Can Stack Overflow Posts *Capture* Library Features?
The 19th Belgium-Netherlands Software Evolution Workshop

Camilo Velázquez-Rodríguez, Eleni Constantinou and Coen De Roover
December 4th, 2020
Introduction

Software products that evolve together in the same environment [1]

Introduction

Software products that evolve together in the same environment [1]

Introduction

Software products that evolve together in the same environment [1]

Introduction

Software products that evolve together in the same environment [1]

Software products that evolve together in the same environment [1]

Introduction

Guava

But trawling through Javadoc isn't always the most effective way to learn how to make best use of a library. Here, we try to provide readable and pleasant explanations of some of the most popular and most powerful features of Guava.
Introduction

Guava

But trawling through Javadoc isn't always the most effective way to learn how to make best use of a library. Here, we try to provide readable and pleasant explanations of **some of the most popular and most powerful features of Guava**.

- **Collections**: Guava's extensions to the JDK collections ecosystem. These are some of the most mature and popular parts of Guava.
  - **Immutable collections**, for defensive programming, constant collections, and improved efficiency.
  - **New collection types**, for use cases that the JDK collections don't address as well as they could: multisets, multimaps, tables, bidirectional maps, and more.
  - **Powerful collection utilities**, for common operations not provided in `java.util.Collections`.
  - **Extension utilities**: writing a `Collection` decorator? Implementing `Iterator`? We can make that easier.
Introduction

Guava

But trawling through Javadoc isn't always the most effective way to learn how to make best use of a library. Here, we try to provide readable and pleasant explanations of some of the most popular and most powerful features of Guava.

- **Collections**: Guava's extensions to the JDK collections ecosystem. These are some of the most mature and popular features of Guava.

- **Graphs**: a library for modeling graph-structured data, that is, entities and the relationships between them. Key features include:
  - **Graph**: a graph whose edges are anonymous entities with no identity or information of their own.
  - **ValueGraph**: a graph whose edges have associated non-unique values.
  - **Network**: a graph whose edges are unique objects.
  - Support for graphs that are mutable and immutable, directed and undirected, and several other properties.

https://github.com/google/guava/wiki
Introduction

Guava

But trawling through Javadoc isn't always the most effective way to learn how to make best use of a library. Here, we try to provide readable and pleasant explanations of some of the most popular and most powerful features of Guava.

- **Collections**: Guava's extensions to the JDK collections ecosystem. These are some of the most mature and popular features of Guava.
- **Graphs**: A library for modeling graph-structured data, that is, entities and the relationships between them. Key features include self-loops, parallel edges, and multi-graphs.
- **Strings**: A few extremely useful string utilities: splitting, joining, padding, and more.
- **Primitives**: Operations on primitive types, like `int` and `char`, not provided by the JDK, including unsigned variants for some types.
- **Ranges**: Guava's powerful API for dealing with ranges on `Comparable` types, both continuous and discrete.
- **I/O**: Simplified I/O operations, especially on whole I/O streams and files, for Java 5 and 6.
- **Hashing**: Tools for more sophisticated hashes than what's provided by `Object.hashCode()`, including Bloom filters.
- **EventBus**: Publish-subscribe-style communication between components without requiring the components to explicitly register with one another.
- **Math**: Optimized, thoroughly tested math utilities not provided by the JDK.
- **Reflection**: Guava utilities for Java's reflective capabilities.

[https://github.com/google/guava/wiki](https://github.com/google/guava/wiki)
Introduction

Stack Overflow Q&A

How do I convert a String to an int in Java?
Guava equivalent for IOUtils.toString(InputStream)

How do I compare strings in Java?
Remove last character of a StringBuilder?

Guava: Splitter and considering Escaping?

initializing a Guava ImmutableMap

Flattening an Iterable<Iterable<T>> in Guava

How to directly initialize a HashMap (in a literal way)?

Google Guava isNullOrEmpty for collections

Google Guava “zip” two lists

Merge ranges with guava

Ranges

What is the simplest way to read a file into String?

Utils to read resource text file to String (Java)

String utilities

Collection utilities

I/O
Pipeline

API Usages and Textual Information → Filter out unrelated usages → Filtered Usages and Textual Information

Named Groups of Usages (a.k.a Features) → Groups of Usages → Group Usages by Textual Similarity
Extracting Usages

Gson gson = new Gson();
PrintWriter out = new PrintWriter();
String json = gson.toJson('{}');
out.print(json)
out.close();

[com.google.Gson: 0.91
org.apache.http.NameValuePair: 0.04]
Gson gson = new Gson();
[java.io.OutputStream: 0.4
java.io.BufferedReader: 0.4]
PrintWriter out = new PrintWriter();
[com.google.Gson: 0.98
org.apache.http.NameValuePair: 0.01]
String json = gson.toJson('{}');
[java.io.OutputStream: 0.95
java.io.BufferedReader: 0.04]
out.print(json)
[java.io.OutputStream: 0.96
java.io.BufferedReader: 0.03]
out.close();
Filtering

Compute Covariance Matrix in Java

Here is a short example, how you can create it with Apache Commons Math (3.5):

```
RealMatrix mx = MatrixUtils.createRealMatrix(new double[][]{
    {1, 2, 3},
    {2, 4, 6}
});
RealMatrix cov = new Covariance(mx).getCovarianceMatrix();
```

How to average BigDecimals using Streams?

```
MutableDoubleList doubles = DoubleLists.mutable.with(1.0, 2.0, 3.0, 4.0);
List<BigDecimal> bigDecimals = doubles.collect(BigDecimal::new);
BigDecimal average =
    bigDecimals.stream()
        .collect(Collectors.summarizingBigDecimal(e -> e))
        .getAverage(MathContext.DECIMAL32);
Assert.assertEquals(BigDecimal.valueOf(2.5), average);
```
Filtering

Compute Covariance Matrix in Java

Here is a short example, how you can create it with Apache Commons Math (3.5):

```java
RealMatrix mx = MatrixUtils.createRealMatrix(new double[][]{
    {1, 2, 3},
    {2, 4, 6}
});
RealMatrix cov = new Covariance(mx).getCovarianceMatrix();
```

How to average BigDecimals using Streams?

```java
MutableDoubleList doubles = DoubleLists.mutable.with(1.0, 2.0, 3.0, 4.0);
List<BigDecimal> bigDecimals = doubles.collect(BigDecimal::new);
BigDecimal average = bigDecimals.stream()
    .collect(Collectors.summarizingBigDecimal(e -> e))
    .getAverage(MathContext.DECIMAL32);
Assert.assertEquals(BigDecimal.valueOf(2.5), average);
```
Filtering

2. Process API usage names.
3. Load prebuilt embeddings model.
4. Compute semantic similarity.

Compute Covariance Matrix in Java

Here is a short example, how you can create it with Apache Commons Math (3.5):

```java
RealMatrix mx = MatrixUtils.createRealMatrix(new double[][]{{1, 2, 3}, {2, 4, 6}});
RealMatrix cov = new Covariance(mx).getCovarianceMatrix();
```

How to average BigDecimals using Streams?

```java
MutableDoubleList doubles = DoubleLists.mutable.with(1.0, 2.0, 3.0, 4.0);
List<BigDecimal> bigDecimals = doubles.collect(BigDecimal::new);
BigDecimal average = bigDecimals.stream()
    .collect(Collectors.summingBigDecimal(e -> e))
    .getAverage(MathContext.DECIMAL32);
Assert.assertEquals(BigDecimal.valueOf(2.5), average);
```
Filtering

2. Process API usage names.
3. Load prebuilt embeddings model.
4. Compute semantic similarity.

Compute Covariance Matrix in Java

Here is a short example, how you can create it with Apache Commons Math (3.5):

```java
RealMatrix mx = MatrixUtils.createRealMatrix(new double[][]{
    {1, 2, 3},
    {2, 4, 6}
});
RealMatrix cov = new Covariance(mx).getCovarianceMatrix();
```

How to average BigDecimals using Streams?

```java
MutableDoubleList doubles = DoubleLists.mutable.with(1.0, 2.0, 3.0, 4.0);
List<BigDecimal> bigDecimals = doubles.collect(BigDecimal::new);
BigDecimal average = bigDecimals.stream()
    .collect(Collectors.summarizingBigDecimal(e -> e)
        .getAverage(MathContext.DECIMAL32);
Assert.assertEquals(BigDecimal.valueOf(2.5), average);
```
Filtering

2. Process API usage names.
3. Load prebuilt embeddings model.
4. Compute semantic similarity.

Compute Covariance Matrix in Java

Here is a short example, how you can create it with Apache Commons Math (3.5):

```java
RealMatrix mx = MatrixUtils.createRealMatrix(new double[][]{
    {1, 2, 3},
    {2, 4, 6}
});
RealMatrix cov = new Covariance(mx).getCovarianceMatrix();
```

How to average BigDecimals using Streams?

```java
MutableDoubleList doubles = DoubleLists.mutable.with(1.0, 2.0, 3.0, 4.0);
List<BigDecimal> bigDecimals = doubles.stream()
    .map(Decimal::new)
    .collect(Collectors.toCollection(ArrayList::new));
BigDecimal average = bigDecimals.stream()
    .collect(Collectors.summarizingBigDecimal(e -> e))
    .getAverage(MathContext.DECIMAL32);
Assert.assertEquals(BigDecimal.valueOf(2.5), average);
```
Grouping


Tag Tag Tag Tag Tag

java http httprequest httpurlconnection urlconnection
Grouping

API Usages and Textual Information → Filter out unrelated usages → Filtered Usages and Textual Information → Group Usages by Textual Similarity → Named Groups of Usages (a.k.a Features) → Groups of Usages

Relation between Tags

Tag → Tag → Tag → Tag → Tag

java → http → httprequest → httpurlconnection → urlconnection

Relation between Tags

http → java → httprequest → urlconnection → httpurlconnection

7278 1034 502 8674 5262 6473 6363 8853 4023 7323
Grouping

API Usages and Textual Information → Filter out unrelated usages → Filtered Usages and Textual Information → Group Usages by Textual Similarity → Groups of Usages → Named Groups of Usages (a.k.a Features)

Relation between Tags

tag1 → tag2 → tag3 → tag4 → tag5

Tags: java, http, httprequest, httpurlconnection, urlconnection
Grouping

API Usages and Textual Information → Filter out unrelated usages → Filtered Usages and Textual Information

Named Groups of Usages (a.k.a Features) → Groups of Usages → Group Usages by Textual Similarity

Relation between Tags:

Tag → Tag → Tag → Tag → Tag

java http httppost httpurlconnection urllibconnection
Grouping

https://github.com/taynaud/python-louvain
Grouping

Identifying Features in Forks

Shunzi Zhou
Carnegie Mellon University

Yingfei Xiong
Peking University

Ștefan Stănicăulescu
IT University of Copenhagen

Andrzej Wasowski
IT University of Copenhagen

Olaf Leffenich
University of Passau

Christian Küstner
Carnegie Mellon University

ABSTRACT

Forking development has been widely used both in open-source communities and in industry, because it gives developers flexibility to modify their own fork without affecting others. Unfortunately, this mechanism has downsides: When the number of forks becomes large, it is difficult for developers to get or maintain an overview of activities in the forks. Current tools provide little help. We introduce Isvorc, an approach to automatically identify non-merged features in forks and to generate an overview of active forks in a project. The approach clusters cohesive code fragments using code and network-analysis techniques and uses information-retrieval techniques to label clusters with keywords. The clustering is effective, with 90% accuracy on a set of known features. In addition, a human-subject evaluation shows that Isvorc can provide actionable insight for developers of forks.

ACM Reference Format:


1 INTRODUCTION

Forking is a lightweight and easy mechanism that allows developers, both in open-source and in industry, to start development from

https://github.com/taynaud/python-louvain

Relation between Tags

Tag

Tag

Tag

Tag

Tag

java

http

httprequest

httpurlconnection

urlconnection

Figure 1: Github's network graph shows commits across known forks, but is difficult to use to gain an overview of activities in projects with many forks. Isvorc’s overview summarizes features in active forks.
Groups

Library: org.apache.commons.math
Mathematics Library from the Apache Foundation
Groups

Library: org.apache.commons.math

Mathematics Library from the Apache Foundation
Groups

Library: com.itextpdf

PDF creation and transformation
Groups

Library: com.itextpdf
PDF creation and transformation
Groups

Library: nz.ac.waikato.cms.weka
Data mining and Machine Learning Library
Groups

Library: nz.ac.waikato.cms.weka
Data mining and Machine Learning Library
Future Work

API Usages and Textual Information

Filter out unrelated usages

Filtered Usages and Textual Information

Group Usages by Textual Similarity

Named Groups of Usages (a.k.a Features)

Groups of Usages
Conclusion

Introduction

Survey of 16 participants from different companies
1. Technical Factors
   • Functionality
   • Quality
   • Release Process
2. Human Factors
   • Stakeholders
     • Individual
     • Organization
   • Community
3. Economic Factors
   • Total cost of ownership
   • Risk

Stack Overflow Q&A

How do I convert a string to an int in Java?
Removing last character of a String

Initializing a Queue

String utilities

What is the simplest way to read a file into String?

Groups

Library: org.apache.commons.math
Mathematics Library from the Apache Foundation

Pipeline

Filter out unrelated usages

Groups of Usages by Textual Similarity

Named Groups of Usages (a.k.a. Features)