# Neural-Augmented Static Analysis of Android Communication

Jinman Zhao* Aws Albarghouthi* Vaibhav Rastogi* Somesh Jha* Damien Octeau†
*University of Wisconsin-Madison †Google
{jz,aws,jha}@cs.wisc.edu vrastogi@wisc.edu docteau@google.com

## Problem

Static analysis of Android-specific message-passing system:
- (*link inference*) Can component *c* communicate with component *d*? (*c* and *d* have a link?)

## Approach

- A machine learning model to automatically learn from "must" links and to predict for "may" links.
- An end-to-end encoder-and-classifier architecture: link inference neural network (LINN) -- requires minimal domain knowledge.
- Type-directed encoder (TDE): schematically composing encoder out of constituent data type definition -- general and flexible.

**Figure 1. Overview of our approach**

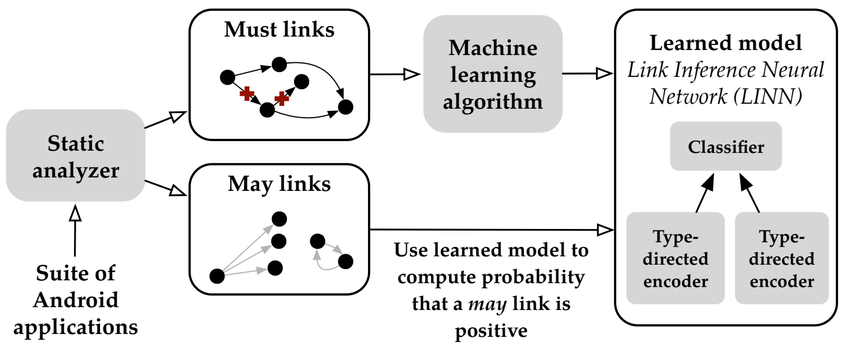



**Figure 4: Rules for type-directed encoders (TDEs)**

$$\frac{}{\lambda x.x \blacktriangleright \mathrm{R}}\ \text{e-real} \qquad \frac{enumEnc : \tau_c \to \mathrm{R}^n}{enumEnc \blacktriangleright \tau_c}\ \text{e-cat}$$

$$\frac{\partial : \tau \to \mathrm{R}^n \quad flat : \mathrm{L}(\mathrm{R}^n) \to \mathrm{R}^m}{\lambda x.\, flat(map\ \partial\ x) \blacktriangleright \mathrm{L}(\tau)}\ \text{e-list} \qquad \frac{\partial : \tau \to \mathrm{R}^n \quad aggr : \mathrm{S}(\mathrm{R}^n) \to \mathrm{R}^m}{\lambda x.\, aggr(map\ \partial\ x) \blacktriangleright \mathrm{S}(\tau)}\ \text{e-set}$$

$$\frac{\partial 1 : \tau 1 \to \mathrm{R}^n \quad \partial 2 : \tau 2 \to \mathrm{R}^m \quad comb : \mathrm{R}^n \times \mathrm{R}^m \to \mathrm{R}^l}{\lambda (x, y).\, comb(\partial 1(x), \partial 2(y)) \blacktriangleright \tau 1 \times \tau 2}\ \text{e-prod}$$

$$\frac{\partial 1 : \tau 1 \to \mathrm{R}^n \quad \partial 2 : \tau 2 \to \mathrm{R}^n}{\lambda x.\, if\ x \in \tau 1\ then\ \partial 1(x)\ else\ \partial 2(x) \blacktriangleright \tau 1 + \tau 2}\ \text{e-sum}$$

$\tau_c$ is a categorical type, e.g., characters.

**Table 1: Types of differentiable functions used in TDEs and possible practical instantiations**

| Encoder | Type | Possible differentiable implementations |
|---|---|---|
| *enumEnc* | $\Sigma \to \mathrm{R}^l$ | Trainable lookup table (*embedding layer*) |
| *flat* | $\mathrm{L}(\mathrm{R}^n) \to \mathrm{R}^m$ | cnn / lstm |
| *aggr* | $\mathrm{S}(\mathrm{R}^n) \to \mathrm{R}^m$ | *sum* / Child-sum Tree-lstm unit |
| *comb* | $\mathrm{R}^n \times \mathrm{R}^m \to \mathrm{R}^l$ | Single-layer mlp / binary Tree-lstm unit |

## Related works

- Android analysis: Formal modeling for ICC process. Hard coded domain knowledge for triaging may links.
- Static analysis alarms: Counting based, more sophisticated statistics, probabilistic information-flow specifications. Manual labeling of small amount of data.
- Machine learning for programs: Learning from "big code". Tree structured neural nets for abstract syntax trees (AST) and expressions.

## Background: Android ICC

- Android applications are conceptually collections of *components*.
- Applications can leverage the functionality of other applications through a sophisticated message-passing system: *Inter-Component Communication* (ICC).
- any inter-component or inter-application program analysis must first begin by computing the ICC links.

**Figure 2a. ICC example with three applications**

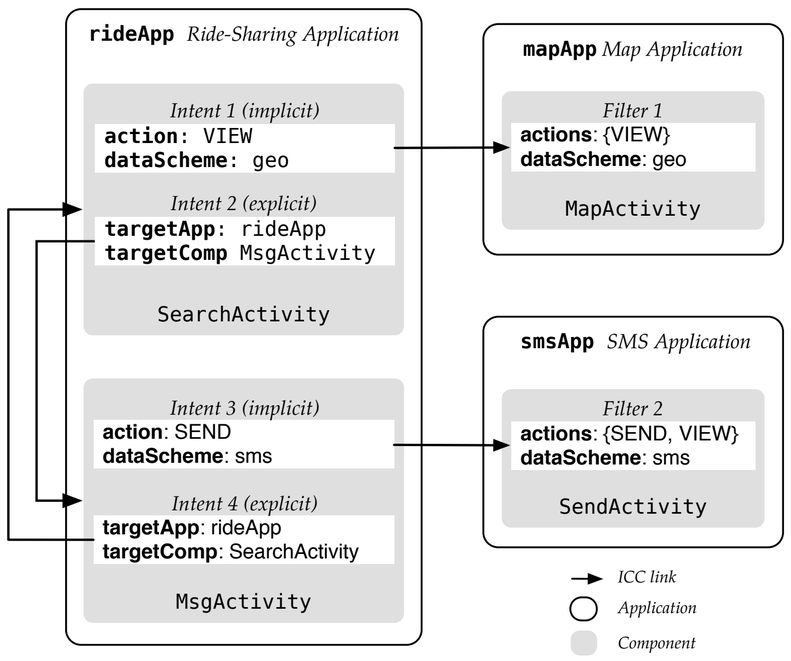



**Figure 2b. Intent for sending an SMS and associated filter**

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

*Code constructing and starting implicit intent*

```
<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*

**Figure 2c. Our abstractions**

Intent *intent := tuple(act, cats)*
Action *act := string*
Categories *cats := set(string)*

Filter *filter := tuple(acts, cats)*
Actions *acts := set(string)*
Categories *cats := set(string)*

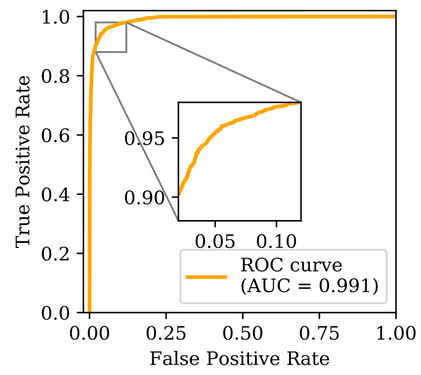
Linkage *link : (intent, filter) ↦ {0, 1, T}*
Classifier $\hat{p}$ *: (intent, filter) ↦ [0, 1]*

## Experiments & Results

- Dataset of 10,500 Android applications from Google Play. See right for Training / testing details.
- Simulated ground truth for may links induced by empirical distribution of imprecisions.
- 4 instantiation of the architecture and TDE.
- Measuring inference time for efficiency; Spearman's $\gamma$ for correlation; F1 score and area under ROC for accuracy and recall; entropy of predicted probabilities $\hat{y}$ for the ability of triaging links; true positive rate (TPR) within $\hat{y} > 0.95$ for the efficiency of manual inspection of highly ranked links .
- All instantiations are good at predicting linkages.
- str-cnn is the smallest and the fastest; typed-tree is the best at predicting links given slightly more resources.

| | # pairs | # positive | # negative |
|---|---|---|---|
| training set | 105,108 | 63,168 | 41,940 |
| testing set | 43,680 | 29,260 | 14,420 |

**Figure 5: Detailed results for the typed-tree instantiation**



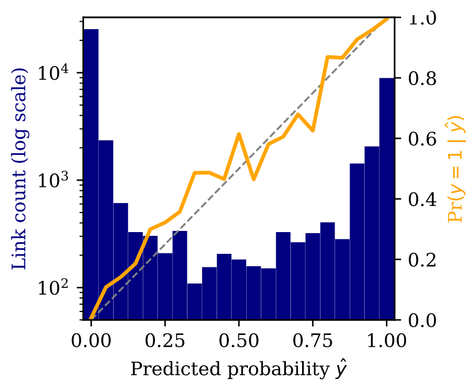




**Table 2: Instantiations of TDE parameters**

| Instantiation | Type | TDE parameters *enumEnc* | *flat* | *aggr* | *comb* |
|---|---|---|---|---|---|
| str-rnn | L(Σ) | *lookup* | rnn | - | - |
| str-cnn | L(Σ) | *lookup* | cnn | - | - |
| typed-simple | full | *lookup* | cnn | *sum* | 1-layer perceptron |
| typed-tree | full | *lookup* | cnn | tree-lstm | tree-lstm |

**Table 4: Summary of model evaluati**

| Instantiation | # parameters | Inference time (μs/link) | Testing $\gamma$ | Testing F1 | Area under ROC | Entropy of $\hat{y}$ | TPR within $\hat{y} > 0.95$ |
|---|---|---|---|---|---|---|---|
| str-rnn | 154,657 | 2220 | 0.970 | 0.891 | 0.975 | 3.002 | 0.980 |
| str-cnn | **27,409** | **57** | 0.988 | 0.917 | 0.988 | 2.534 | **0.998** |
| typed-simple | 142,417 | 157 | 0.989 | 0.920 | 0.988 | 2.399 | 0.996 |
| typed-tree | 634,881 | 171 | **0.992** | **0.931** | **0.991** | **2.220** | 0.994 |

## Interpretability

- Kernel activations: single kernel detects useful patterns but also my not be so precise.
- Sensitivities to masking: model picks up right parts of information.
- Encoding visualizations: encoder is able to automatically learn semantic artifacts.
- Error inspection: some fails regarding the exact meaning of `(.*)`.

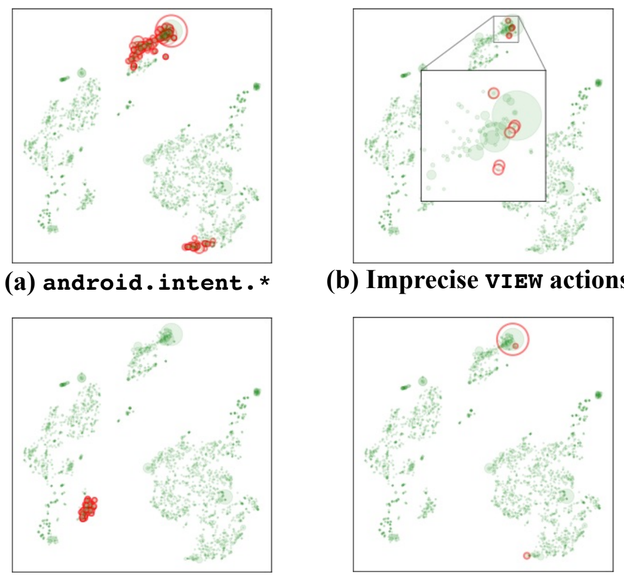
**Table 5: Some CNN kernels and their top stimuli**

| conv1d_size5: 14 segment | activation | conv1d_size5: 3 segment | activation | conv1d_size7: 0 segment | activation |
|---|---|---|---|---|---|
| (.*)R | 1.951 | null} | 3.796 | TAVIEWA | 3.704 |
| (.*)u | 1.894 | null, | 2.822 | n.VIEW" | 3.543 |
| (.*)t | 1.893 | sulle | 2.488 | y.VIEW" | 3.384 |

**Figure 6: Explaining sensitivities to masking**

```
{"action": "NULL-CONSTANT", "categories": null}
{"actions": ["NULL-CONSTANTPOP_DIALOG", "NULL-CONSTANTPUSH_DIALOG_(.*)",
"(.*)REPLACE_DIALOG_(.*)", "APP-00489869YB964702HUPDATE_VIEW"], "categories":
null}

{"action": "NULL-CONSTANTREPLACE_DIALOG_(.*)", "categories": null}
{"actions": ["(.*).CLOSE"], "categories": null}

{"action": "(.*)", "categories": null}
{"actions": ["android.media.RINGER_MODE_CHANGED",
"sakurasoft.action.ALWAYS_LOCK", "android.intent.action.BOOT_COMPLETED"],
"categories": null}

{"action": "(.*)LOGIN_SUCCESS", "categories": null}
{"actions": ["NULL-CONSTANTLOGIN_FAIL", "NULL-
CONSTANTCREATE_PAYMENT_SUCCESS", "(.*)FATAL_ERROR",
"(.*)CREATE_PAYMENT_FAIL", "NULL-CONSTANTLOGIN_SUCCESS"], "categories": null}

{"action": "APP-00489869YB964702HREPLACE_DIALOG_(.*)", "categories": null}
{"actions": ["APP-00489869YB964702HLOGIN_FAIL", "APP-
00489869YB964702HCREATE_PAYMENT_FAIL", "NULL-CONSTANTCREATE_PAYMENT_SUCCESS",
"(.*)FATAL_ERROR", "NULL-CONSTANTLOGIN_SUCCESS"], "categories": null}

{"action": "com.joboevan.push.message.(.*)", "categories": null}
{"actions": ["com.joboevan.push.message.NULL-CONSTANT"], "categories": null}

{"action": "", "categories": ["(.*)"]}
{"actions": ["com.dreamware.Hells_Kitchen.CONCORRENTE"], "categories":
["android.intent.category.DEFAULT"]}

{"action": "(.*)", "categories": null}
{"actions": ["android.intent.action.MEDIA_BUTTON",
"com.ez.addon.MUSIC_COMMAND", "android.media.AUDIO_BECOMING_NOISY"],
"categories": null}
```

**Figure 7: Intent encodings visualized using t-SNE**



(a) `android.intent.*` (b) Imprecise `VIEW` actions (c) `dev*.app*.*.FEED*` (d) `DEFAULT`, `(.*)` and null categories

This work is partially supported by the National Science Foundation Grants CNS-1563831 and CCF-1652140 and by the Google Faculty Research Award.