Predicting English keywords from Java Bytecodes using Machine Learning

Pablo Ariel Duboue

Les Laboratoires Foulab
(Montreal Hackerspace)
999 du College
Montreal, QC, H4C 2S3

REcon – June 15th, 2012

Outline

Long Term Vision

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Long Term Vision

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Long Term Vision

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
```


```
rd.q.e()
rd.q.e()
t.w()
```

Long Term Vision

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
    4 invokeinterface org.jpc.support.j.e()
    9 aload_0
    10 getfield org.jpc.emulator.f.i
    13 invokevirtual org.jpc.emulator.motherboard.q.e()
    16 aload_0
    17 getfield org.jpc.emulator.f.j
    20 invokevirtual org.jpc.emulator.motherboard.q.e()
    23 iconst_0
    24 istore_2
    25 iload_1
    26 ifle 128
    29 aload_0
    30 getfield org.jpc.emulator.f.b
    33 invokevirtual org.jpc.emulator.processor.t.w()
```

Long Term Vision

```
private final int c(int) {
    0 aload_0
    1 getfield org.jpc.emulator.f.v
```


```
rd.q.e()
rd.q.e()
t.w()
```

Current Status

- Does this work now? Not really.

Current Status

- Does this work now? Not really.

About the Speaker

- From Cordoba, Argentina.
- Natural Language Generation at Columbia University.
  - PhD thesis on learning the structure of biographies.
- Research Scientist at IBM.
  - DeepQA Watson Project.
- Independent Scientist.
  - Member of Foulab.
- Interested in Crypto and RE.
  - Attended ToorCamp in ’10.

Intuitions Behind the Approach

- Attended REcon last year.
- Gap between tools and the way practitioners work.
  - Tools: from the programming language and translators community (i.e., compilers).
  - Practitioners: more similar to NLP practitioners, more driven by intuitions.
- More machine learning-driven tools?

Intuitions Behind the Approach

- Attended REcon last year.
- Gap between tools and the way practitioners work.
  - Tools: from the programming language and translators community (i.e., compilers).
  - Practitioners: more similar to NLP practitioners, more driven by intuitions.
- More machine learning-driven tools?

Objective of this Talk

- Share some really early results.
  - Really share them, the data, models and scripts are available at `http://keywords4bytecodes.org`.
- Find potential users and **collaborators**.

Outline

Using the Debian Archive

- `apt-file search --package-only .jar`
  - 1,400+ packages
- `dpkg-query -p` *package name*
  - Look for `Source` field
- `dpkg-source -x` *source .dsc*
  - Search for Java source files.
- `dpkg -x` *binary .deb*
  - Search for jars, disassemble the methods.

Assembling the Bytecodes / Javadoc File

- Disassemble using `jclassinfo --disasm`
- Dump Javadoc comments using qdox.
  - A lightweight Java source parsing library.
- Heuristically match source methods to compiled methods.
  - Normalize source code signatures to binary signatures.
- Final corpus:
  - 350,000+ methods.
  - 5.9M words.
  - 8.4M JVM instructions.

Preprocessing

- What is a term is a key issue.
  - Currently: lowercase, split on spaces and certain punctuations.
  - Everything not alphanumeric is replaced with '_'
- Poor choice moved to unsupervised tokenization.
  - Discussed later.
- Took top 2,000 keywords after the top 100 keywords
  - Stopword removal.

Outline

Train *n* classifiers for large *n*

- For each of the top 2,000 keywords:
  - Put aside the methods that contain the keyword on its Javadoc.
  - Sample the methods that do not.
- Positive and negative training data.
- Train a classifier for each keyword.
  - For classifier, I use `dbacl`.
  - High performance, very robust tokenizer.
  - The input to the classifier is the **ascii rendering** of the disassembled bytecodes.

Naïve Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Go from observed to predicted.

$$P(\textit{calculation}|\texttt{fadd}) = \frac{P(\texttt{fadd}|\textit{calculation})P(\texttt{fadd})}{P(\textit{calculation})}$$

- Naïve Bayes: complete independence assumptions.
  - Clearly false in the case of bytecode and human language.
  - Good start.
  - Very resilient to noise in the data.
  - Used in spam detection.

Results

- Trained on 10x more negatives.
- Random accuracy: 1/2000 = 0.05%
- Because there are so many negatives, accuracy is not meaningful:
  - For *policy*, we get 99.67% accuracy because it gets 70,138 true negatives right.
  - And only 2 true positives (!)
- Precision and Recall
  - $\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$
  - $\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$
  - $\text{F} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Results

Keyword	P	R	F
com	0.33	0.36	0.34
smartgwt	0.08	0.96	0.15
widgets	0.07	0.75	0.13
advanced	0.09	0.17	0.12
occurs	0.08	0.15	0.11
public	0.06	0.32	0.10
setting	0.12	0.08	0.09
receiver	0.08	0.11	0.09
most	0.07	0.12	0.09
show	0.04	0.12	0.06

Analysis

- com? smartgwt? $\Rightarrow$ package names
  - Lot of signal on the `invoke` instruction
- A missing keyword does not make it *wrong*
- There's signal, more work to go...

Outline

In Reverse Engineering

- Hinting Subroutines
  - The motivating example at the beginning.
  - “Beacon identification” in Software Engineering.
  - Even if it fails, it might work in a predictable manner that can still be useful.
- Custom (malware) VMs
  - Identifying which methods correspond to different VM operations (addition, jump, etc).
  - Lack of training data?
  - Native code.

In Security

- Dalvik Word Clouds.
  - Use `dex2jar`, obtain word clouds for the whole executable.
  - Maybe the user can tell if anything looks fishy there?
- Flagging Suspicious Methods.
  - Finding methods that can be described with keywords very different from the rest of the existing methods.
  - Kullback-Leibler divergence.
  - Lots of false positives.
  - Can be done with dynamically generated bytecodes.

Elsewhere

- Method Search
  - Searching for a method related to certain keywords.
  - Useful in case source code is missing.
  - Can be evaluated against a set of queries without annotated data.
- Helping map similar methods.
  - A heuristic approach to “code clone detection”.

Outline

Improving the Data

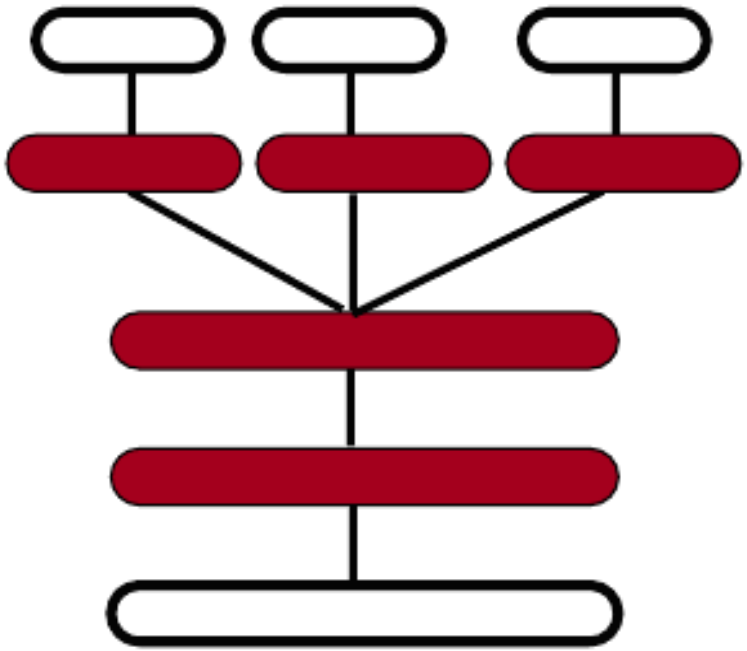
- Clustering: reducing errors due to missing text.
  - Predict keywords that could have been used.
- Aligning bytecode to Javadoc the “Right Way”
  - Modifying the compiler and re-building the packages.
- Morfessor
  - Unsupervised tokenization.
- LDA: dimensionality reduction.
  - Finding similarities at the bytecode level.

Discriminative Learning

- Naïve Bayes is generative
  - Provides a full probabilistic model for all variables.
  - “Reversible”
- Discriminative models only model the target variables.
  - Many times have better performance than generative models.

Multi-task Neural Networks

- Inspired from the talk by Prof. Bengio last week at Semantic Interpretation in an Actionable Context workshop.
- Learn about the code, then generate from there.



Outline

Other Issues

- Obfuscate the Training
- Dalvik
- Beyond Keywords: Full Phrases
- Dynamic vs. Static features

Conclusions

- An idea with potential.
- Might be too early?
  - We will see.

- mailto:pablo.duboue@gmail.com
- http://keywords4bytecodes.org
- @pablodduboue
- DrDub on FreeNode

Acknowledgements

- Foulab.
  - Danukeru.
- REcon organizers.
  - Subgraph.
- Annie Ying.

- mailto:pablo.duboue@gmail.com
- http://keywords4bytecodes.org
- @pablodduboue
- DrDub on FreeNode