).

T	Model	P	R	F1	Acc.
0.5%	NL-T	67.8	68.8	**66.7**	68.8
	PL-T	61.2	63.0	61.4	63.0
	NL	**68.0**	**69.0**	**66.7**	**69.0**
	PL	64.7	66.3	63.4	66.3
1%	NL-T	58.5	62.3	57.2	62.3
	PL-T	65.2	65.3	65.3	65.3
	NL	**73.0**	**72.2**	**72.5**	**72.2**
	PL	68.5	67.3	60.9	67.3
2%	NL-T	75.9	76.3	76.0	76.3
	PL-T	68.4	65.9	66.5	65.9
	NL	**76.2**	**76.5**	**76.3**	**76.5**
	PL	72.4	73.1	72.4	73.1
4%	NL-T	77.6	77.0	76.9	77.0
	PL-T	71.2	69.3	69.7	69.3
	NL	**77.7**	**77.9**	**77.8**	**77.9**
	PL	73.4	70.6	70.9	70.6

Table 2: Co-training experimental results. T: labeled training data as a percentage of the dataset. NL-T, PL-T: supervised models trained only on T. NL, PL: co-trained models. P/R/F1: average weighted precision, recall, and F1 score. Acc.: accuracy.

- Under both supervised & co-training, NL classifier performs better than PL classifier. Maybe due to challenging nature of obtaining good representations for programming languages as well as the length of code changes frequently being longer than the commit message and CodeBERT's 512 token limit.

Conclusion

- Co-training useful in scarce labeled data setting.

Further research directions:

- Semi-supervised learning for semantic search in repositories.
- Apply models that allow longer sequences as input, e.g., the Longformer[11].

References

[1] Hindle, A, et al.: in 2009 IEEE 17th International Conference on Program Comprehension, IEEE 2009 30–39.

[2] Levin, S, et al.: in Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering 2017 97–106.

[3] Hönel, S, et al.: in 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), IEEE 2019 109–120.

[4] Soto, M, et al.: in 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), IEEE 2017 483–486.

[5] Vasilescu, B, et al.: in 2015 IEEE/ACM 12th working conference on mining software repositories, IEEE 2015 514–517.

[6] Tian, Y, et al.: in 2012 34th international conference on software engineering (ICSE), IEEE 2012 386–396.

[7] Casalnuovo, C, et al.: in Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis 2017 396–399.

[8] Zafar, S, et al.: in 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE 2019 1–6.

[9] Foundation, TL: -, journal 2021.

[10] Reis, SO, et al.: *arXiv*, journal 2021.

[11] Beltagy, I, et al.: *arXiv preprint arXiv:200405150*, journal 2020.