Visualizing API Usage Examples at Scale

Elena L. Glassman\(^1\)*, Tianyi Zhang\(^2\)*, Björn Hartmann\(^1\), Miryung Kim\(^2\)

\(^1\)University of California, Berkeley
\(^2\)University of California, Los Angeles

The two lead authors contributed equally to the work as part of an equal collaboration between both institutions.
Using APIs properly is a key challenge in programming
Status quo for answering “How have others used this API?”

Developers often search online for code examples to learn APIs [Sadowski et al., 2016]
Status quo for answering
“How have others used this API?”

• **Programmers only inspect a few of search results.**

• Individual code examples may suffer from
  • API usage violations [Zhang 2018]
  • insecure coding practices [Fischer 2017]
  • unchecked obsolete usage [Zhou & Walker 2016]
  • low readability [Treude & Robillard 2017]
How can we enable programmers to inspect more examples?
How is EVERYONE using this API?
"How do other people create a FileInputStream object?"

API call of interest

new FileInputStream()

API Usage Questions:

• What arguments to pass into this call?
• How do I create these arguments?
• Do I need to check any pre-condition?
• What other methods to call together?
• What exception(s) does it throw?
• How do I handle the created object?

[Ko et al. 2004, Duala-Ekoko & Robillard 2012]
Designing an API Skeleton

```java
declarations
try {
    pre method call
    if (...) {
        new FileInputStream()
        if (...) {
            post method call
        }
    }
} catch (...) {
    exception handling call
}
finally {
    ...
}
What other methods are called before and after?
What exception(s) are thrown?
Focal API call
```
Examplore: Visualizing Code Examples at Scale

- **Focal API**: `new FileInputStream()`
- **Many code examples using this call**
- **Interactive visualization showing common usage and frequency**
Mining API Usage from a Large Code Corpus

API call of interest

new FileInputStream()
crawl

380K GitHub repositories

Many code examples using this call

```java
if (file != null) {
    return new FileInputStream(file);
} else {
    return new ByteArrayInputStream(...
}
File file = new File(_basePath + "\" + path);
try {
    return new FileInputStream(file);
} catch (FileNotFoundException e) {
    throw new IllegalArgumentException(e);
}
File propertiesFile = getPropertiesFile();
try {
    InputStream in = new FileInputStream(propertiesFile);
    workspaceProperties.load(in);
} catch (IOException e) {
}
```
Program Slicing and Labeling

Labeled Code Examples

private void getLatestVersion() {
    // TODO Auto-generated method stub
    File temp = new File(Environment.getExternalStorageDirectory().toString() + "/pdTemp");
    try {
        List<File> listMain = IoUtils.extractZipResource(new FileInputStream(pdzZipPath), temp, true);
        if (listMain.size() != 0) {
            for (File f : listMain) {
                if (f.isDirectory()) folderName = f.getName();
                dpMainfileName = f.getName();
                InputStream s = new FileInputStream(f);
                BufferedReader reader = new BufferedReader(new InputStreamReader(s));
                String line;
                while ((line = reader.readLine()) != null) {
                    String version;
                    if (line.contains(" version: ")) {
                        Log.d("LatestVersionLine", line);
                        version = line.substring(line.lastIndexOf(":") + 1, line.length() - 1);
                        this.latestVersion = Float.parseFloat(version);
                        break;
                    } else {
                        version = "0";
                        this.latestVersion = Float.parseFloat(version);
                    }
                    reader.close();
                    Log.d("LatestVersion", latestVersion + ");
                    break;
                }
            }
        }
        if (foundmainPd) {
            closePd();
        } else {
            closePd();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
Code Canonicalization

Program Slicing and Labeling

Labeled Code Examples
Back-end Architecture of Code Mining and Slicing

- AST Traversal
- Program Slicing
- Type Annotation

380K Java Projects from GitHub

Sliced examples with annotations

Canonicalization

Feature Extraction

Json

API usage features

MongoDB
Examplore Interface

```java
if (file != null) {
    return new FileInputStream(file);
} else {
    return new ByteArrayInputStream(…);
}

File file = new File(String);
try {
    return new FileInputStream(file);
} catch (FileNotFoundException e) {
    throw new IllegalArgumentException(e);
}

File file = getPropertiesFile();
try {
    InputStream stream = new FileInputStream(file);
    workspaceProperties.load(stream);
} catch (IOException e) {
}
```
Examplore Interface

```java
if (file != null) {
    return new FileInputStream(file);
} else {
    return new ByteArrayInputStream(...);
}

File file = new File(String);
try {
    return new FileInputStream(file);
} catch (FileNotFoundException e) {
    throw new IllegalArgumentException(e);
}

File file = getPropertiesFile();
try {
    InputStream stream = new FileInputStream(file);
    workspaceProperties.load(stream);
} catch (IOException e) {
    // Handle IOException
}
```
if (file != null) {
    return new FileInputStream(file);
} else {
    return new ByteArrayInputStream(…
}

File file = new File(String);
try {
    return new FileInputStream(file);
} catch (FileNotFoundException e) {
    throw new IllegalArgumentException(e);
}

File file = getPropertiesFile();
try {
    InputStream stream = new FileInputStream(file);
    workspaceProperties.load(stream);
} catch (IOException e) {
Live Demo
Theoretical Basis

Mutual alignment of contrasting examples

• Mutual alignment can promote comprehension and abstraction.

• Comparison brings greater insight into the common structure.

• Best results come from
  • jointly interpreting examples
  • listing specific correspondences across examples

[Kurtz et al. Learning by Analogical Bootstrapping, J. Learning Sciences, 2001]
Evaluation
Within-Subjects Lab Study on Answering API Usage Questions

• Recruited 16 CS students from UC Berkeley

• Picked out 3 APIs
  • 75% of participants had used `Map.get`
  • 38% had used `SQLiteDatabase.query`
  • 19% had used `Activity.findViewById`

• 50 min user study answering usage questions
  • 25 min block for API₁ [Baseline / Examplore]
  • 25 min block for API₂ [Examplore / Baseline]
Evaluation
Within-Subjects Lab Study on Answering API Usage Questions

Sample of API Usage Questions

• Q2. How do I create or initialize the arguments so I can call this API method?
• Q6. How do programmers handle the return value of this API method?
• Q7. What are the exceptions that programmers catch and how do programmers handle potential exceptions?
Evaluation
Within-Subjects Lab Study on Answering API Usage Questions

Sample of API Usage Questions

- Q8. How might you **modify this code example** on Stack Overflow if you were going to copy and paste it into your own solution to the original prompt?
Lab Study Results

• Examplore users investigated many relevant examples.

• Baseline users often answered based on one example or by guessing.

Average # of correct answers on Q1-7

Baseline: 0/7
Examplore: 7/7

Mean difference is statistically significant (paired t-test: t=3.02, df=15, p-value<0.01)
Lab Study Results

For Q8, 88% of participants gave valid comments about the StackOverflow answer.

The majority of participants’ critiques...

• (Using the baseline) were about style and the mechanics of adaptation

• (Using Examplore) were about safety

• Q8. How might you modify this code example on Stack Overflow if you were going to copy and paste it into your own solution to the original prompt?
Lab Study Results

“[EXAMPLORERE] provided structure to learning about the API.

This structure guides functionality while still showing variety of use.

The frequency of [each option] shows me if I am looking at a random corner case or something commonly used.”

-P16
Future applications

- Release as a public resource based on Github
- For a specific codebase / organization
- Code review
- In-editor sidebar display
- Corporate on-boarding
- Data-driven library design and revision
Summary

• The API skeleton is a key enabler for visualizing a large collection of API usage examples.

• The statistical distribution demonstrates the common and uncommon API usage in the community.

• By interacting with the skeleton, users can easily filter code examples and quickly drill down to those of interest.

Demo is available at https://eglassman.github.io/examplore/