Exploring transformers for multi-label classification of Java vulnerabilities

Cláudia Mamede, Eduard Pinconschi, Rui Abreu, José Campos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

22nd IEEE International Conference on Software Quality, Reliability, and Security

December, 2022 • Guangzhou, China

Society is becoming **more dependent on technology**

Companies must develop code ASAP without compromising security

How can we do that?

Static analysis

Dynamic analysis

Pattern matching

(...)

- Too time-consuming
- High False Positive rate
- Applied by experts

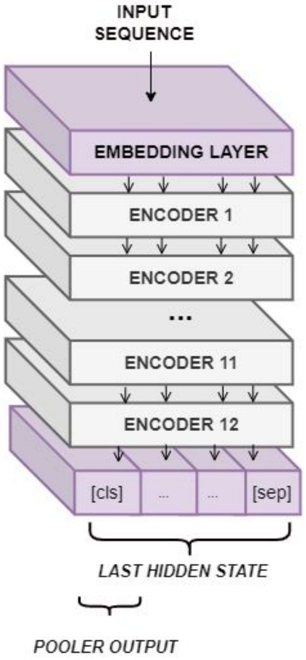
Machine learning

TRANSFORMERS

- Handle long range dependencies
- Transfer learning mechanism
- SOTA results for vulnerability detection

JavaBERT

CodeBERT



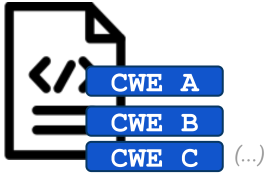
What about the data ?

- Using synthetic samples from the Juliet Test Suite
- Function-level granularity
- 113 898 methods (70% non-vulnerable; 30% vulnerable)
- 20 CWES

MULTILABEL: WHY?

Research focus on **binary** or **multi-class** classification:

Binary classification



Multi class classification

Both classifications lack information. For example:

```
public void action(byte data) throws Throwable {

    /* POTENTIAL FLAW: data == Byte.MAX_VALUE */
    byte result = (byte)(++data);

    IO.writeLine("result: " + result);
}
```

Binary: UNSAFE

What problem will you solve?

Multi-class: CWE-190

Is the code truly vulnerable?

Multi-label: UNSAFE CWE-190

Multilabel helps developers speed up software development by helping them in decision-making.

```
public void action(byte data) throws Throwable {
    /* POTENTIAL FLAW: data == Byte.MAX_VALUE */
    byte result = (byte)(++data);

    IO.writeLine("result: " + result);
}

/* (...) */
/* Restriction on the value of data */
    if( data < Byte.MAX_VALUE) {
        action(data);
    }
```

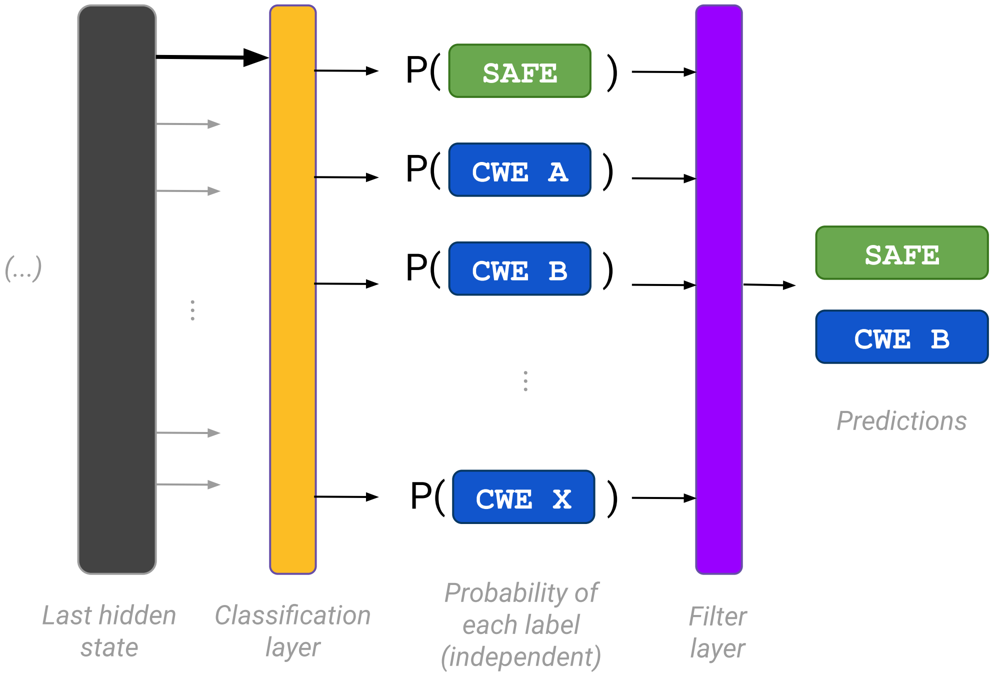
Multi-label: SAFE CWE-190

- **Threshold selection** to solve the problem of multi-label [1]
- **Threshold = 0.5** to filter the labels
- Theoretically, we can discover ≥ 1 CWE

  (no data to properly test this theory)

MULTILABEL: HOW?

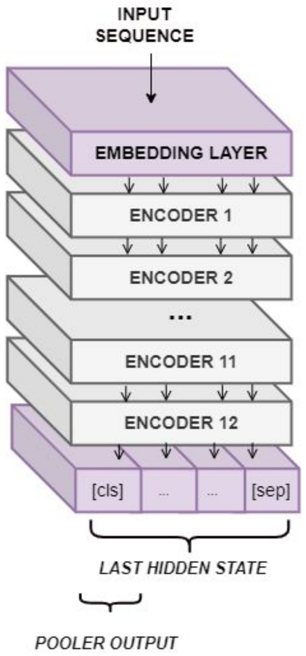
For example:

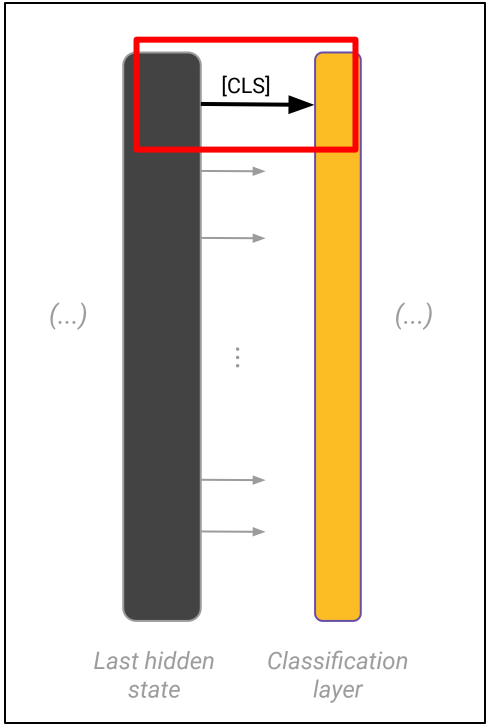


[2] H. Fallah *et al.* Adapting transformers for multi-label text classification. In *CIRCLE'22*

RQ 1 | How do different output configurations impact the learning of BERT-based models?

BERT-based model architecture



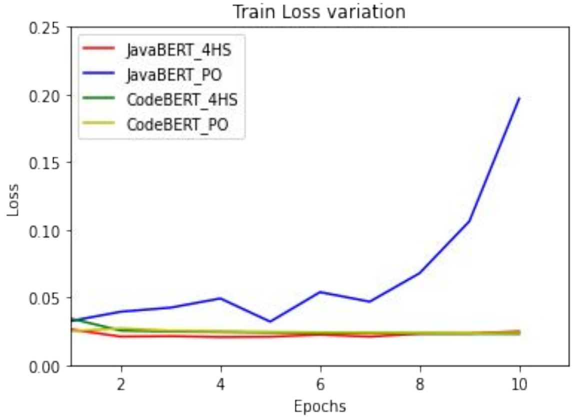


How to improve the representation of the [CLS] token?

Configurations:

- Pooler output (default in most libraries) 🤗
- Concatenation of the [CLS] tokens of the last 4 hidden states

RQ 1 | How do different output configurations impact the learning of BERT-based models?



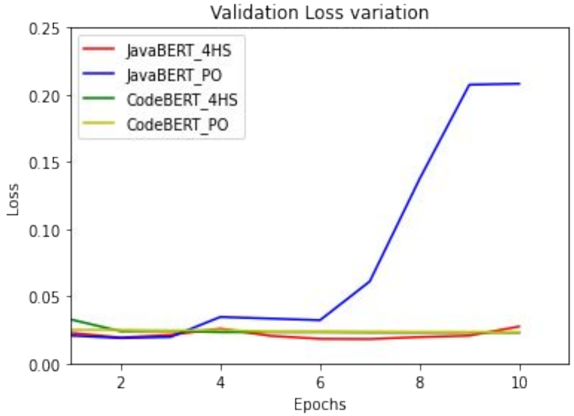


Figure | Learning curves (loss variations) during training (left) and validation (right) for all models.

Finding 1

The pooler output configuration compromises the transfer learning capabilities of JavaBERT.

RQ 2 | Which BERT-based model configuration achieves better vulnerability identifications?

Model	#Epoch	Accuracy	$\overline{wF1}$	$\overline{wPrecision}$	$\overline{wRecall}$	$\overline{FNR}$	$\overline{FPR}$
JavaBERT_4HS	8	98.90%	94.0%	95.0%	93.0%	7.12%	0.98%
CodeBERT_4HS	10	98.68%	93.0%	95.0%	91.0%	12.28%	1.02%
CodeBERT_PO	9	98.67%	93.0%	95.0%	91.0%	12.39%	1.06%

Table | Performance results for JavaBERT and CodeBERT with different model configurations

Finding 2

Combining the outputs of the last four hidden layers yields more accurate predictions.

RQ 3 | To what extent does implicit bias in datasets affect the ability of the model to learn?

- Find **problematic tokens** in datasets
- **Our hypothesis:**
  Problematic tokens are most likely over represented, causing the model to make wrong predictions.

POINTWISE MUTUAL INFORMATION

Class	Token	PMI
Unsafe or vulnerable	##ad	0.98
	bad	0.88
Safe or non-vulnerable	good	1
	##BS	1
CWE-15	##15	1
CWE-23	##23	1

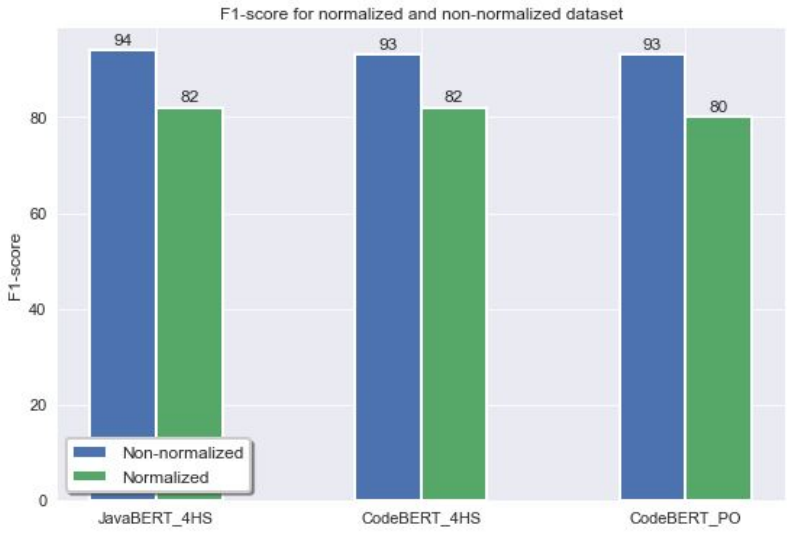
Table | Top PMI scores for some labels

RQ 3 | To what extent does implicit bias in datasets affect the ability of the model to learn?

Original dataset normalized method and variable names

Missing: exceptions, some method calls and global variables

Replace problematic tokens with random strings (of the same size). Repeat training.



Findings 3 and 4

We can use the Pointwise Mutual Score (PMI) to identify problematic tokens in code.

Removing token that bias the model substantially reduces the f1 score (up to 12%).

RQ 4 | How do BERT-based models perform when exposed to real-world samples?

Synthetic data

- Similar to real-world data
- Programmatically generated
- Similar style and structure

  (particularly single-sourced synthetic samples)

Real-world data

- (Usually) no rules for naming vars/methods
- (Usually) no particular code structure

Original real-world dataset from T. Le *et al.* [2]

CVE → CWE

Filter the samples our models can identify (by CWE)

Final test set: **70 vulnerable samples** (targeting only **8 known CWEs**)

[2] T. Le and M. Babar. On the use of fine-grained vulnerable code statements for software vulnerability assessment models. In *MSR'22*

RQ 4 | How do BERT-based models perform when exposed to real-world samples?

Model	Accuracy	$\overline{wF1}$	$\overline{FNR}$	$\overline{FPR}$
JavaBERT_4HS	**90.06%** (-8.87%)	**44.0%** (-50%)	**36.03%** (+28.91%)	**4.12%** (+3.14%)
CodeBERT_4HS	**86.88%** (-11.8%)	**23.0%** (-70%)	**37.74%** (+24.46%)	**5.39%** (+4.37%)
CodeBERT_PO	**85.86%** (-12.81%)	**20.0%** (-73%)	**39.52%** (+27.13%)	**9.85%** (+8.79%)

Table | Performance results for models tested with real-world samples

Finding 5

Models trained on synthetic data have a tendency to identify true vulnerable samples as non-vulnerable.

RQ 5 | To what extent BERT-based models can predict unknown vulnerabilities?

“ (...) **generalizability** measures how applicable the results of a study are to a broader group.

In this context, a model is said to have good generalizability if it can be successfully applied to **identify unknown flaws.**”

Test with samples of unknown vulnerability types that are **related** to the ones the models know.

Software Fault Patterns

Test with samples of unknown vulnerability types that are **unrelated** to the ones the models know.

RQ 5 | To what extent BERT-based models can predict unknown vulnerabilities?

SFP Secondary Cluster	CWE	# samples (training set)
Glitch in computation	190, 191, 369, 197	12 020
Tainted input to command	89, 113, 134, 80, 78, 643, 90	6 569
Tainted input to variable	606, 15	960
Path traversal	23, 36	554

Not listed in the SFP view: 129, 789 and 690
1-to-1 mapping: 400, 470 and 319

Test set for unknown and related CWEs

1. CWE-611 and CWE-79 `ds_611_79`
   (Tainted Input to Command; 239 samples)
2. CWE-22 `ds_22`
   (Path Traversal; 179 samples)

Test set for unknown and unrelated CWEs

1. CWE-287 `ds_287`
   (Authentication Bypass; 159 samples)

RQ 5 | To what extent BERT-based models can predict unknown vulnerabilities?

"Vulnerable" class:

Model	Dataset	Accuracy	F1	FNR	FPR
JavaBERT_4hs	**ds_611_79**	**74.47%**	**85.37%**	**25.52%**	**0%**
	ds_22	55.86%	71.69%	44.13%	0%
	ds_287	38.36%	55.45%	61.63%	0%
CodeBERT_4hs (CodeBERT_PO behaves similarly)	ds_611_79	17.57%	29.89%	82.4%	0%
	ds_22	12.85%	22.77%	87.15%	0%
	ds_287	4.40%	8.43%	95.60%	0%

Findings 6 and 7

JavaBERT fine-tuned on synthetic data can successfully predict unknown and relatable vulnerabilities

BERT-based models fine-tuned on synthetic data cannot predict unknown and unrelated vulnerabilities.

THREATS TO VALIDITY

- Small and imbalanced datasets
- Not using cross-validation
- No preprocessing strategies (e.g.: sampling)
- Lack information regarding flaw location

FUTURE IMPROVEMENTS

- **More data (!!)**
  - Train with synthetic & real-world samples
  - Include non-vulnerable samples in the test set
  - Balance the dataset
- Explore the models' **ability to discover ≥ 1 CWE**

Code & datasets available: **https://github.com/TQRG/VDET-for-Java**