Neural-Augmented Static Analysis of Android Communication

Jinman Zhao, Aws Albarghouthi, Vaibhav Rastogi, Somesh Jha, Damien Octeau
University of Wisconsin-Madison, Google
Key Idea

Use machine learning to refine results from static analysis.
Static Analysis: False Positives

Program & Property

Static Analyzer

Must True

Unsure...

Must False

False Positives

Ranking problem
Machine Learning to Augment

Program & Property

Static Analyzer

Must True  Likelihood $\in [0, 1]$  Must False

Train

Model

Predict
Link Inference for Android Communication

- Inter-Component Communication Links
- Static Analyzer
- Must True Links
- May Links
- Must False Links

Train Model
Predict
Task

Link Inference in Android Communication
Android ICC: A User’s Experience

Intent

Restaurant
1234 Alice St.
Orlando, FL
Send a message

I’d like to make a reservation ...

Inter-Component Communication

Component w/ Filter

Malicious APP
Android ICC: An Example

Intent

```java
public void sendImplicitIntent() {
    Intent intent = new Intent();
    intent.setAction("SEND");
    msg = ... // contains phone # and msg
    intent.setData(msg);
    startActivity(intent);
}
```

Code View

Filter

```xml
<intent-filter>
    <action android:name="SEND"/>
    <action android:name="VIEW"/>
    <data android:scheme="sms"/>
    <category android:name="DEFAULT"/>
</intent-filter>
```

Intent filter for a sms component

(part of) the resolution logic

ICC link?

Yes!
\[\text{match}(i, f) = \text{type}(i, f) \land \text{visibility}(i, f) \land \text{perm}(i, f) \land (\text{explicit}(i, f) \lor \text{implicit}(i, f)).\]

\[\text{type}(i, f) = i_{\text{type}} \subseteq f_{\text{type}}\]

\[\text{visibility}(i, f) = i_{\text{app.name}} \subseteq f_{\text{app.name}} \lor f_{\text{exported}} \subseteq \{\text{true}\}\]

\[\text{perm}(i, f) = i_{\text{perm}} \subseteq f_{\text{uses.perm}} \land f_{\text{perm}} \subseteq i_{\text{uses.perm}}\]

\[\text{explicit}(i, f) = i_{\text{target.comp}} \neq \emptyset \land i_{\text{target.app}} \subseteq f_{\text{app.name}} \land i_{\text{target.comp}} \subseteq f_{\text{comp.name}}\]

\[\text{implicit}(i, f) = i_{\text{target.comp}} = \emptyset \land i_{\text{action}} \subseteq f_{\text{actions}} \land i_{\text{category}} \subseteq f_{\text{categories}} \land \text{data}(i, f),\]

(Bigger part of) the resolution logic
(Octeau et al., POPL’16)
Previous Work: PRIMO

- PRIMO (Octeau et al., POPL’16) uses a hand-crafted probabilistic model that assigns probabilities to ICC links inferred by static analysis.
  - Laborious, error-prone and requiring expert domain knowledge.
  - Difficulty catching up with constantly evolving Android system.
Questions
How can we triage many links with minimal expert domain knowledge?

Neural networks.
How can we process inputs of complex data types in a systematic way?

Type-directed encoder.
How do our models perform?

Very good!
Are the models learning the right things?

Seems like so.
We are not trying to...

- Propose new NN module
- Eliminate use of domain knowledge
- Rule out manual effort

We are trying to...

- Propose systematic way to construct NN
- Provide decent performance without expert knowledge
- Use less labour with more automation
Approach

Part 1

How can we triage may links with minimal expert domain knowledge?
Link-Inference Neural Network

LINN: An end-to-end encoder-and-classifier architecture.
Approach

Part 2

How can we process inputs of complex data types in a systematic way?
Model

Classifier

[0,1]

Intent

Encoder

Encoder

Filter
Type-Directed Encoder

TDE: mapping type signature to neural network architecture.
An example: Encoding Product Types

Instance $t := (a, b)$
Type $T := \text{tuple}(A, B)$

$\lambda(x, y). \text{comb}(g_1(x), g_2(y)) \triangleright \tau_1 \times \tau_2$
Rules for type-directed encoding

\[ \lambda x. x \rightarrow \mathbb{R} \]

\[ \text{enumEnc} : \tau_c \rightarrow \mathbb{R}^n \]

\[ \text{aggr} : S(\mathbb{R}^n) \rightarrow \mathbb{R}^m \]

\[ \lambda x. \text{aggr}(\text{map} \ g \ x) \rightarrow S(\tau) \]

\[ g : \tau \rightarrow \mathbb{R}^n \]

\[ \text{flat} : L(\mathbb{R}^n) \rightarrow \mathbb{R}^m \]

\[ \lambda x. \text{flat}(\text{map} \ g \ x) \rightarrow L(\tau) \]

\[ \text{comb} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^l \]

\[ \lambda (x, y). \text{comb}(g_1(x), g_2(y)) \rightarrow \tau_1 \times \tau_2 \]

\[ g_1 : \tau_1 \rightarrow \mathbb{R}^n \]

\[ g_2 : \tau_2 \rightarrow \mathbb{R}^m \]

\[ \lambda x. \text{if} \ x \in \tau_1 \text{ then } g_1(x) \text{ else } g_2(x) \rightarrow \tau_1 + \tau_2 \]

\[ \tau_c \text{ is a categorical type, e.g., characters.} \]
Android ICC: Our Abstraction

Type signatures

**Intent** `intent := tuple(act, cats)`

**Action** `act := optional(string)`

**Categories** `cats := set(string)`

**Filter** `filter := tuple(acts, cats)`

**Actions** `acts := set(string)`

**Categories** `cats := set(string)`
Type-Directed Encoder

Type signature

intent
  └── tuple
    └── act
        └── optional
            └── string
                └── list
                    └── char
    └── cats
        └── set
            └── string
                └── list
                    └── char

Neural network template

intent-en
  └── comb
    └── act-en
        └── union
            └── str-en
                └── flat
                    └── char-en
                        └── enum
                            └── char
    └── cats-en
        └── aggr
            └── str-en
                └── flat
                    └── char-en
                        └── enum
                            └── char
Type-Directed Encoder: Instantiation

Neural network template

Neural network (typed-tree)
Type-Directed Encoder: Instantiation

Neural network template

Neural network (str-rnn)
A systematic way to build and explore structured NN.
Experiments

Are our models correctly predicting links?
Setup

- Dataset of 10,500 Android APPs from Google Play.
- IC3 (Octeau et al., ICSE’15) for static analysis.
- PRIMO’s abstract matching for may/must partition.
- Simulated ground truth for may links.
- 4 instantiations of the TDE architecture.

<table>
<thead>
<tr>
<th></th>
<th># pairs</th>
<th># positive</th>
<th># negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>training set</td>
<td>105,108</td>
<td>63,168</td>
<td>41,940</td>
</tr>
<tr>
<td>testing set</td>
<td>43,680</td>
<td>29,260</td>
<td>14,420</td>
</tr>
</tbody>
</table>
All instantiated models perform as good as PRIMO.
Our best model (typed-tree) fills the correlation gap by 72% compared to PRIMO despite the harder setting.
More Results for Our Best Model

ROC (left) and the distribution of predicted likelihood (right) from typed-tree model.
Interpretability

How do we know the model is learning the right thing?
Sensitivity to Masking

Picking distinctive values

Ignoring less useful parts
Learned Encodings

Semantically closer values receive more similar encodings.

Visualized by t-SNE.
Conclusion

- Neural-augmented static analysis
- Type-directed encoder
- Increased accuracy with less domain knowledge
- Interpretability study
Future Works

- Apply to other analysis tasks
- Push machine learning into static analysis procedure
Thanks for listening!

Q & A