Uncovering Library Features from API Usage on Stack Overflow

The 29th IEEE International Conference on Software Analysis, Evolution and Reengineering

Camilo Velázquez-Rodríguez, Eleni Constantinou and Coen De Roover

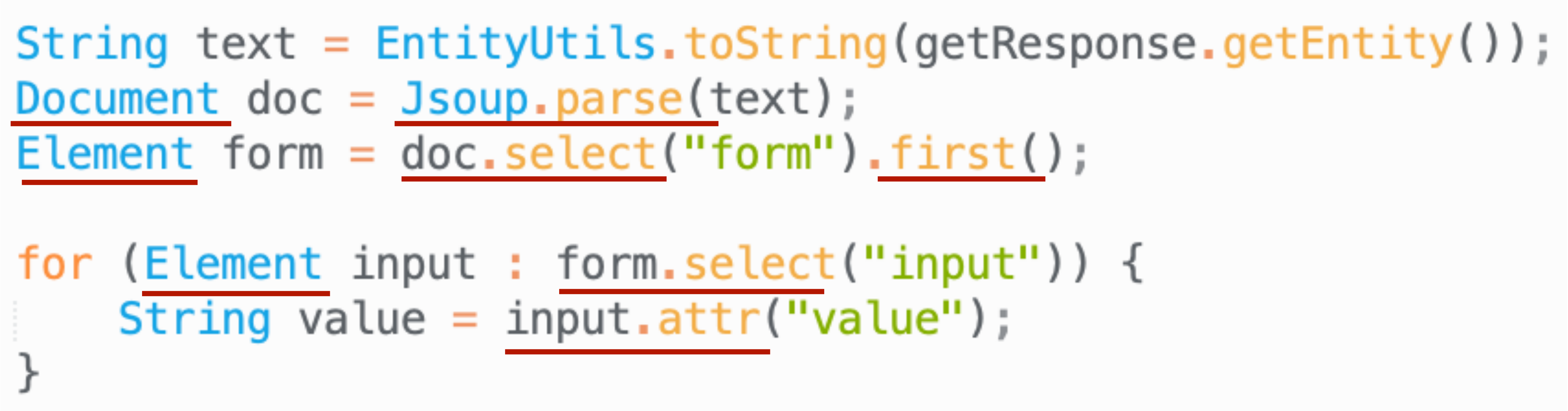
March 15th, 2022

VRIJE UNIVERSITEIT BRUSSEL
EINDHOVEN UNIVERSITY OF TECHNOLOGY
SECO-Assist

@cvelazquezr

Introduction

JSoup library

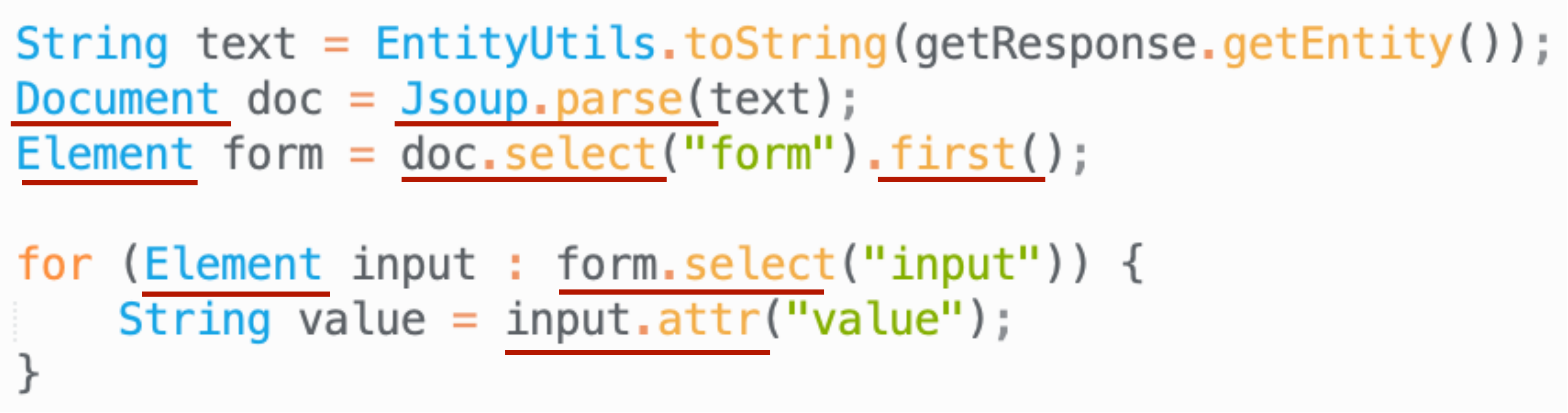


```
String text = EntityUtils.toString(getResponse.getEntity());
Document doc = Jsoup.parse(text);
Element form = doc.select("form").first();

for (Element input : form.select("input")) {
    String value = input.attr("value");
}
```

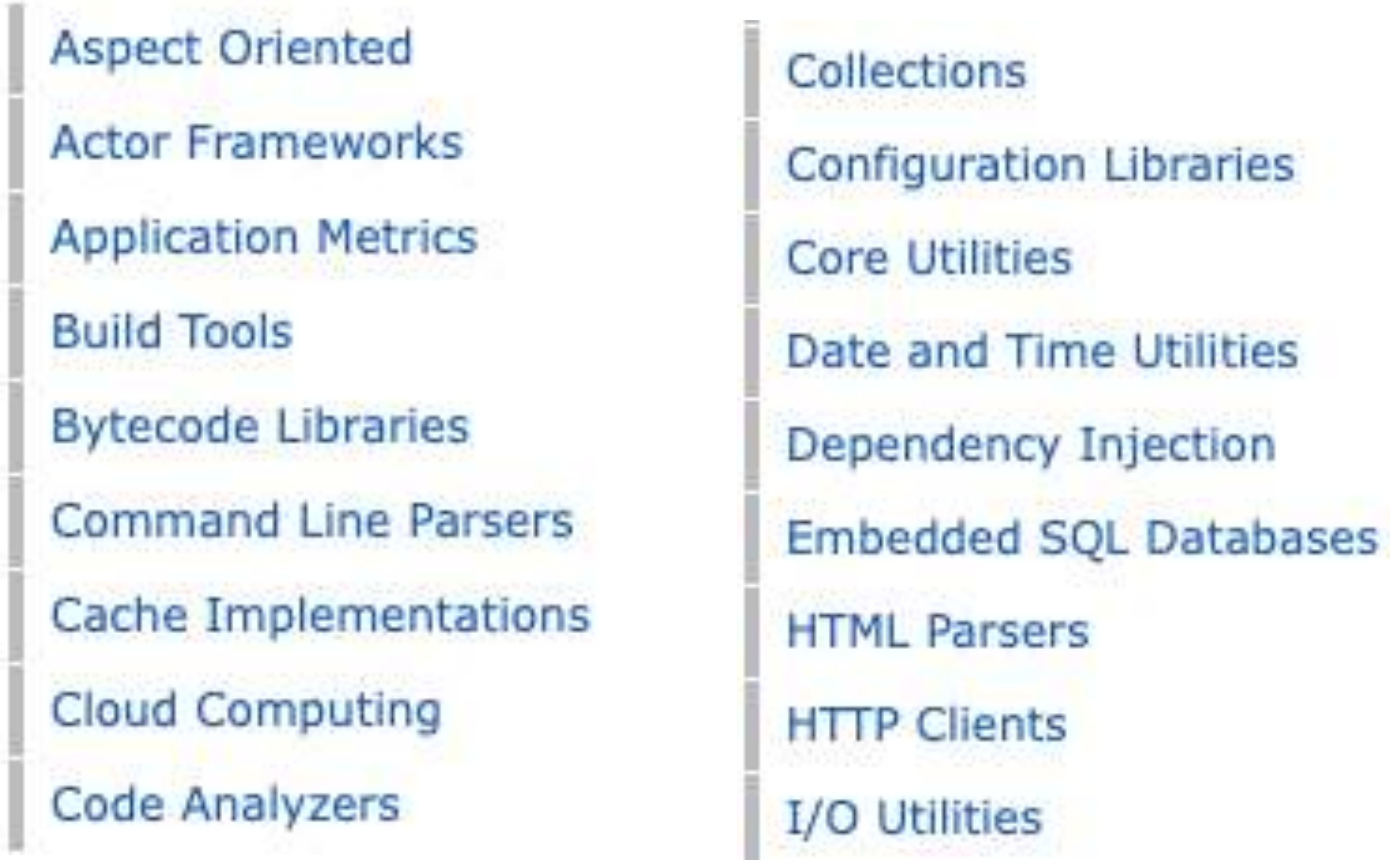
Introduction

JSoup library



```
String text = EntityUtils.toString(getResponse.getEntity());
Document doc = Jsoup.parse(text);
Element form = doc.select("form").first();

for (Element input : form.select("input")) {
    String value = input.attr("value");
}
```

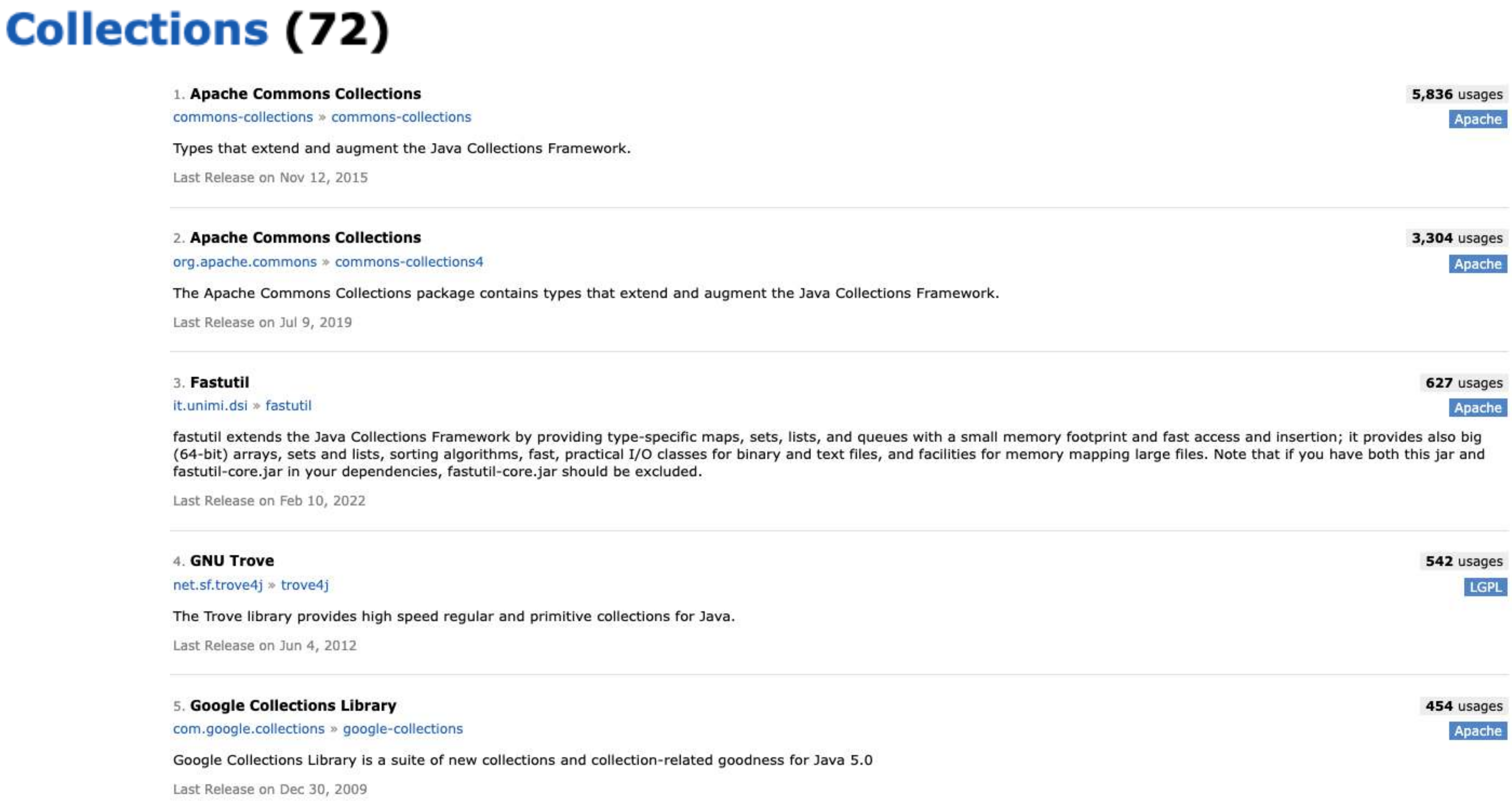


Aspect Oriented
Actor Frameworks
Application Metrics
Build Tools
Bytecode Libraries
Command Line Parsers
Cache Implementations
Cloud Computing
Code Analyzers

Collections
Configuration Libraries
Core Utilities
Date and Time Utilities
Dependency Injection
Embedded SQL Databases
HTML Parsers
HTTP Clients
I/O Utilities

⋮

Introduction



Collections (72)

1. **Apache Commons Collections** — **5,836** usages — Apache
commons-collections » commons-collections

Types that extend and augment the Java Collections Framework.

Last Release on Nov 12, 2015

2. **Apache Commons Collections** — **3,304** usages — Apache
org.apache.commons » commons-collections4

The Apache Commons Collections package contains types that extend and augment the Java Collections Framework.

Last Release on Jul 9, 2019

3. **Fastutil** — **627** usages — Apache
it.unimi.dsi » fastutil

fastutil extends the Java Collections Framework by providing type-specific maps, sets, lists, and queues with a small memory footprint and fast access and insertion; it provides also big (64-bit) arrays, sets and lists, sorting algorithms, fast, practical I/O classes for binary and text files, and facilities for memory mapping large files. Note that if you have both this jar and fastutil-core.jar in your dependencies, fastutil-core.jar should be excluded.

Last Release on Feb 10, 2022

4. **GNU Trove** — **542** usages — LGPL
net.sf.trove4j » trove4j

The Trove library provides high speed regular and primitive collections for Java.

Last Release on Jun 4, 2012

5. **Google Collections Library** — **454** usages — Apache
com.google.collections » google-collections

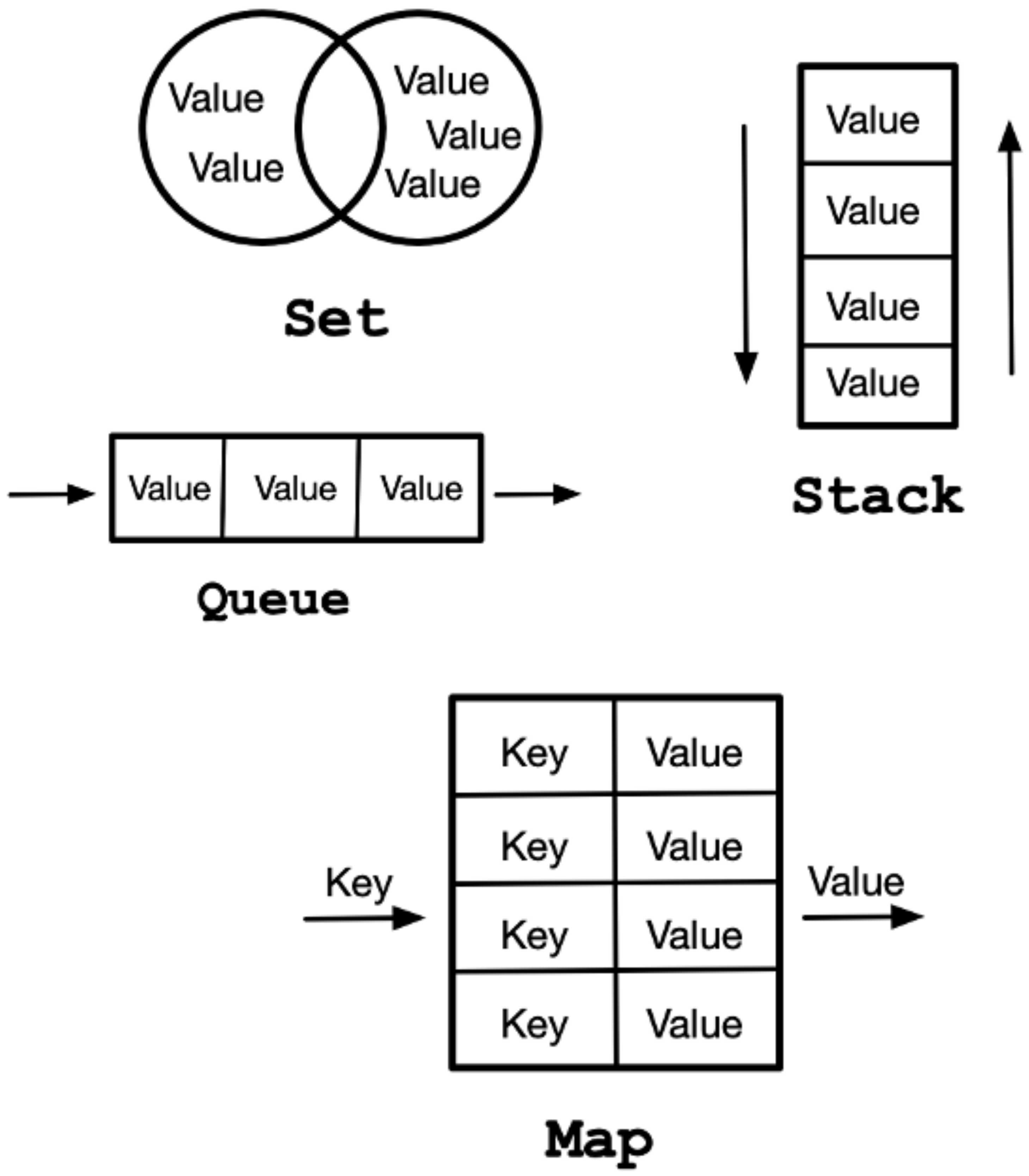
Google Collections Library is a suite of new collections and collection-related goodness for Java 5.0

Last Release on Dec 30, 2009

Introduction

Collections (72)

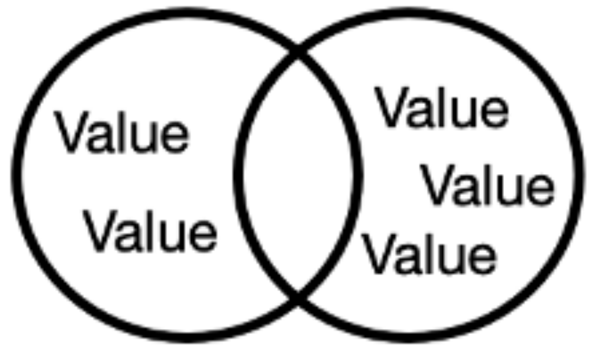
API Features



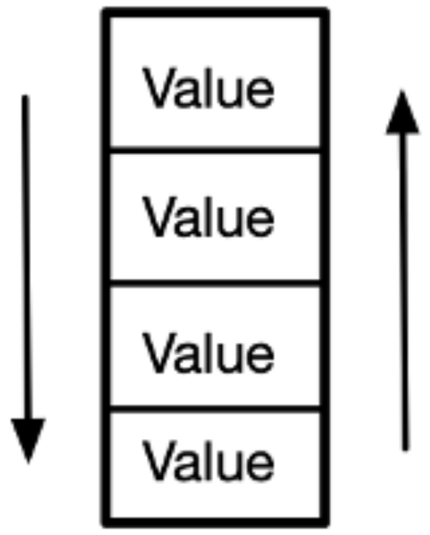
Introduction

Collections (72)

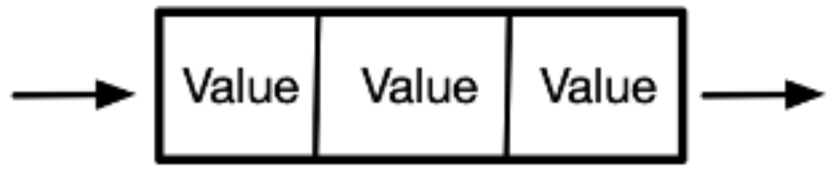
API Features



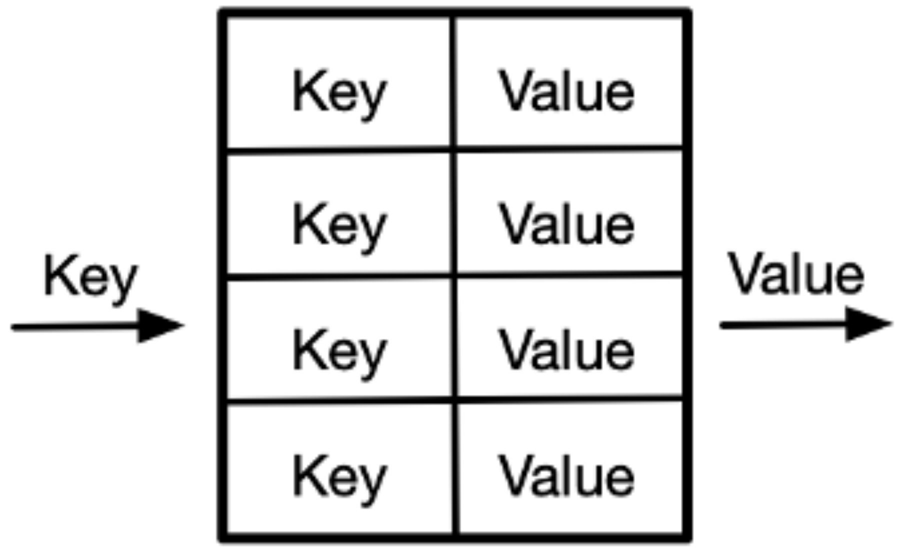
Set



Stack



Queue



Map

Features

Persistent or thread-safe?

Methods for sorting?

Methods for reversing?

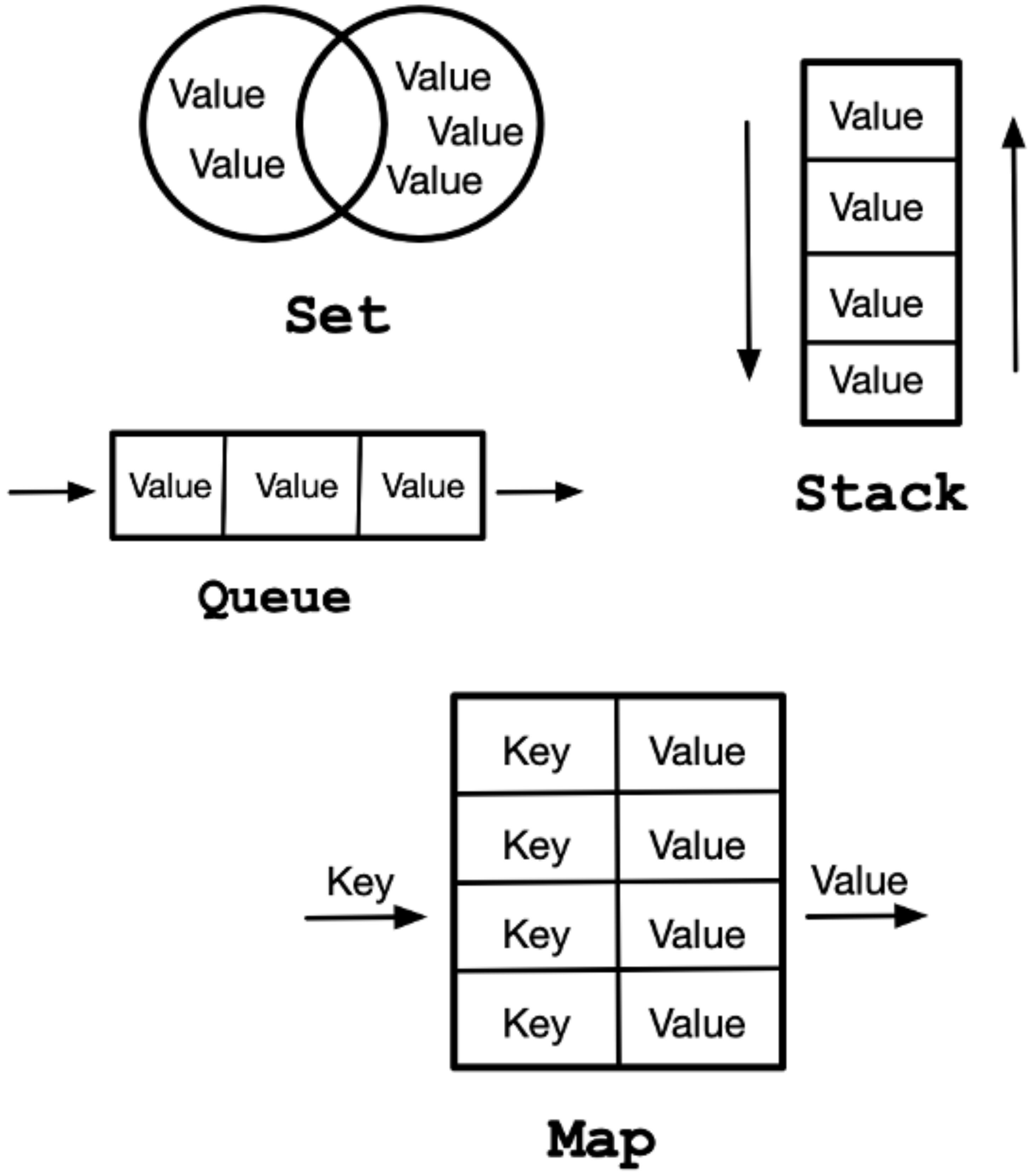
Filter a collection?

Transform collections?

Introduction

Collections (72)

API Features



Features

Persistent or thread-safe?

Methods for sorting?

Methods for reversing?

Filter a collection?

Transform collections?

Use of Features

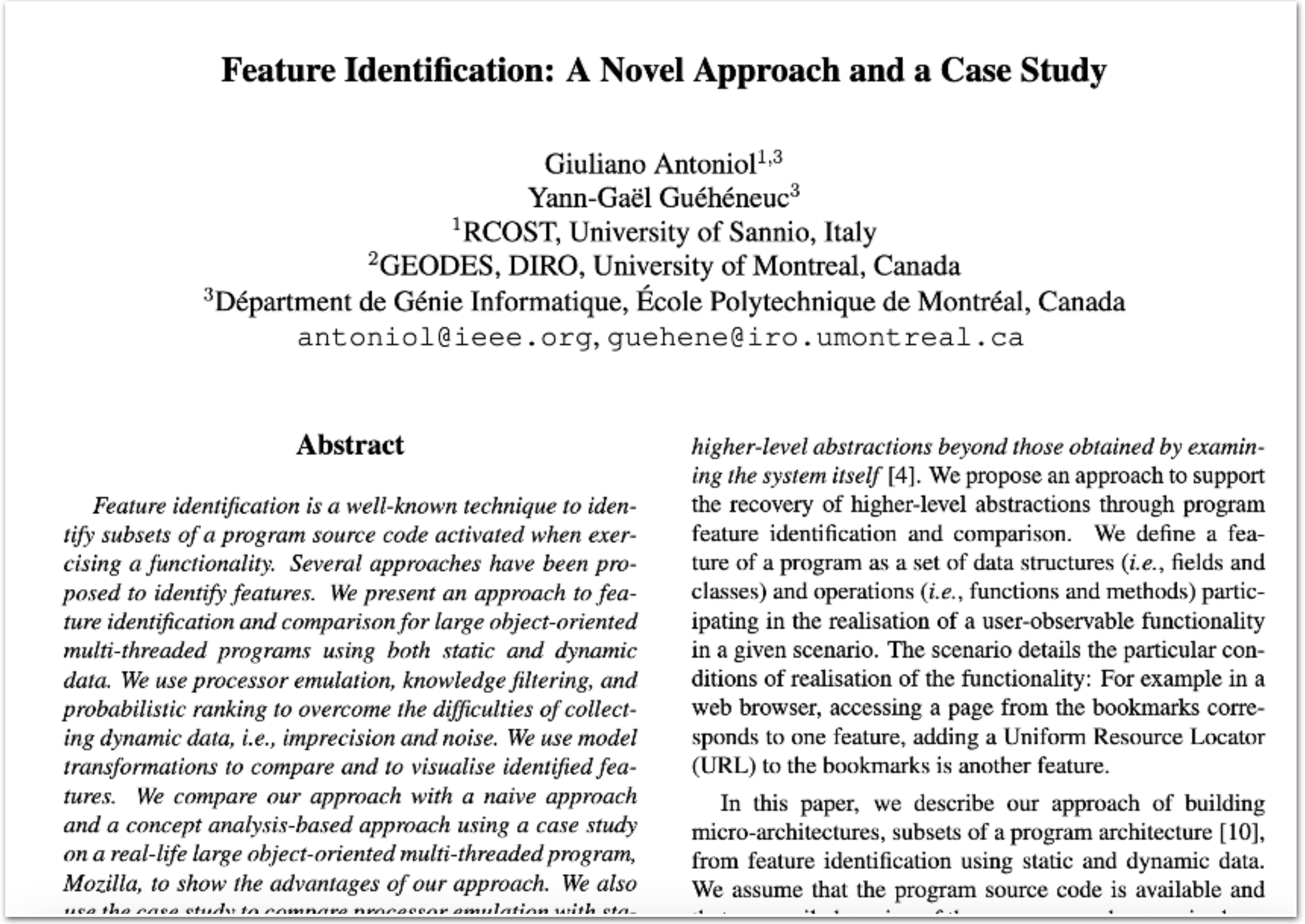
Single method call?

Group of method calls?

Static or instance method calls?

How do libraries compare feature-wise?

Introduction



Feature Identification: A Novel Approach and a Case Study

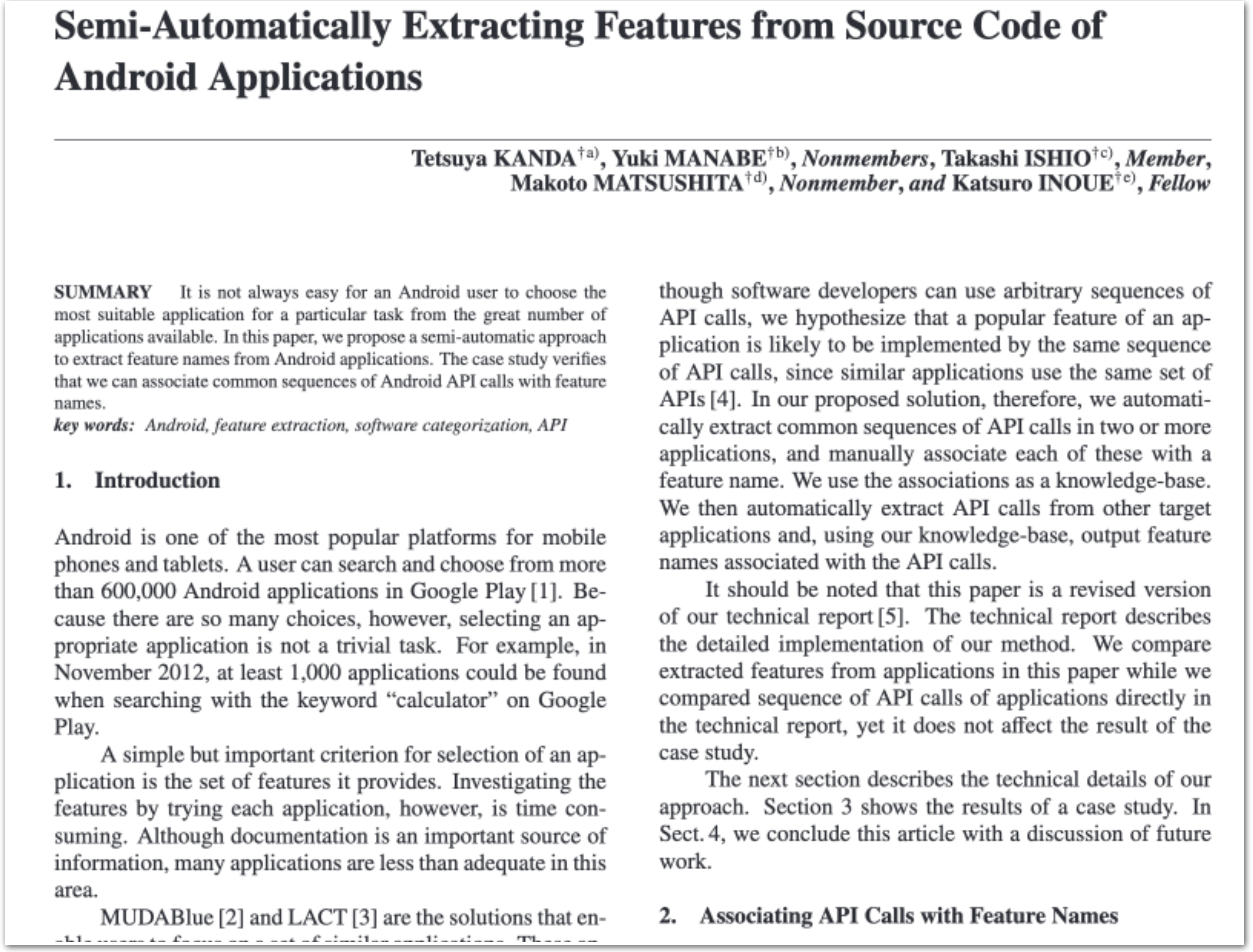
Giuliano Antoniol[1,3]
Yann-Gaël Guéhéneuc[3]
[1]RCOST, University of Sannio, Italy
[2]GEODES, DIRO, University of Montreal, Canada
[3]Départment de Génie Informatique, École Polytechnique de Montréal, Canada
antoniol@ieee.org, guehene@iro.umontreal.ca

Abstract

Feature identification is a well-known technique to identify subsets of a program source code activated when exercising a functionality. Several approaches have been proposed to identify features. We present an approach to feature identification and comparison for large object-oriented multi-threaded programs using both static and dynamic data. We use processor emulation, knowledge filtering, and probabilistic ranking to overcome the difficulties of collecting dynamic data, i.e., imprecision and noise. We use model transformations to compare and to visualise identified features. We compare our approach with a naive approach and a concept analysis-based approach using a case study on a real-life large object-oriented multi-threaded program, Mozilla, to show the advantages of our approach. We also use the case study to compare processor emulation with sta-

higher-level abstractions beyond those obtained by examining the system itself [4]. We propose an approach to support the recovery of higher-level abstractions through program feature identification and comparison. We define a feature of a program as a set of data structures (*i.e.*, fields and classes) and operations (*i.e.*, functions and methods) participating in the realisation of a user-observable functionality in a given scenario. The scenario details the particular conditions of realisation of the functionality: For example in a web browser, accessing a page from the bookmarks corresponds to one feature, adding a Uniform Resource Locator (URL) to the bookmarks is another feature.

In this paper, we describe our approach of building micro-architectures, subsets of a program architecture [10], from feature identification using static and dynamic data. We assume that the program source code is available and



Semi-Automatically Extracting Features from Source Code of Android Applications

Tetsuya KANDA[†a)], **Yuki MANABE**[†b)], ***Nonmembers***, **Takashi ISHIO**[†c)], ***Member***,
Makoto MATSUSHITA[†d)], ***Nonmember***, ***and*** **Katsuro INOUE**[†e)], ***Fellow***

SUMMARY It is not always easy for an Android user to choose the most suitable application for a particular task from the great number of applications available. In this paper, we propose a semi-automatic approach to extract feature names from Android applications. The case study verifies that we can associate common sequences of Android API calls with feature names.

key words: *Android, feature extraction, software categorization, API*

1. Introduction

Android is one of the most popular platforms for mobile phones and tablets. A user can search and choose from more than 600,000 Android applications in Google Play [1]. Because there are so many choices, however, selecting an appropriate application is not a trivial task. For example, in November 2012, at least 1,000 applications could be found when searching with the keyword “calculator” on Google Play.

A simple but important criterion for selection of an application is the set of features it provides. Investigating the features by trying each application, however, is time consuming. Although documentation is an important source of information, many applications are less than adequate in this area.

MUDABlue [2] and LACT [3] are the solutions that en-

though software developers can use arbitrary sequences of API calls, we hypothesize that a popular feature of an application is likely to be implemented by the same sequence of API calls, since similar applications use the same set of APIs [4]. In our proposed solution, therefore, we automatically extract common sequences of API calls in two or more applications, and manually associate each of these with a feature name. We use the associations as a knowledge-base. We then automatically extract API calls from other target applications and, using our knowledge-base, output feature names associated with the API calls.

It should be noted that this paper is a revised version of our technical report [5]. The technical report describes the detailed implementation of our method. We compare extracted features from applications in this paper while we compared sequence of API calls of applications directly in the technical report, yet it does not affect the result of the case study.

The next section describes the technical details of our approach. Section 3 shows the results of a case study. In Sect. 4, we conclude this article with a discussion of future work.

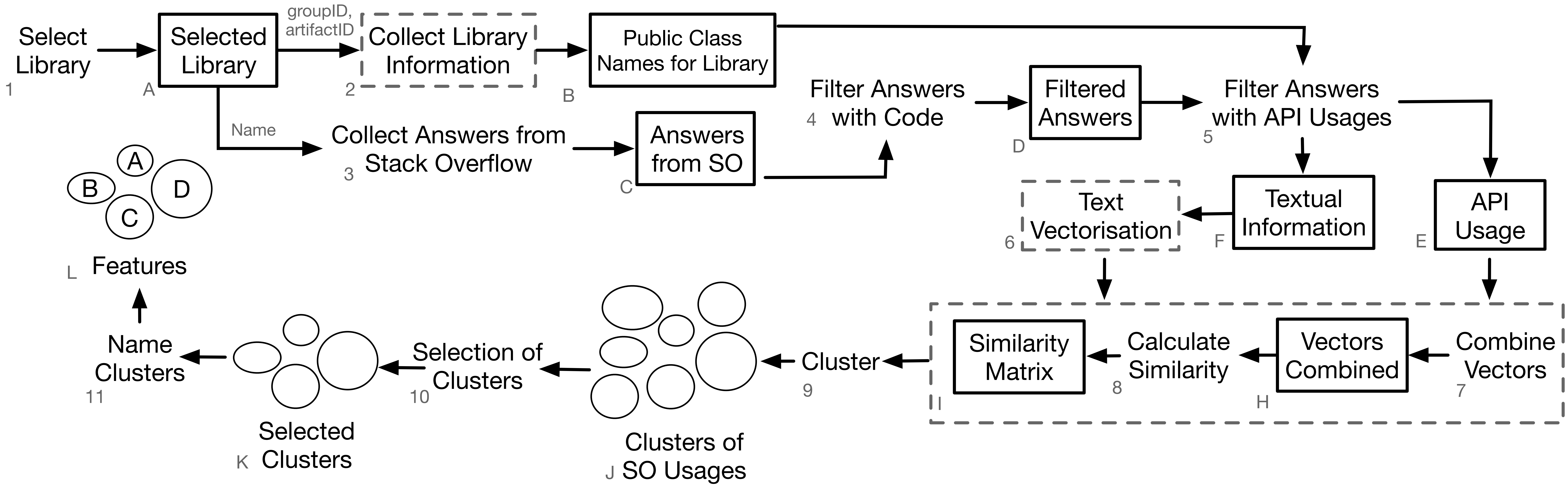
2. Associating API Calls with Feature Names

“…a set of data structures (i.e., fields and classes) and operations (i.e., functions and methods) participating in the realisation of a (…) functionality.”

“…a set of API calls with a corresponding name.”

[1] - Antoniol, G. & Guéhéneuc, Y.-G. Feature Identification: A Novel Approach and a Case Study. *21st Ieee Int Conf Softw Maintenance Icsm’05* 1–10 (2005).
[2] - Kanda, T., Manabe, Y., Ishio, T., Matsushita, M. & Inoue, K. Semi-Automatically Extracting Features from Source Code of Android Applications. *Ieice T Inf Syst*, 2857–2859 (2013).

Approach



Select Library 1

Selected Library A

groupID, artifactID

Collect Library Information 2

Public Class Names for Library B

Name

Collect Answers from Stack Overflow 3

Answers from SO C

Filter Answers with Code 4

Filtered Answers D

Filter Answers with API Usages 5

API Usage E

Textual Information F

Text Vectorisation 6

Combine Vectors 7

Vectors Combined H

Calculate Similarity 8

Similarity Matrix I

Cluster 9

Clusters of SO Usages J

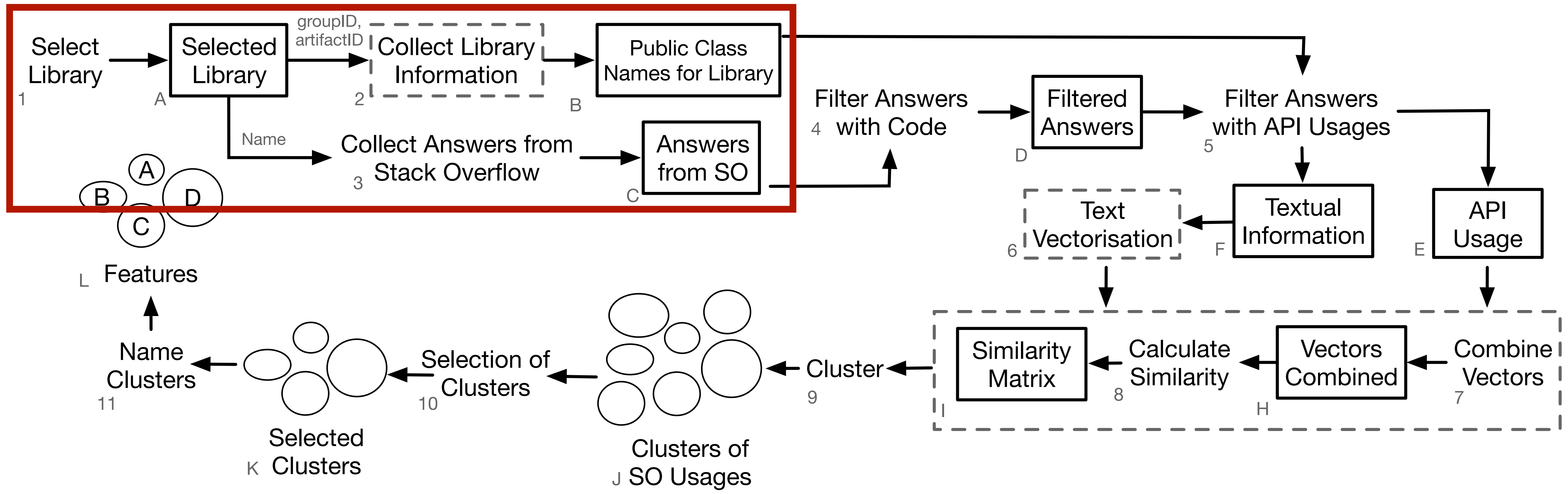
Selection of Clusters 10

Selected Clusters K

Name Clusters 11

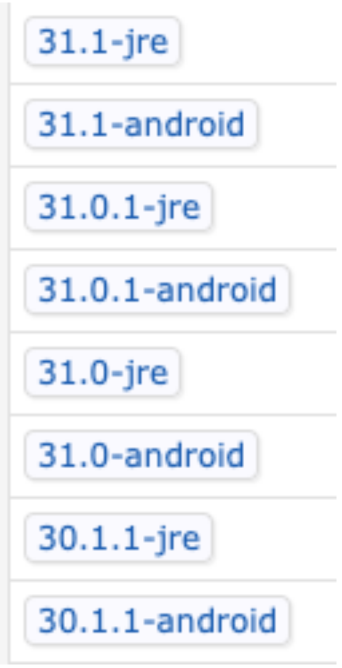
Features L

Approach



Guava

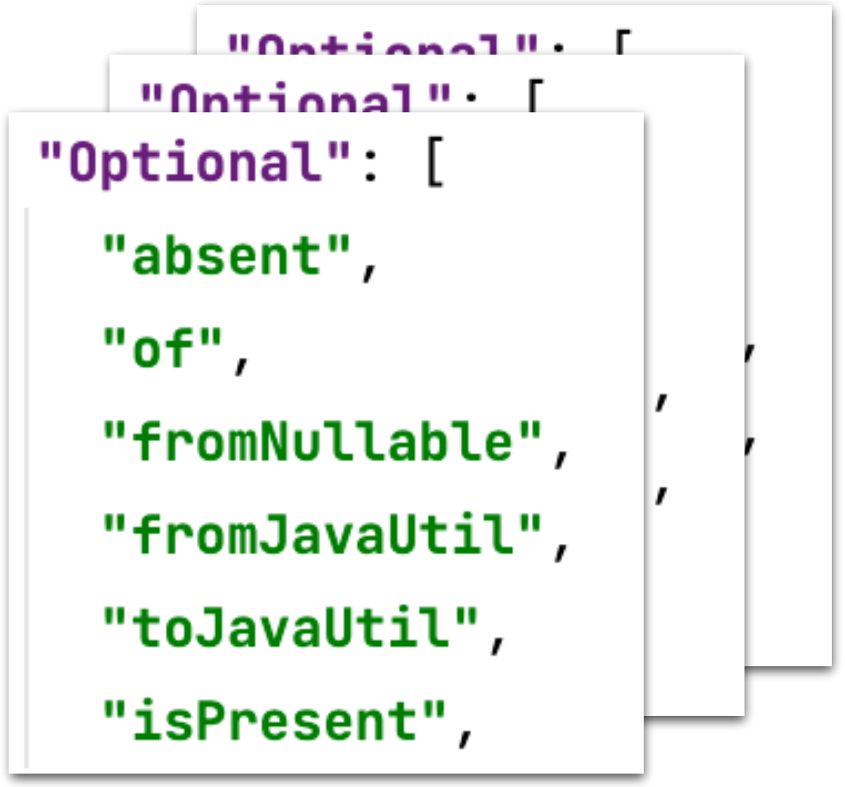
Home » com.google.guava » guava



31.1-jre
31.1-android
31.0.1-jre
31.0.1-android
31.0-jre
31.0-android
30.1.1-jre
30.1.1-android

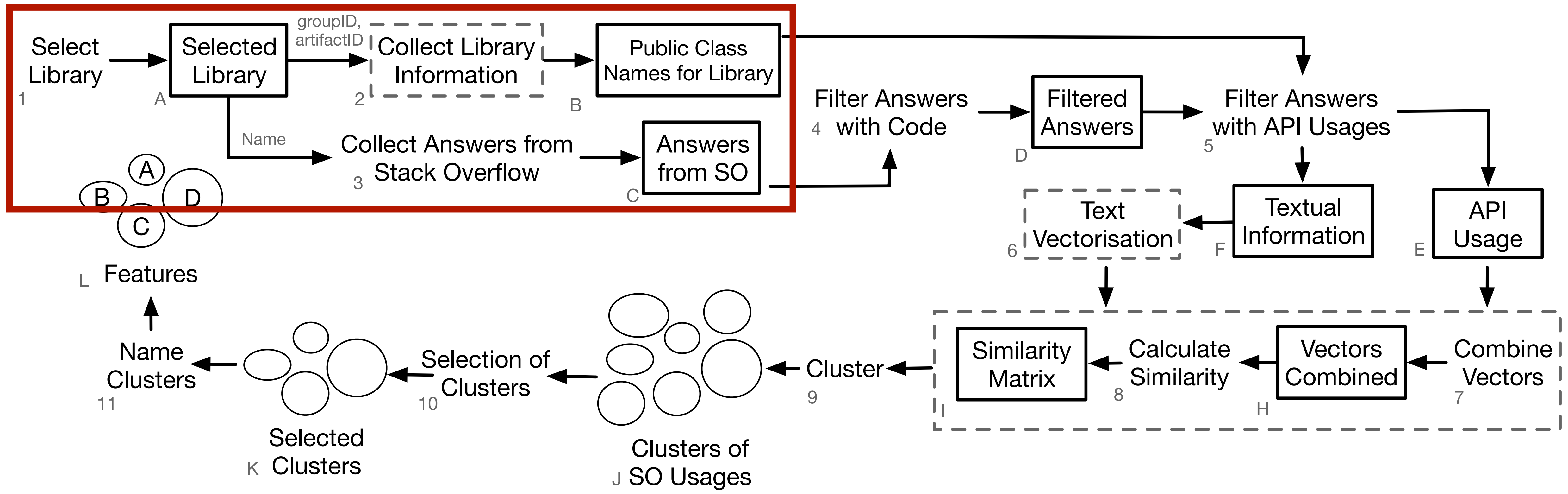
⋮

Apache BCEL



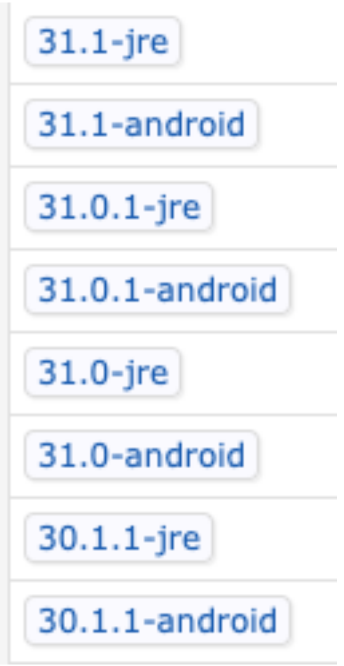
```
"Optional": [
  "absent",
  "of",
  "fromNullable",
  "fromJavaUtil",
  "toJavaUtil",
  "isPresent",
```

Approach



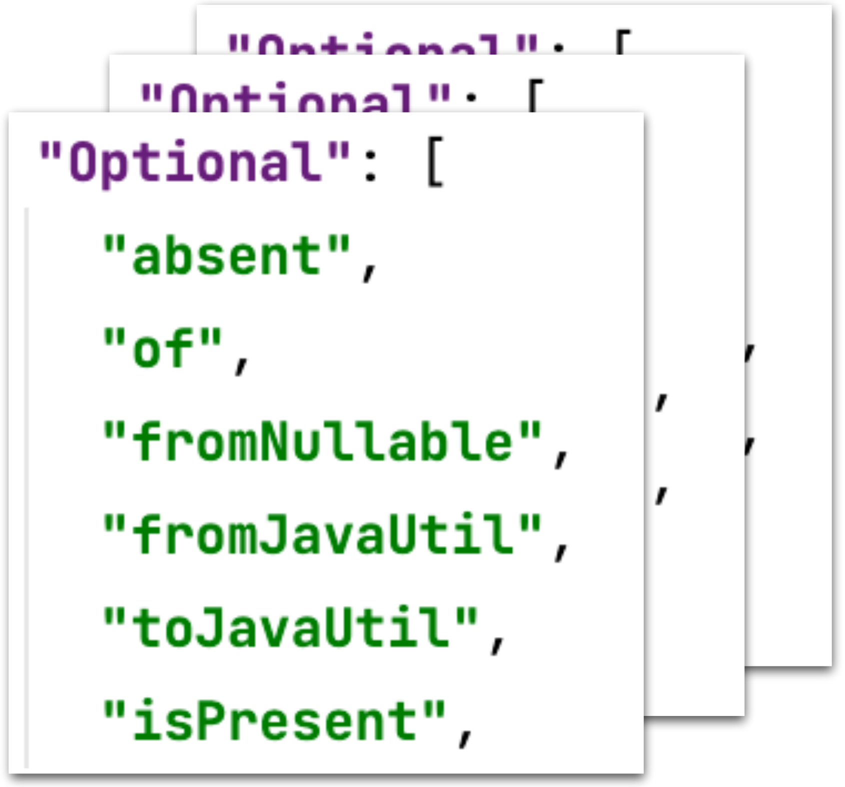
Guava

Home » com.google.guava » guava


31.1-jre
31.1-android
31.0.1-jre
31.0.1-android
31.0-jre
31.0-android
30.1.1-jre
30.1.1-android

⋮

Apache BCEL

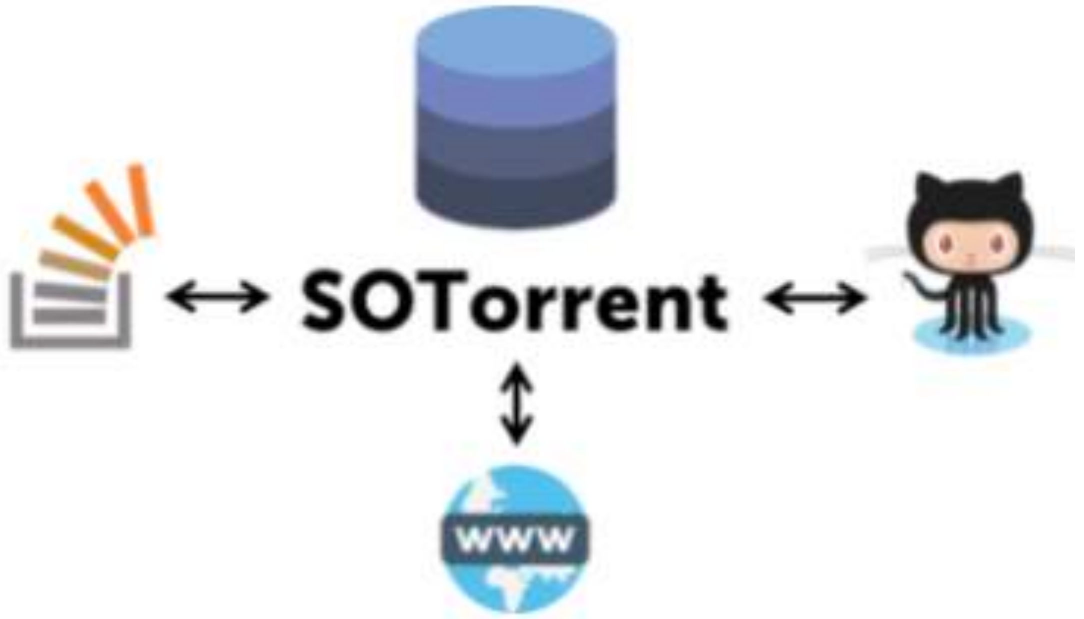


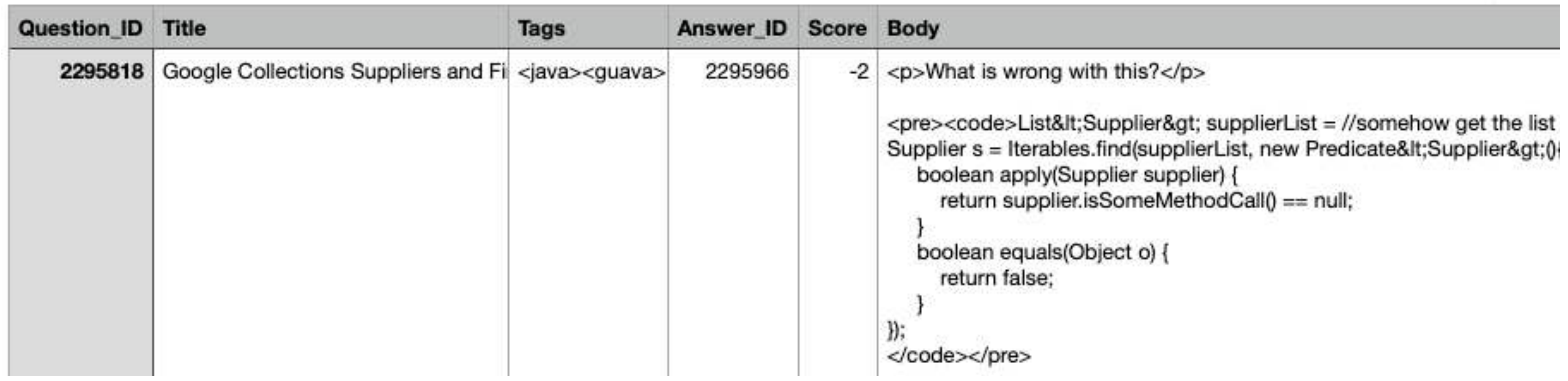
```
"Optional": [
  "absent",
  "of",
  "fromNullable",
  "fromJavaUtil",
  "toJavaUtil",
  "isPresent",
```

stackoverflow

Questions tagged [guava]

Google's Core Java Library for Java and Android development.

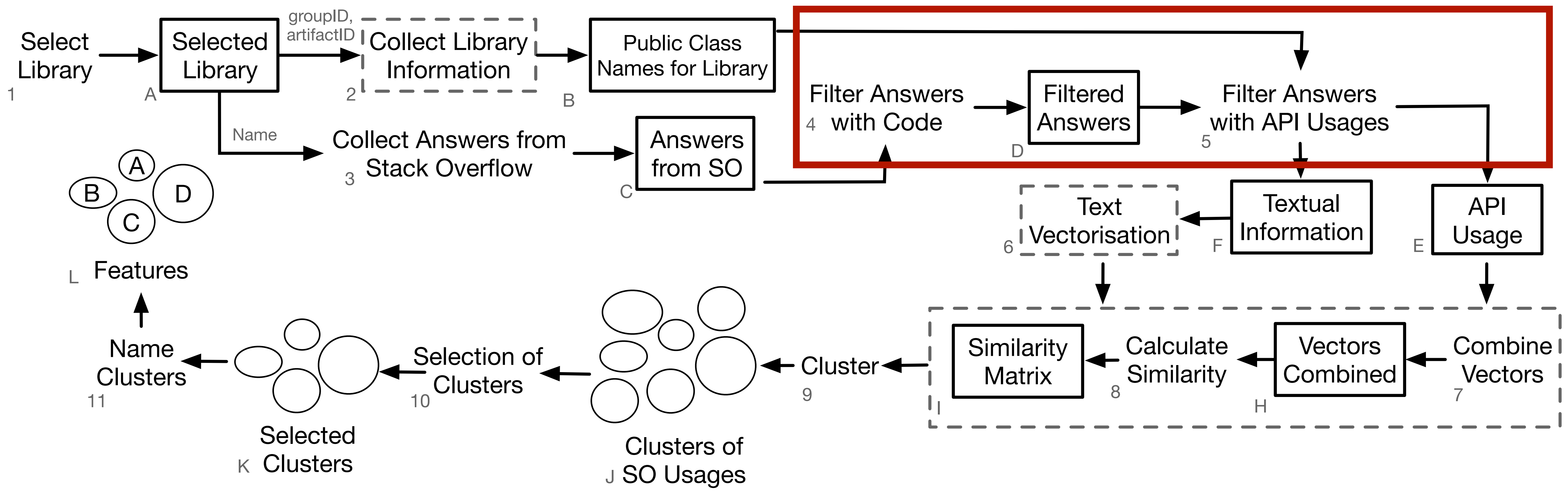


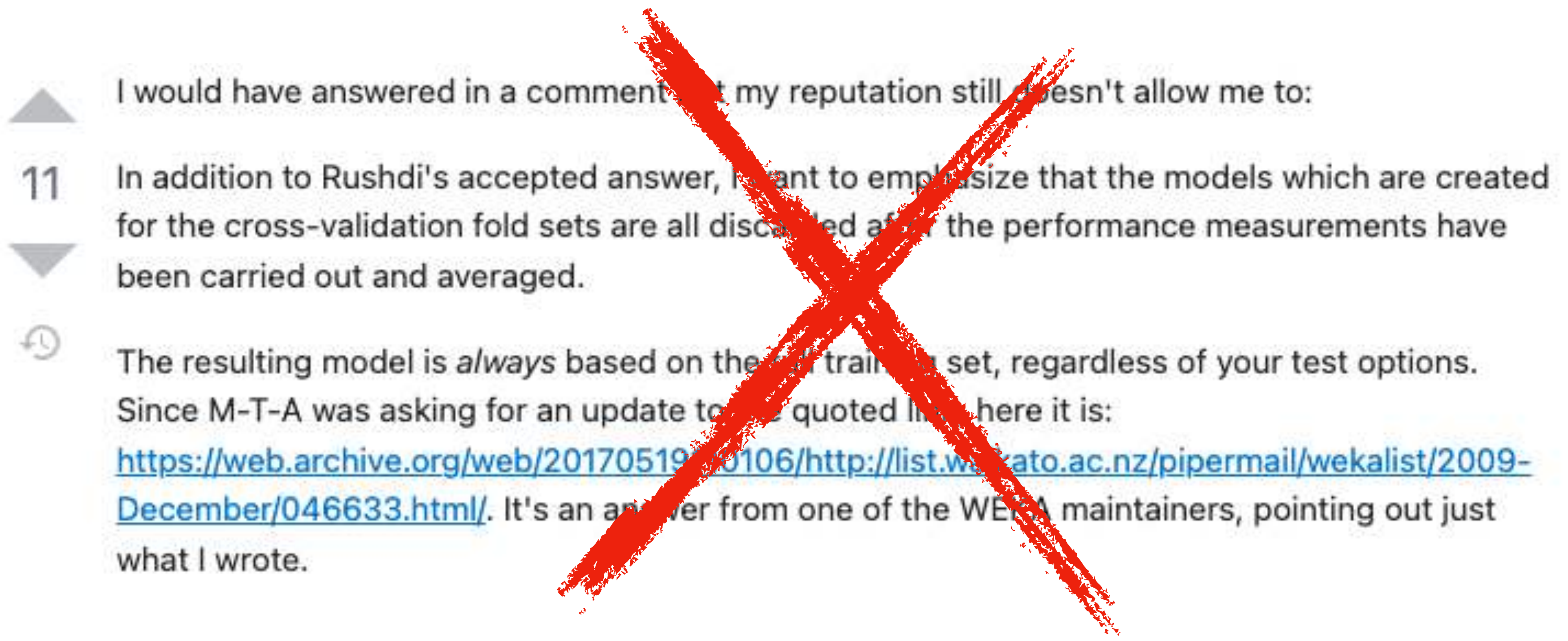


Question_ID	Title	Tags	Answer_ID	Score	Body

2295818	Google Collections Suppliers and Fi	<java><guava>	2295966	-2	<p>What is wrong with this?</p>
					
					<pre><code>List&lt;Supplier&gt; supplierList = //somehow get the list
					Supplier s = Iterables.find(supplierList, new Predicate&lt;Supplier&gt;()
					boolean apply(Supplier supplier) {
					return supplier.isSomeMethodCall() == null;
					}
					boolean equals(Object o) {
					return false;
					}
					});
					</code></pre>

Approach



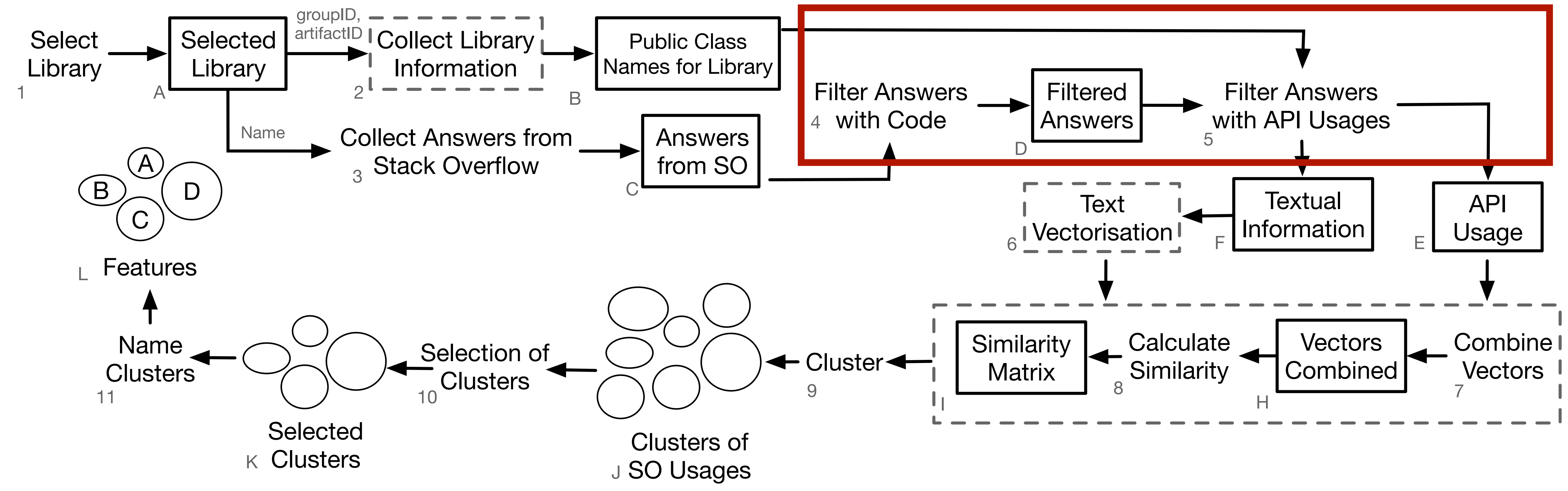


I would have answered in a comment, but my reputation still doesn't allow me to:

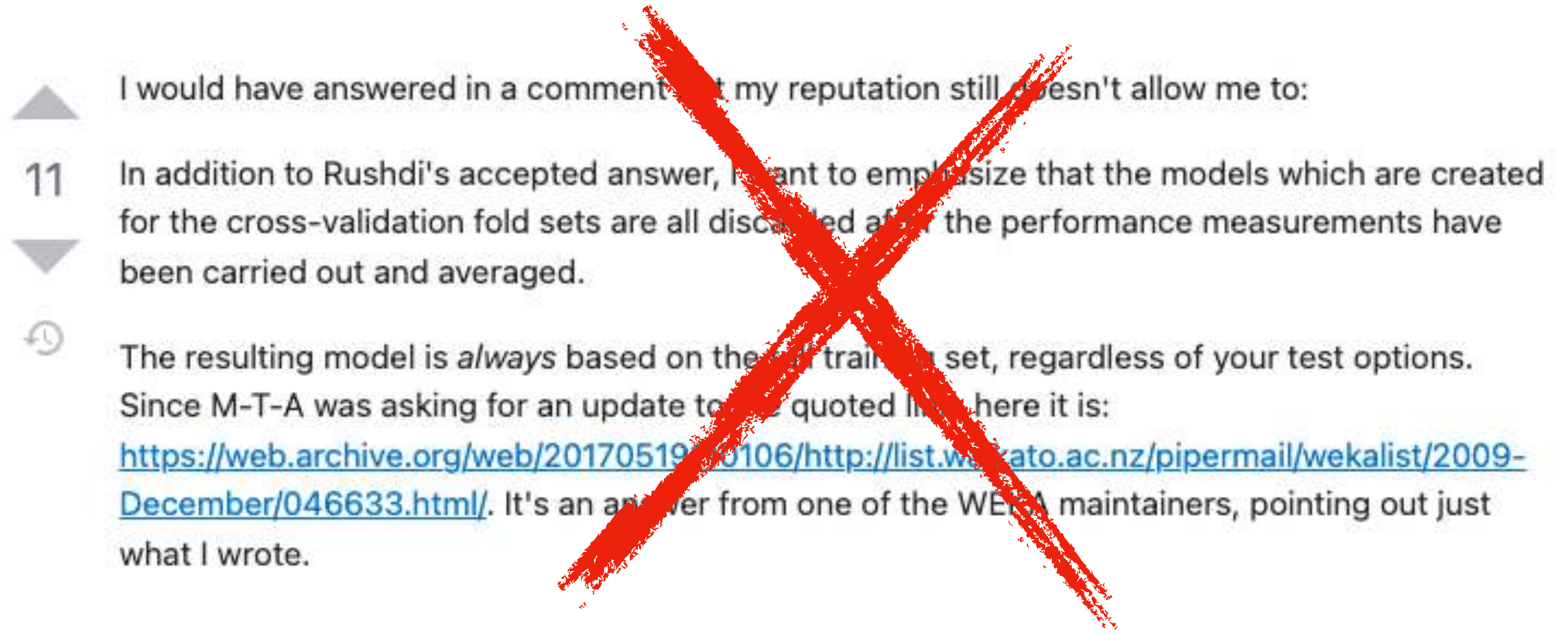
In addition to Rushdi's accepted answer, I want to emphasize that the models which are created for the cross-validation fold sets are all discarded after the performance measurements have been carried out and averaged.

The resulting model is *always* based on the full training set, regardless of your test options. Since M-T-A was asking for an update to the quoted link, here it is: https://web.archive.org/web/20170519110106/http://list.waikato.ac.nz/pipermail/wekalist/2009-December/046633.html/. It's an answer from one of the WEKA maintainers, pointing out just what I wrote.

Approach



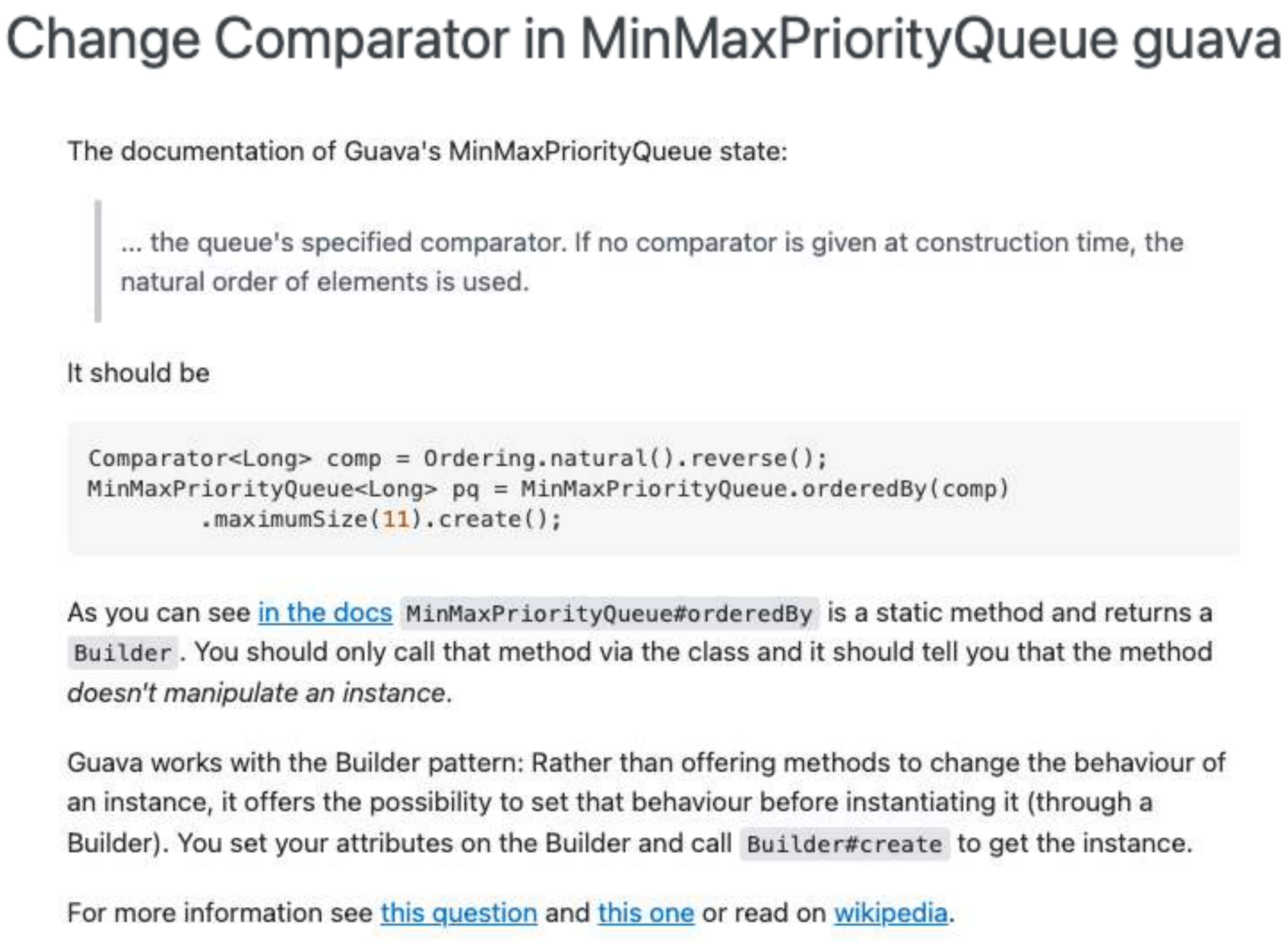
Select Library 1
Selected Library A
groupID, artifactID
Collect Library Information 2
Public Class Names for Library B
Name
Collect Answers from Stack Overflow 3
Answers from SO C
Filter Answers with Code 4
Filtered Answers D
Filter Answers with API Usages 5
Textual Information F
API Usage E
Text Vectorisation 6
Combine Vectors 7
Vectors Combined H
Calculate Similarity 8
Similarity Matrix I
Cluster 9
Clusters of SO Usages J
Selection of Clusters 10
Selected Clusters K
Name Clusters 11
Features L



~~I would have answered in a comment but my reputation still doesn't allow me to:~~

~~In addition to Rushdi's accepted answer, I want to emphasize that the models which are created for the cross-validation fold sets are all discarded after the performance measurements have been carried out and averaged.~~

~~The resulting model is *always* based on the full training set, regardless of your test options. Since M-T-A was asking for an update to the quoted link, here it is: https://web.archive.org/web/20170519[illegible]0106/http://list.waikato.ac.nz/pipermail/wekalist/2009-December/046633.html/. It's an answer from one of the WEKA maintainers, pointing out just what I wrote.~~



Change Comparator in MinMaxPriorityQueue guava

The documentation of Guava's MinMaxPriorityQueue state:

> ... the queue's specified comparator. If no comparator is given at construction time, the natural order of elements is used.

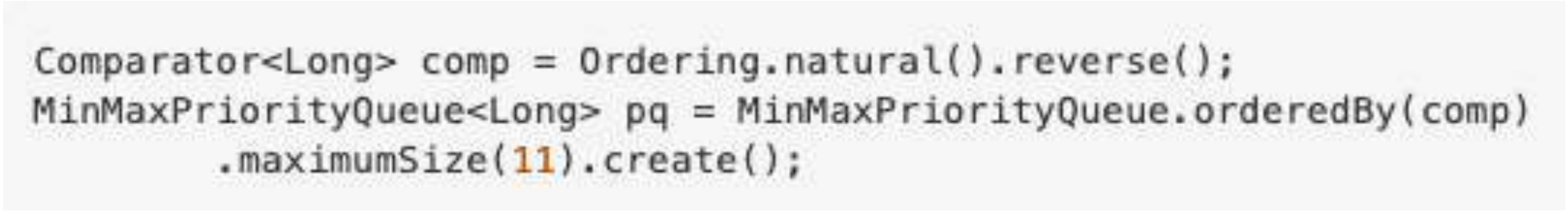
It should be

```
Comparator<Long> comp = Ordering.natural().reverse();
MinMaxPriorityQueue<Long> pq = MinMaxPriorityQueue.orderedBy(comp)
        .maximumSize(11).create();
```

As you can see in the docs `MinMaxPriorityQueue#orderedBy` is a static method and returns a `Builder`. You should only call that method via the class and it should tell you that the method *doesn't manipulate an instance.*

Guava works with the Builder pattern: Rather than offering methods to change the behaviour of an instance, it offers the possibility to set that behaviour before instantiating it (through a Builder). You set your attributes on the Builder and call `Builder#create` to get the instance.

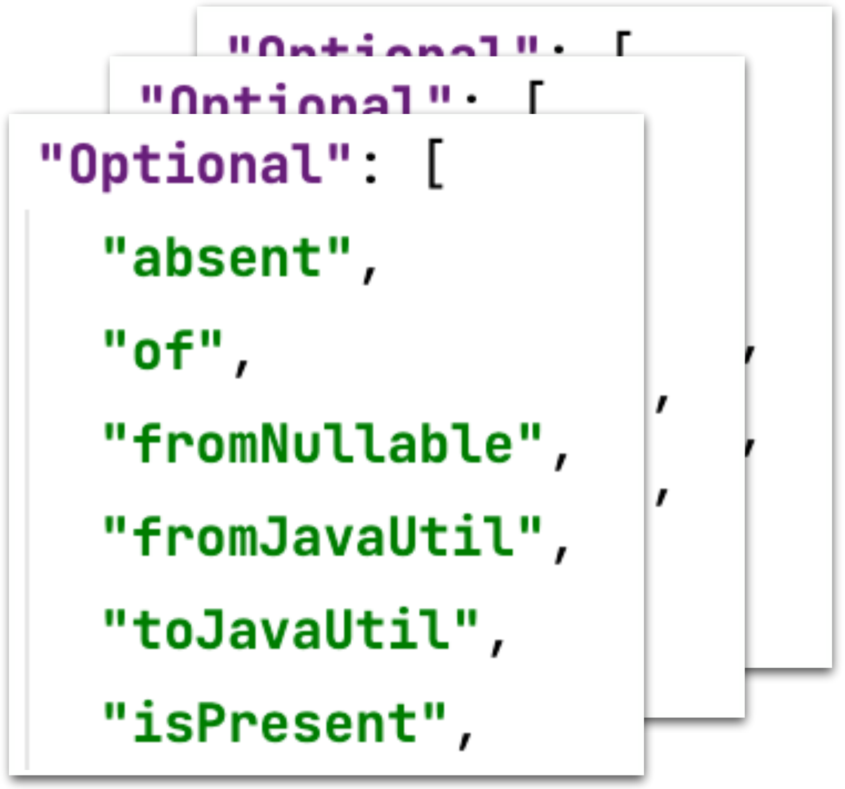
For more information see this question and this one or read on wikipedia.



```
Comparator<Long> comp = Ordering.natural().reverse();
MinMaxPriorityQueue<Long> pq = MinMaxPriorityQueue.orderedBy(comp)
        .maximumSize(11).create();
```

`MinMaxPriorityQueue#orderedBy`

`Builder#create`

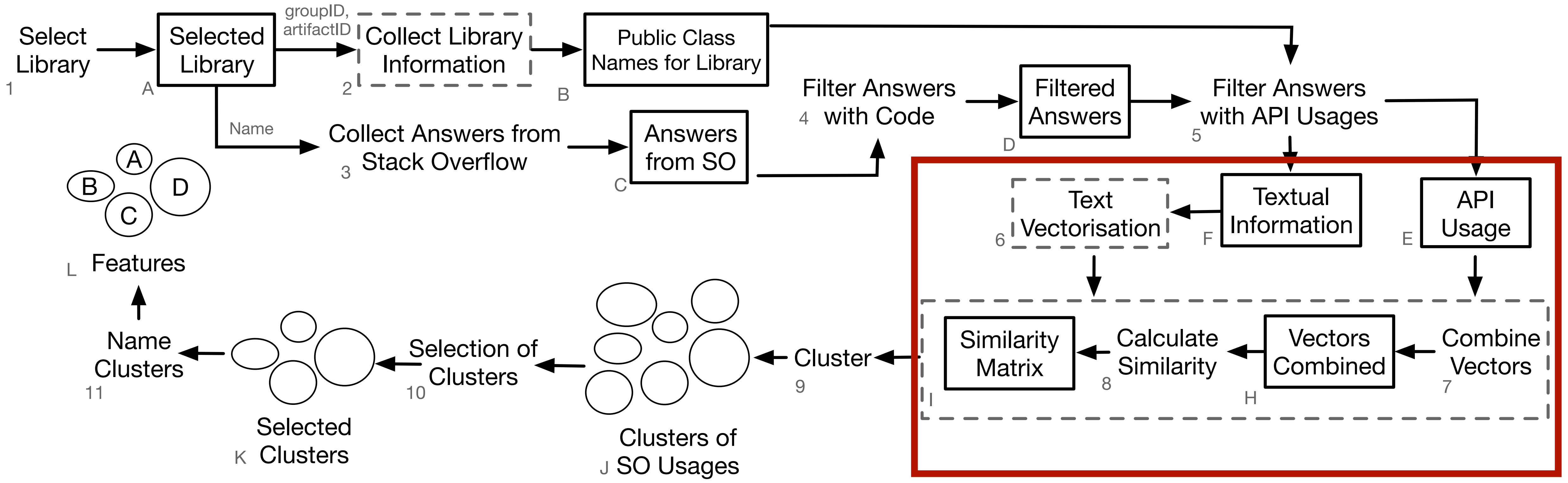


```
"Optional": [
  "absent",
  "of",
  "fromNullable",
  "fromJavaUtil",
  "toJavaUtil",
  "isPresent",
```

`Ordering.natural.reverse`

`MinMaxPriorityQueue.orderedBy.maximumSize.create`

Approach



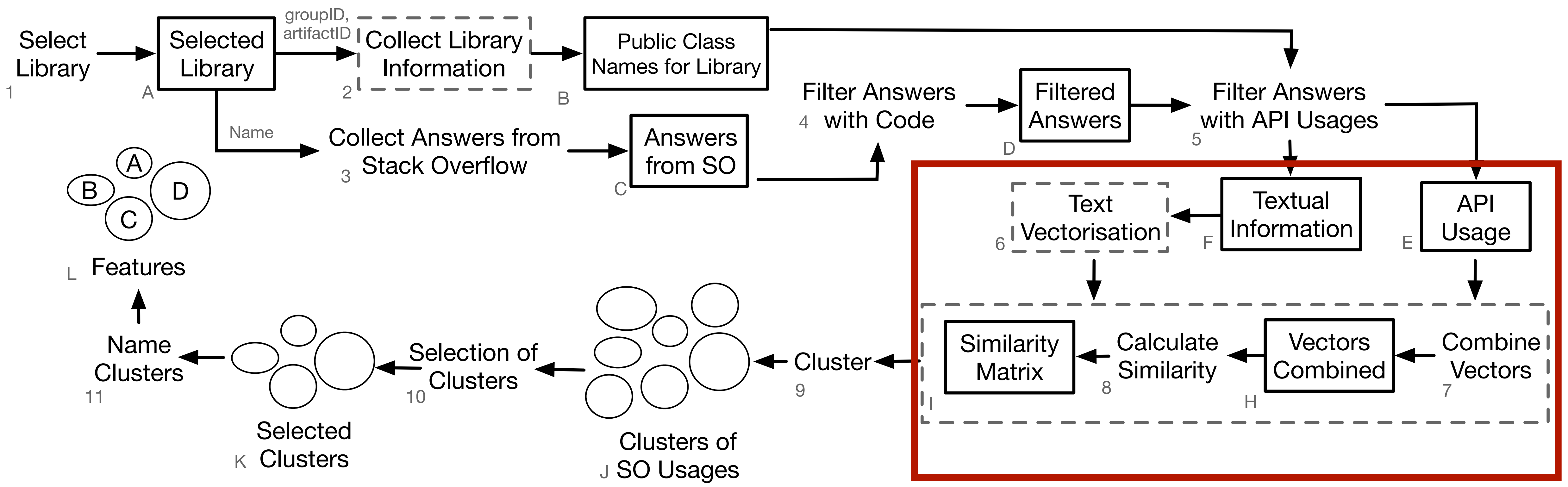
Textual Information

- Post Titles
- Body of the Question
- Body of the Answer

API Usage Information

- Method Names
- Method Names split by Camel Case
- API Calls

Approach

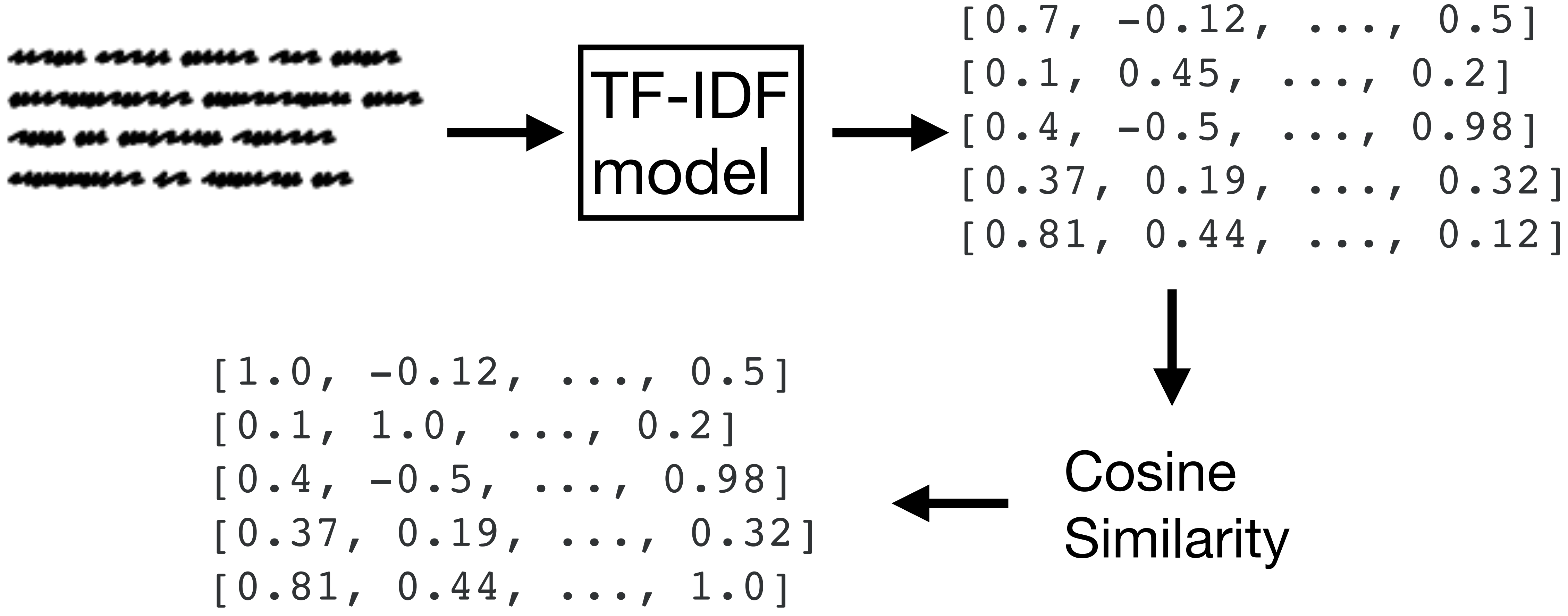


Textual Information

- Post Titles
- Body of the Question
- Body of the Answer

API Usage Information

- Method Names
- Method Names split by Camel Case
- API Calls

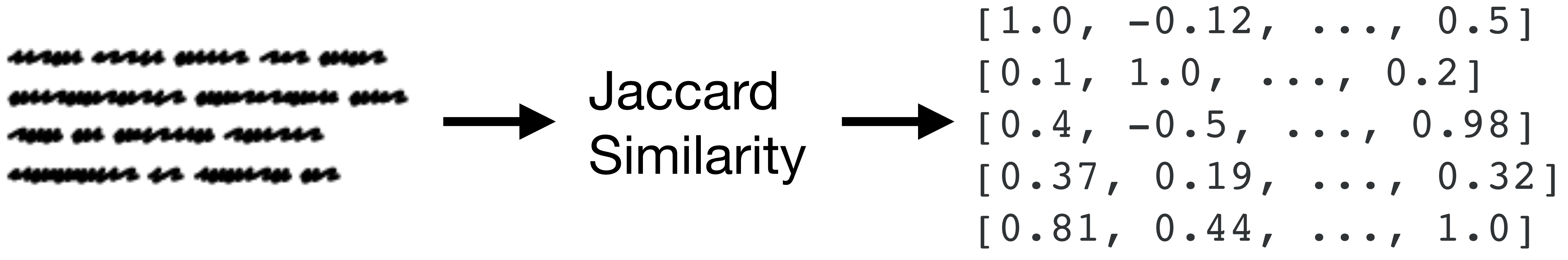


TF-IDF model

```
[0.7, -0.12, ..., 0.5]
[0.1, 0.45, ..., 0.2]
[0.4, -0.5, ..., 0.98]
[0.37, 0.19, ..., 0.32]
[0.81, 0.44, ..., 0.12]
```

Cosine Similarity

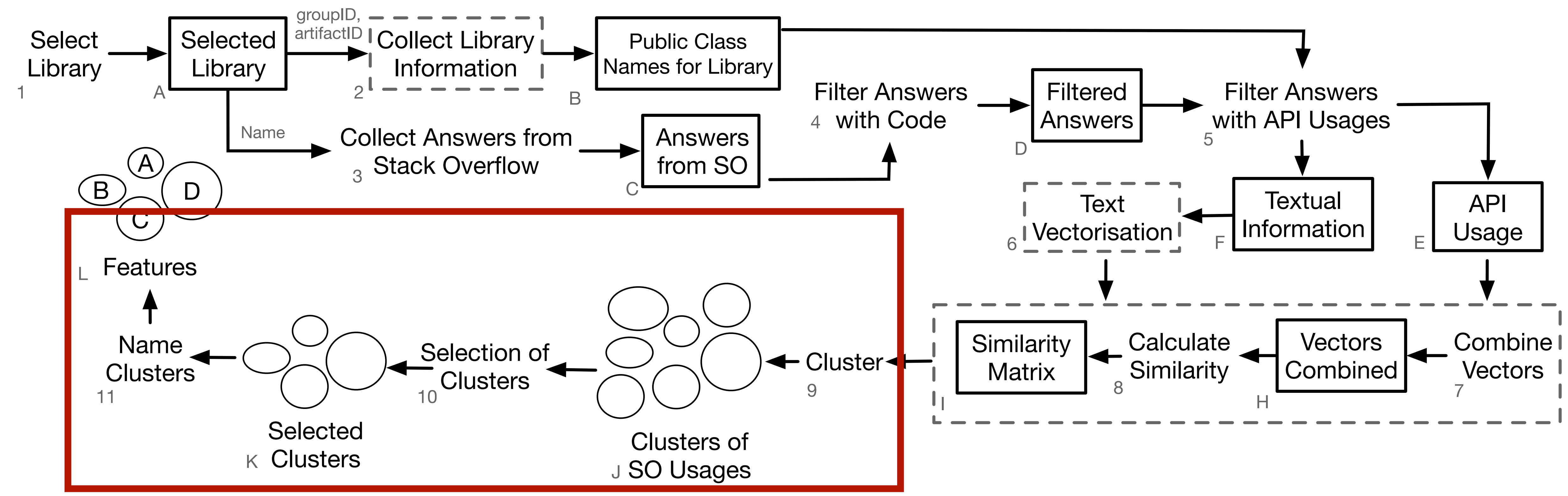
```
[1.0, -0.12, ..., 0.5]
[0.1, 1.0, ..., 0.2]
[0.4, -0.5, ..., 0.98]
[0.37, 0.19, ..., 0.32]
[0.81, 0.44, ..., 1.0]
```



Jaccard Similarity

```
[1.0, -0.12, ..., 0.5]
[0.1, 1.0, ..., 0.2]
[0.4, -0.5, ..., 0.98]
[0.37, 0.19, ..., 0.32]
[0.81, 0.44, ..., 1.0]
```

Approach



Select Library 1

Selected Library A

groupID, artifactID

Collect Library Information 2

Public Class Names for Library B

Name

Collect Answers from Stack Overflow 3

Answers from SO C

Filter Answers with Code 4

Filtered Answers D

Filter Answers with API Usages 5

Textual Information F

API Usage E

Text Vectorisation 6

Combine Vectors 7

Vectors Combined H

Calculate Similarity 8

Similarity Matrix I

Cluster 9

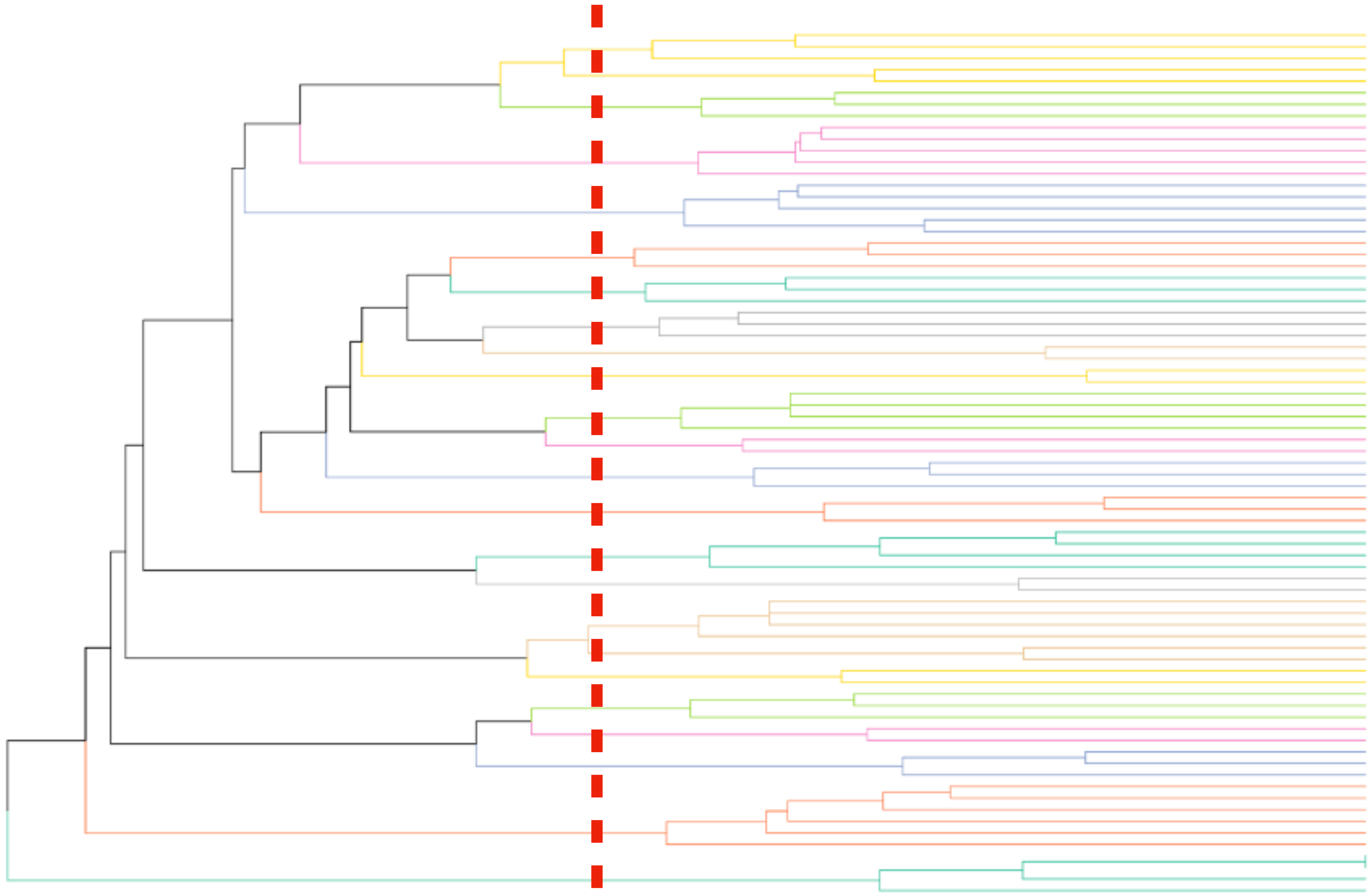
Clusters of SO Usages J

Selection of Clusters 10

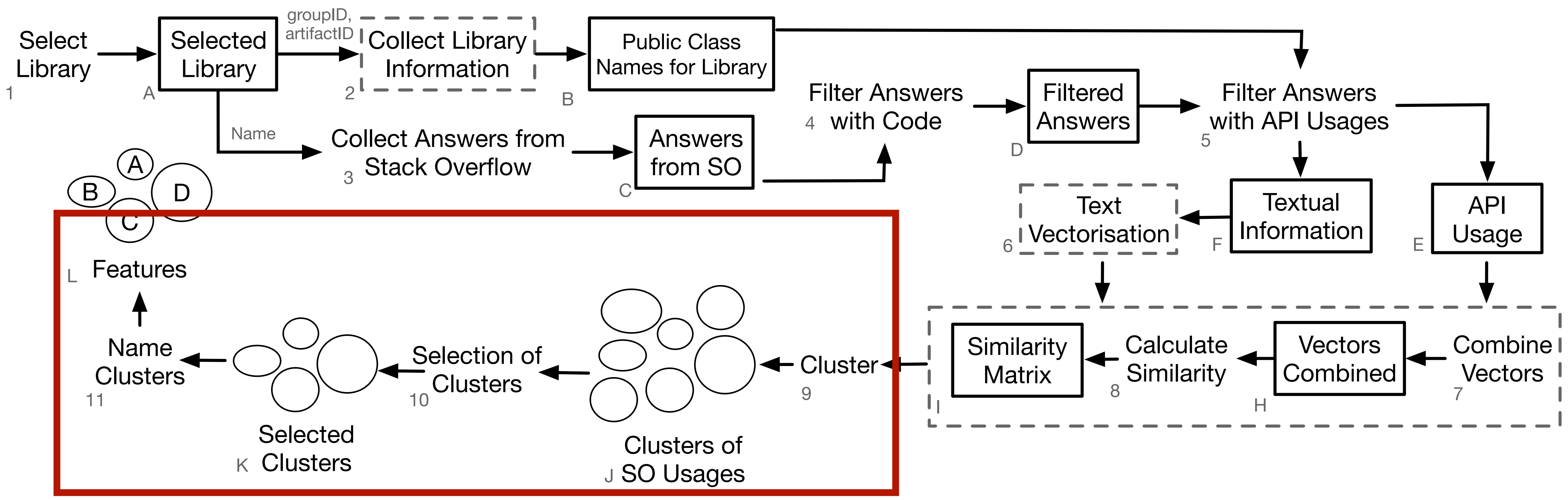
Selected Clusters K

Name Clusters 11

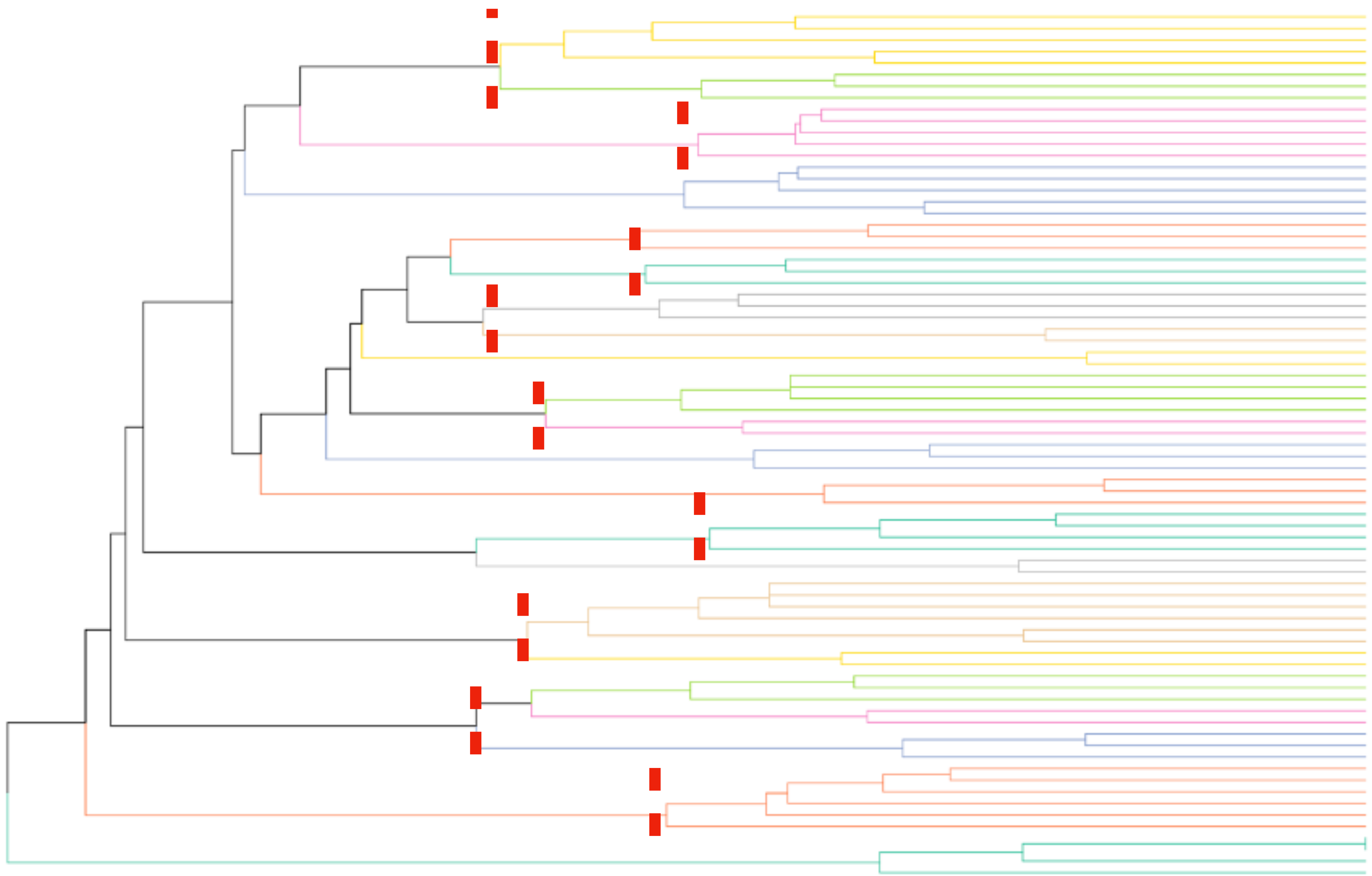
Features L

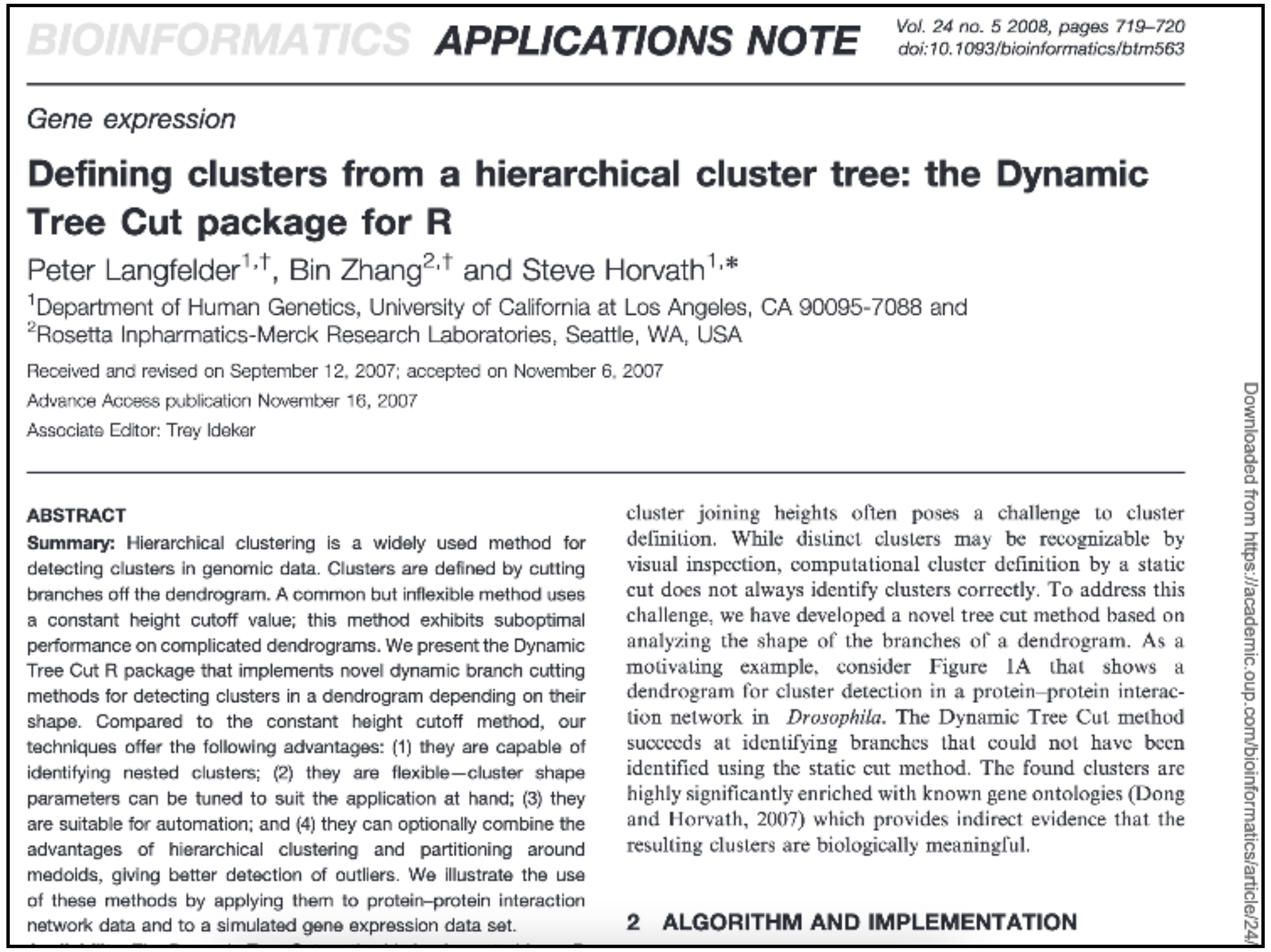


Approach



Select Library 1
Selected Library A
groupID, artifactID
Collect Library Information 2
Public Class Names for Library B
Name
Collect Answers from Stack Overflow 3
Answers from SO C
Filter Answers with Code 4
Filtered Answers D
Filter Answers with API Usages 5
API Usage E
Textual Information F
Text Vectorisation 6
Combine Vectors 7
Vectors Combined H
Calculate Similarity 8
Similarity Matrix I
Cluster 9
Clusters of SO Usages J
Selection of Clusters 10
Selected Clusters K
Name Clusters 11
Features L
A
B
C
D





BIOINFORMATICS **APPLICATIONS NOTE**

Vol. 24 no. 5 2008, pages 719–720
doi:10.1093/bioinformatics/btm563

Gene expression

Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R

Peter Langfelder[1,†], Bin Zhang[2,†] and Steve Horvath[1,*]

[1]Department of Human Genetics, University of California at Los Angeles, CA 90095-7088 and
[2]Rosetta Inpharmatics-Merck Research Laboratories, Seattle, WA, USA

Received and revised on September 12, 2007; accepted on November 6, 2007
Advance Access publication November 16, 2007
Associate Editor: Trey Ideker

ABSTRACT

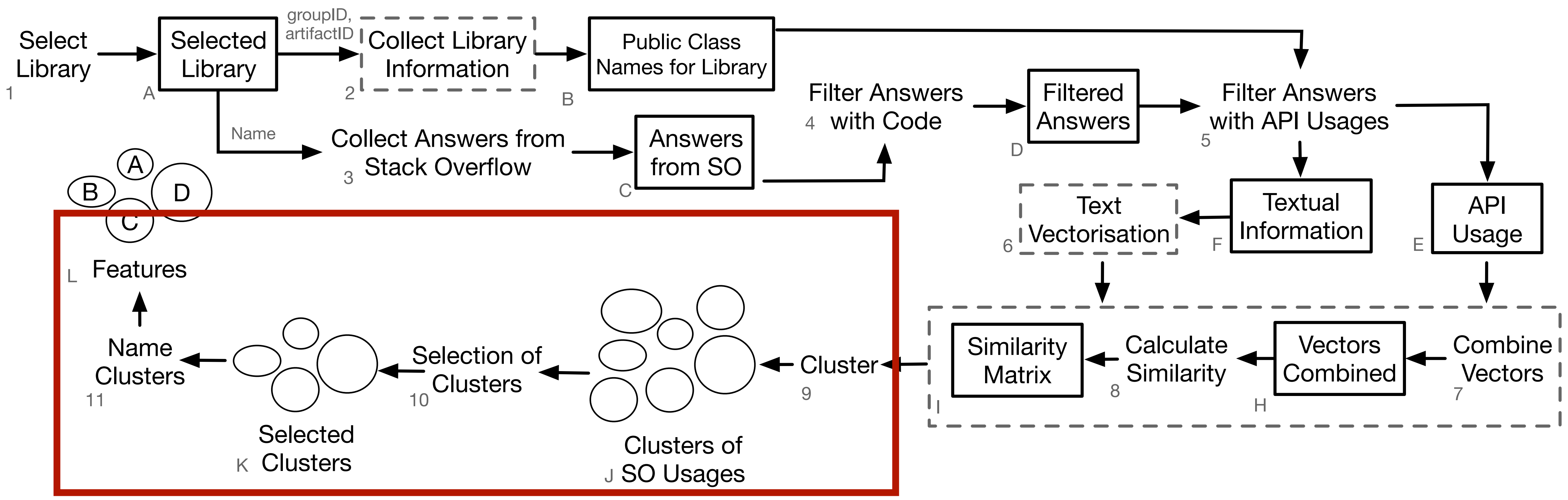
Summary: Hierarchical clustering is a widely used method for detecting clusters in genomic data. Clusters are defined by cutting branches off the dendrogram. A common but inflexible method uses a constant height cutoff value; this method exhibits suboptimal performance on complicated dendrograms. We present the Dynamic Tree Cut R package that implements novel dynamic branch cutting methods for detecting clusters in a dendrogram depending on their shape. Compared to the constant height cutoff method, our techniques offer the following advantages: (1) they are capable of identifying nested clusters; (2) they are flexible—cluster shape parameters can be tuned to suit the application at hand; (3) they are suitable for automation; and (4) they can optionally combine the advantages of hierarchical clustering and partitioning around medoids, giving better detection of outliers. We illustrate the use of these methods by applying them to protein–protein interaction network data and to a simulated gene expression data set.

cluster joining heights often poses a challenge to cluster definition. While distinct clusters may be recognizable by visual inspection, computational cluster definition by a static cut does not always identify clusters correctly. To address this challenge, we have developed a novel tree cut method based on analyzing the shape of the branches of a dendrogram. As a motivating example, consider Figure 1A that shows a dendrogram for cluster detection in a protein–protein interaction network in *Drosophila*. The Dynamic Tree Cut method succeeds at identifying branches that could not have been identified using the static cut method. The found clusters are highly significantly enriched with known gene ontologies (Dong and Horvath, 2007) which provides indirect evidence that the resulting clusters are biologically meaningful.

2 ALGORITHM AND IMPLEMENTATION

[3] - P. Langfelder, B. Zhang, and S. Horvath, “Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R,” *Bioinformatics*, vol. 24, no. 5, pp. 719–720, 2008.

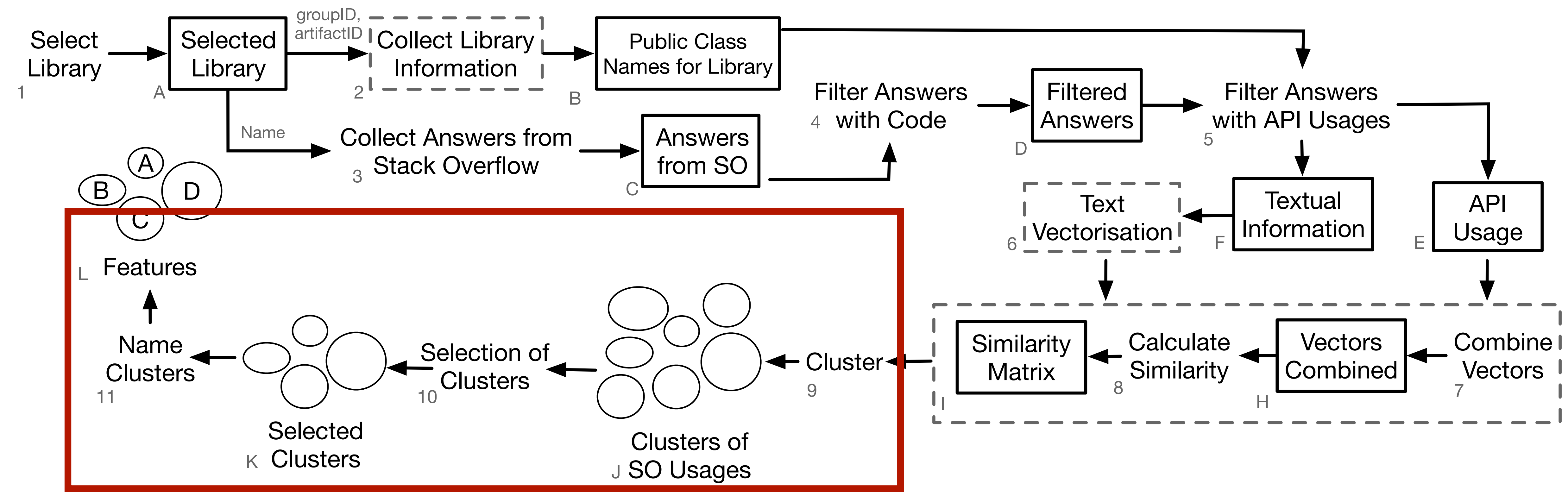
Approach



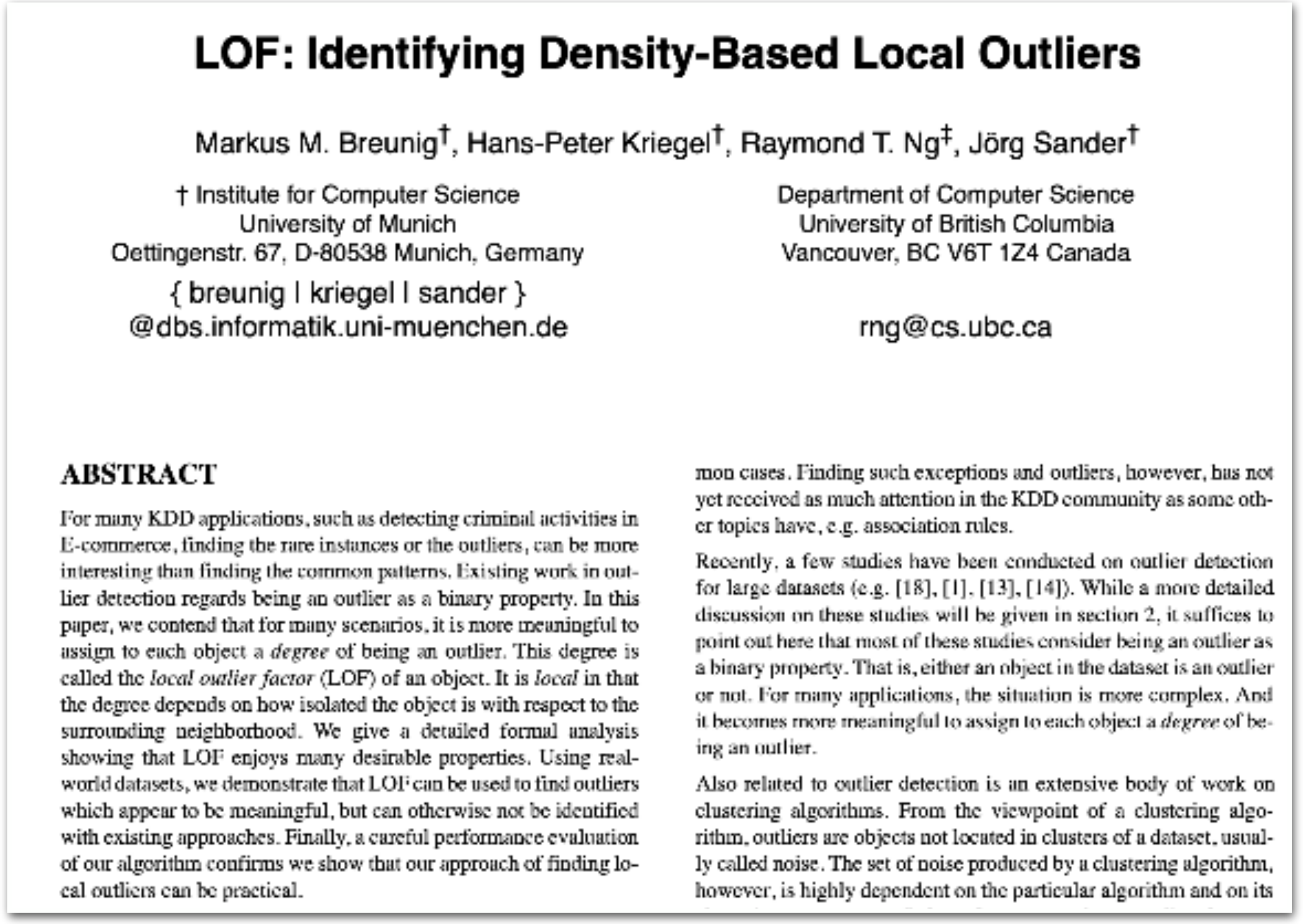
Formed clusters would ideally have frequent API calls, characterising the code-part of the cluster.

[4] - Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104. 2000.
[5] - C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *The 52nd Annual Meeting of the Association for Compu- tational Linguistics: System Demonstrations*, pp. 55–60. 2014.

Approach



Formed clusters would ideally have frequent API calls, characterising the code-part of the cluster.



LOF: Identifying Density-Based Local Outliers

Markus M. Breunig[†], Hans-Peter Kriegel[†], Raymond T. Ng[‡], Jörg Sander[†]

[†] Institute for Computer Science
University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
{ breunig | kriegel | sander }
@dbs.informatik.uni-muenchen.de

Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada
rng@cs.ubc.ca

ABSTRACT

For many KDD applications, such as detecting criminal activities in E-commerce, finding the rare instances or the outliers, can be more interesting than finding the common patterns. Existing work in outlier detection regards being an outlier as a binary property. In this paper, we contend that for many scenarios, it is more meaningful to assign to each object a *degree* of being an outlier. This degree is called the *local outlier factor* (LOF) of an object. It is *local* in that the degree depends on how isolated the object is with respect to the surrounding neighborhood. We give a detailed formal analysis showing that LOF enjoys many desirable properties. Using real-world datasets, we demonstrate that LOF can be used to find outliers which appear to be meaningful, but can otherwise not be identified with existing approaches. Finally, a careful performance evaluation of our algorithm confirms we show that our approach of finding local outliers can be practical.

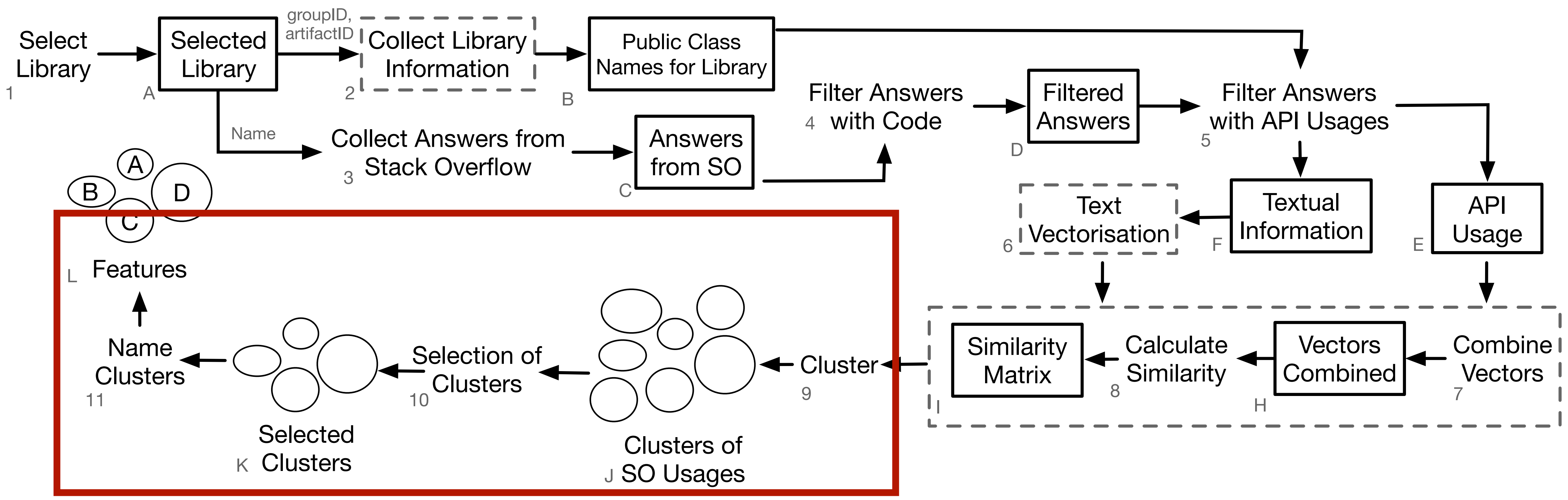
mon cases. Finding such exceptions and outliers, however, has not yet received as much attention in the KDD community as some other topics have, e.g. association rules.

Recently, a few studies have been conducted on outlier detection for large datasets (e.g. [18], [1], [13], [14]). While a more detailed discussion on these studies will be given in section 2, it suffices to point out here that most of these studies consider being an outlier as a binary property. That is, either an object in the dataset is an outlier or not. For many applications, the situation is more complex. And it becomes more meaningful to assign to each object a *degree* of being an outlier.

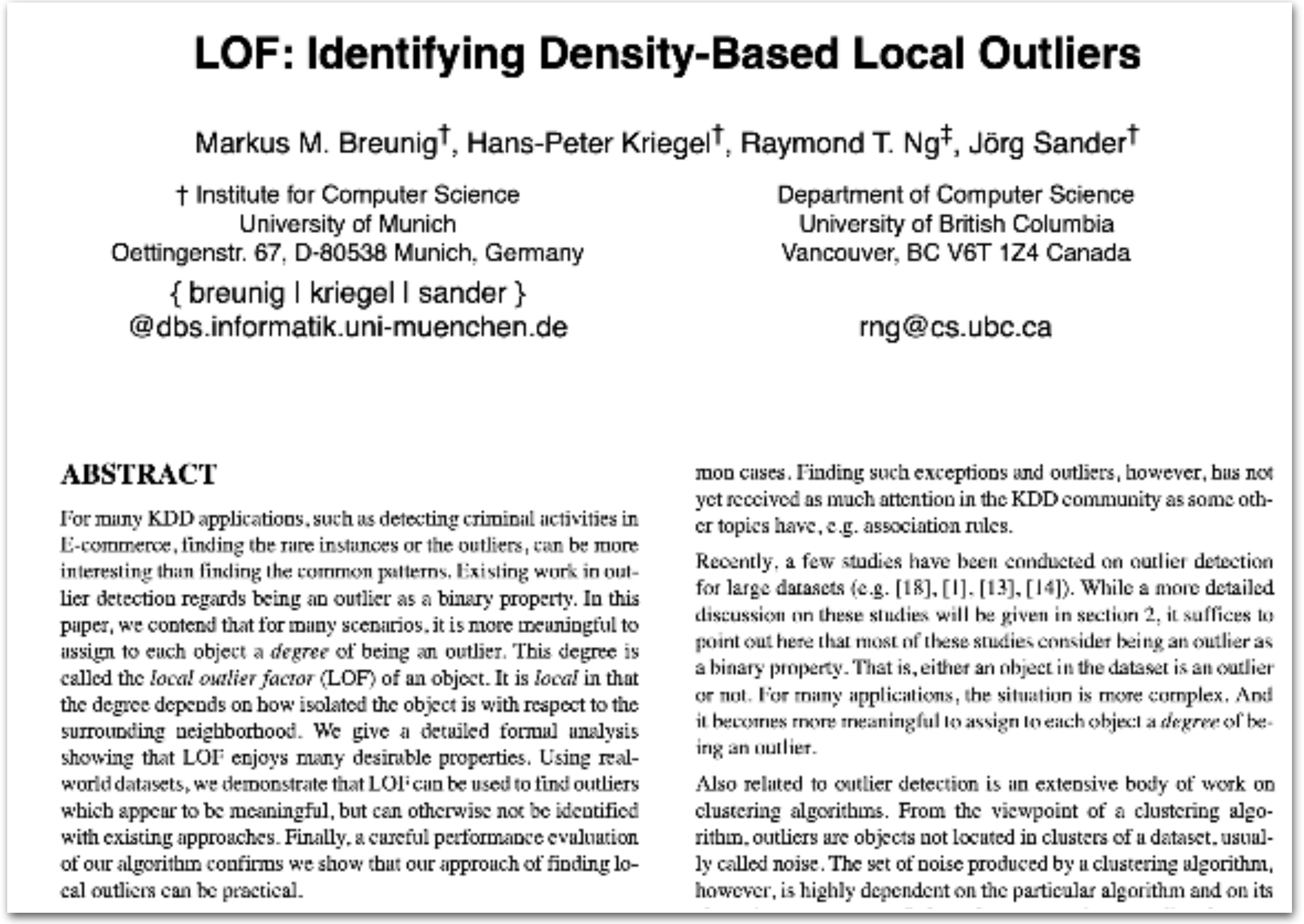
Also related to outlier detection is an extensive body of work on clustering algorithms. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of a dataset, usually called noise. The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its

[4] - Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104. 2000.
[5] - C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *The 52nd Annual Meeting of the Association for Compu- tational Linguistics: System Demonstrations*, pp. 55–60. 2014.

Approach



Formed clusters would ideally have frequent API calls, characterising the code-part of the cluster.



LOF: Identifying Density-Based Local Outliers

Markus M. Breunig[†], Hans-Peter Kriegel[†], Raymond T. Ng[‡], Jörg Sander[†]

† Institute for Computer Science
University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
{ breunig | kriegel | sander }
@dbs.informatik.uni-muenchen.de

Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada
rng@cs.ubc.ca

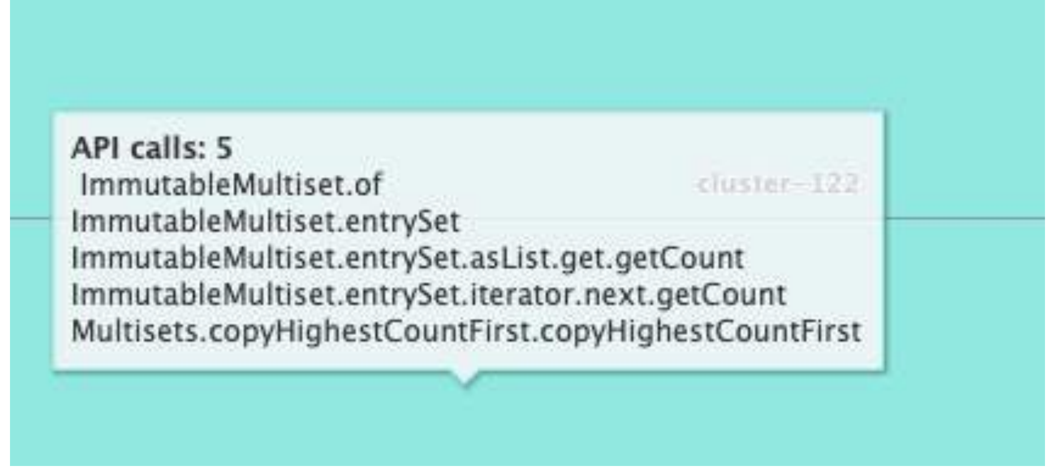
ABSTRACT

For many KDD applications, such as detecting criminal activities in E-commerce, finding the rare instances or the outliers, can be more interesting than finding the common patterns. Existing work in outlier detection regards being an outlier as a binary property. In this paper, we contend that for many scenarios, it is more meaningful to assign to each object a *degree* of being an outlier. This degree is called the *local outlier factor* (LOF) of an object. It is *local* in that the degree depends on how isolated the object is with respect to the surrounding neighborhood. We give a detailed formal analysis showing that LOF enjoys many desirable properties. Using real-world datasets, we demonstrate that LOF can be used to find outliers which appear to be meaningful, but can otherwise not be identified with existing approaches. Finally, a careful performance evaluation of our algorithm confirms we show that our approach of finding local outliers can be practical.

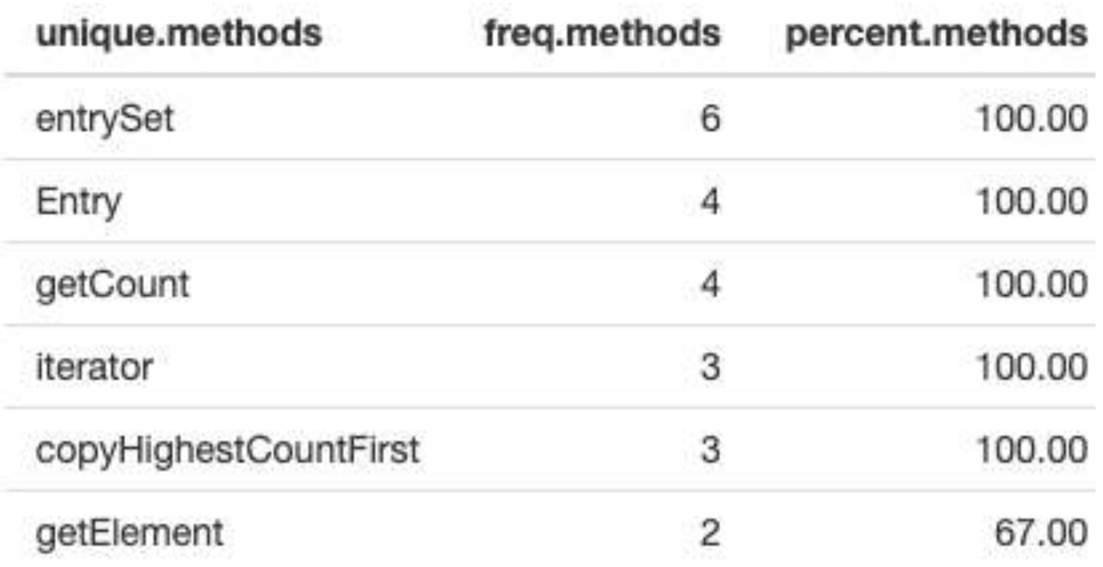
mon cases. Finding such exceptions and outliers, however, has not yet received as much attention in the KDD community as some other topics have, e.g. association rules.

Recently, a few studies have been conducted on outlier detection for large datasets (e.g. [18], [1], [13], [14]). While a more detailed discussion on these studies will be given in section 2, it suffices to point out here that most of these studies consider being an outlier as a binary property. That is, either an object in the dataset is an outlier or not. For many applications, the situation is more complex. And it becomes more meaningful to assign to each object a *degree* of being an outlier.

Also related to outlier detection is an extensive body of work on clustering algorithms. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of a dataset, usually called noise. The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its



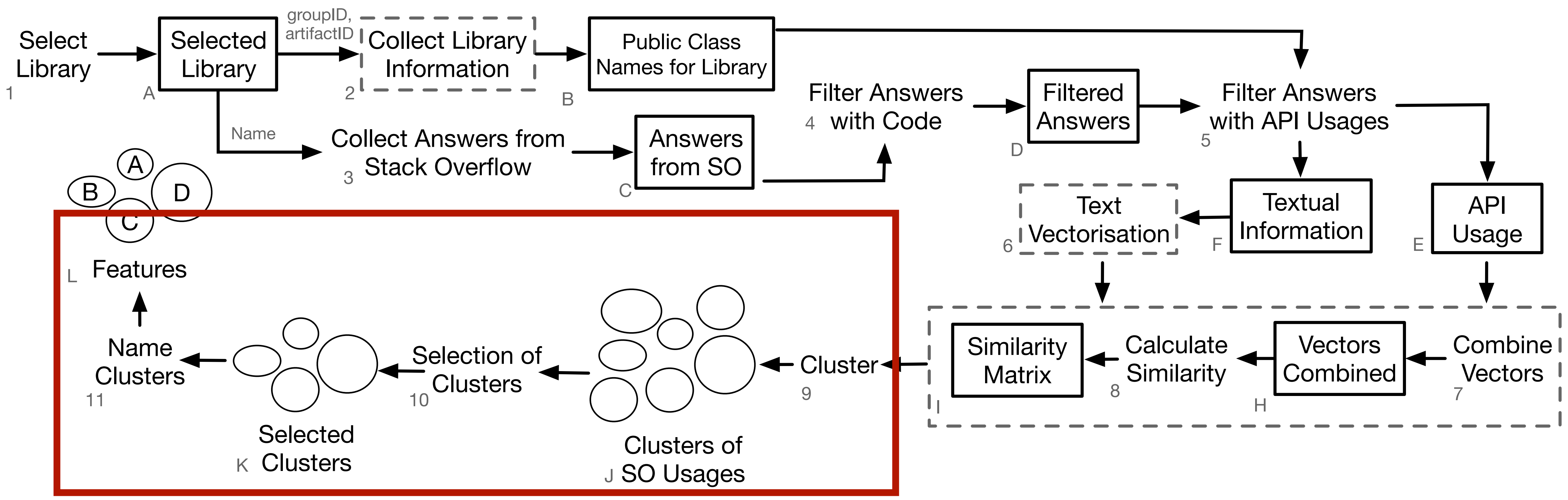
```
API calls: 5
 ImmutableMultiset.of                         cluster-122
ImmutableMultiset.entrySet
ImmutableMultiset.entrySet.asList.get.getCount
ImmutableMultiset.entrySet.iterator.next.getCount
Multisets.copyHighestCountFirst.copyHighestCountFirst
```



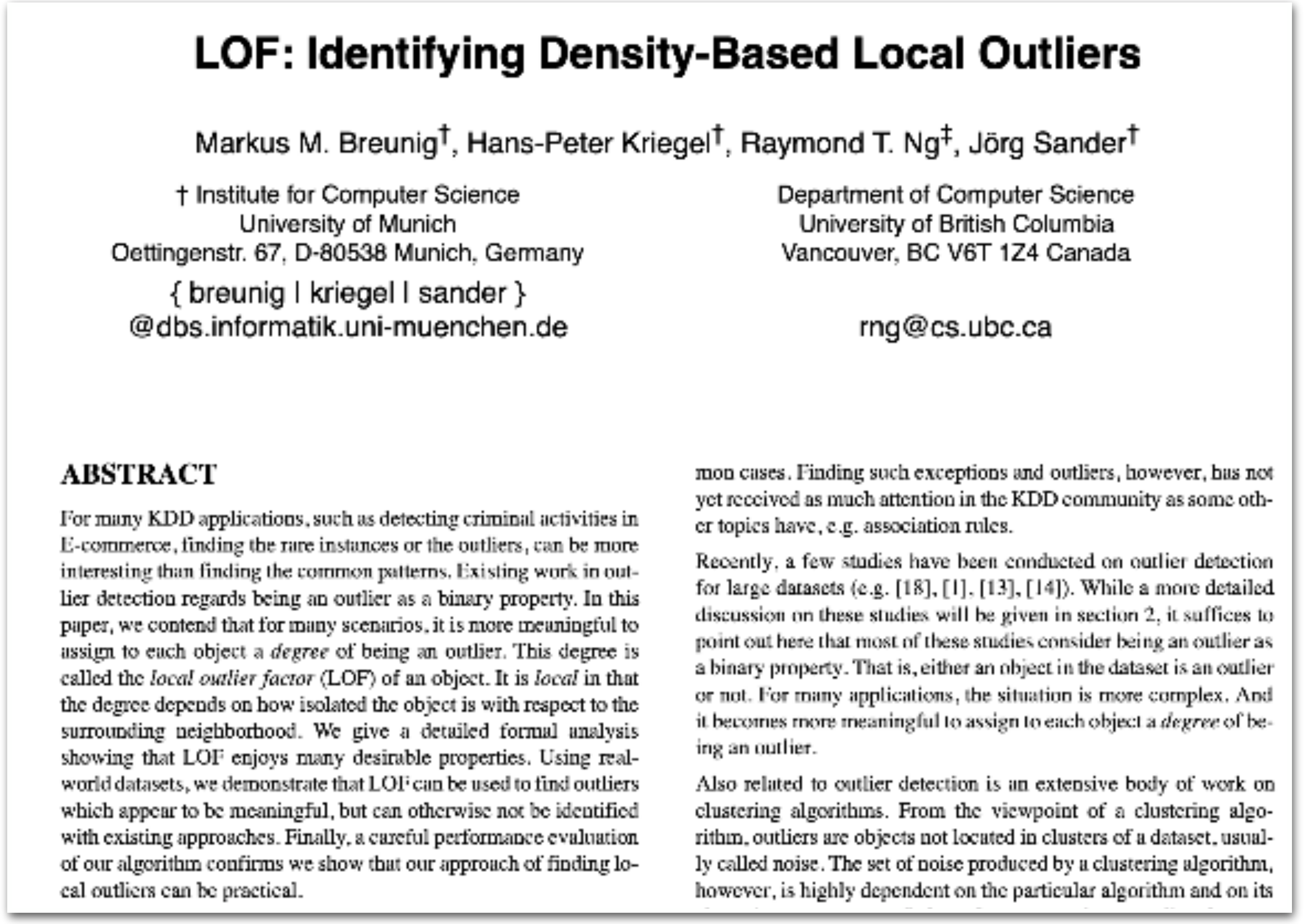
unique.methods	freq.methods	percent.methods
entrySet	6	100.00
Entry	4	100.00
getCount	4	100.00
iterator	3	100.00
copyHighestCountFirst	3	100.00
getElement	2	67.00

[4] - Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104. 2000.
[5] - C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *The 52nd Annual Meeting of the Association for Compu- tational Linguistics: System Demonstrations*, pp. 55–60. 2014.

Approach



Formed clusters would ideally have frequent API calls, characterising the code-part of the cluster.



LOF: Identifying Density-Based Local Outliers

Markus M. Breunig[†], Hans-Peter Kriegel[†], Raymond T. Ng[‡], Jörg Sander[†]

† Institute for Computer Science
University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
{ breunig | kriegel | sander }
@dbs.informatik.uni-muenchen.de

Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada
rng@cs.ubc.ca

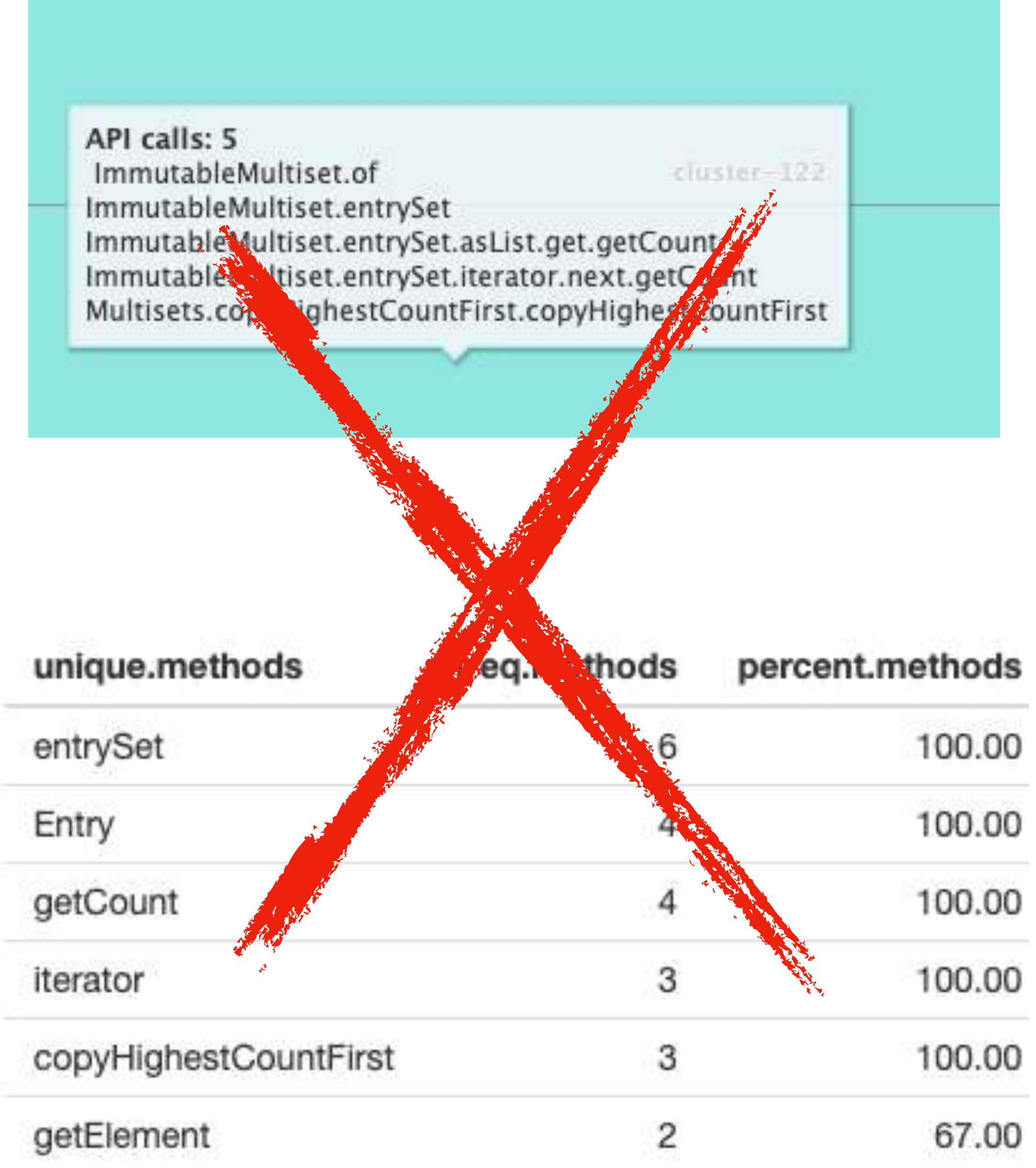
ABSTRACT

For many KDD applications, such as detecting criminal activities in E-commerce, finding the rare instances or the outliers, can be more interesting than finding the common patterns. Existing work in outlier detection regards being an outlier as a binary property. In this paper, we contend that for many scenarios, it is more meaningful to assign to each object a *degree* of being an outlier. This degree is called the *local outlier factor* (LOF) of an object. It is *local* in that the degree depends on how isolated the object is with respect to the surrounding neighborhood. We give a detailed formal analysis showing that LOF enjoys many desirable properties. Using real-world datasets, we demonstrate that LOF can be used to find outliers which appear to be meaningful, but can otherwise not be identified with existing approaches. Finally, a careful performance evaluation of our algorithm confirms we show that our approach of finding local outliers can be practical.

mon cases. Finding such exceptions and outliers, however, has not yet received as much attention in the KDD community as some other topics have, e.g. association rules.

Recently, a few studies have been conducted on outlier detection for large datasets (e.g. [18], [1], [13], [14]). While a more detailed discussion on these studies will be given in section 2, it suffices to point out here that most of these studies consider being an outlier as a binary property. That is, either an object in the dataset is an outlier or not. For many applications, the situation is more complex. And it becomes more meaningful to assign to each object a *degree* of being an outlier.

Also related to outlier detection is an extensive body of work on clustering algorithms. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of a dataset, usually called noise. The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its

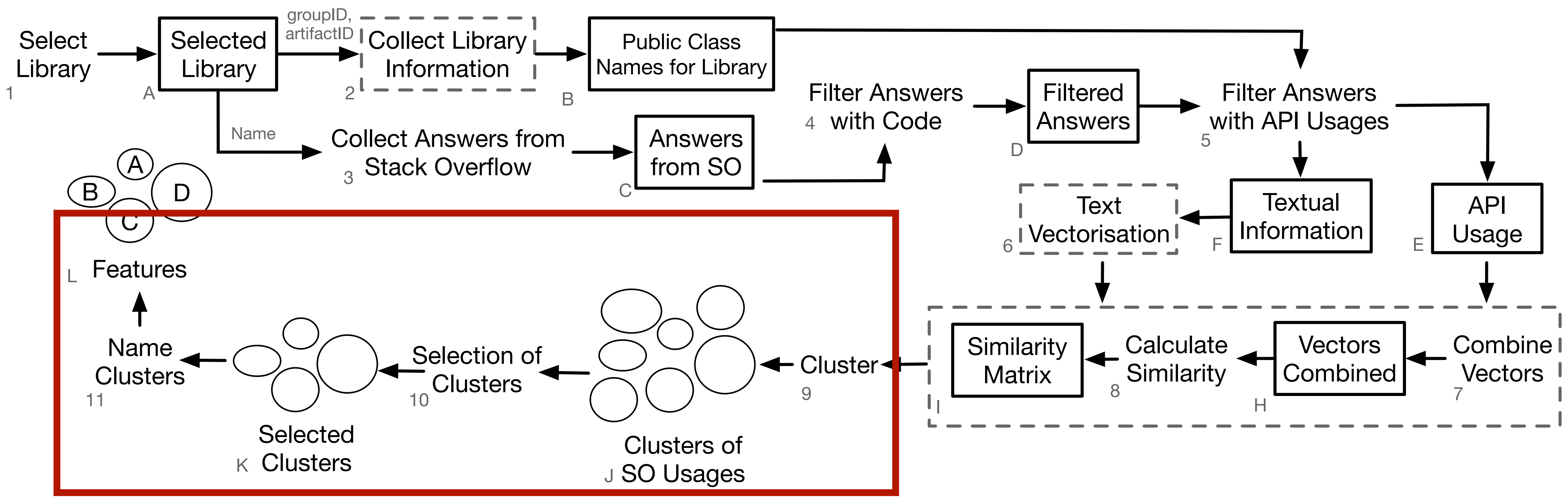


```
API calls: 5
 ImmutableMultiset.of
ImmutableMultiset.entrySet
ImmutableMultiset.entrySet.asList.get.getCount
ImmutableMultiset.entrySet.iterator.next.getCount
Multisets.copyHighestCountFirst.copyHighestCountFirst
```

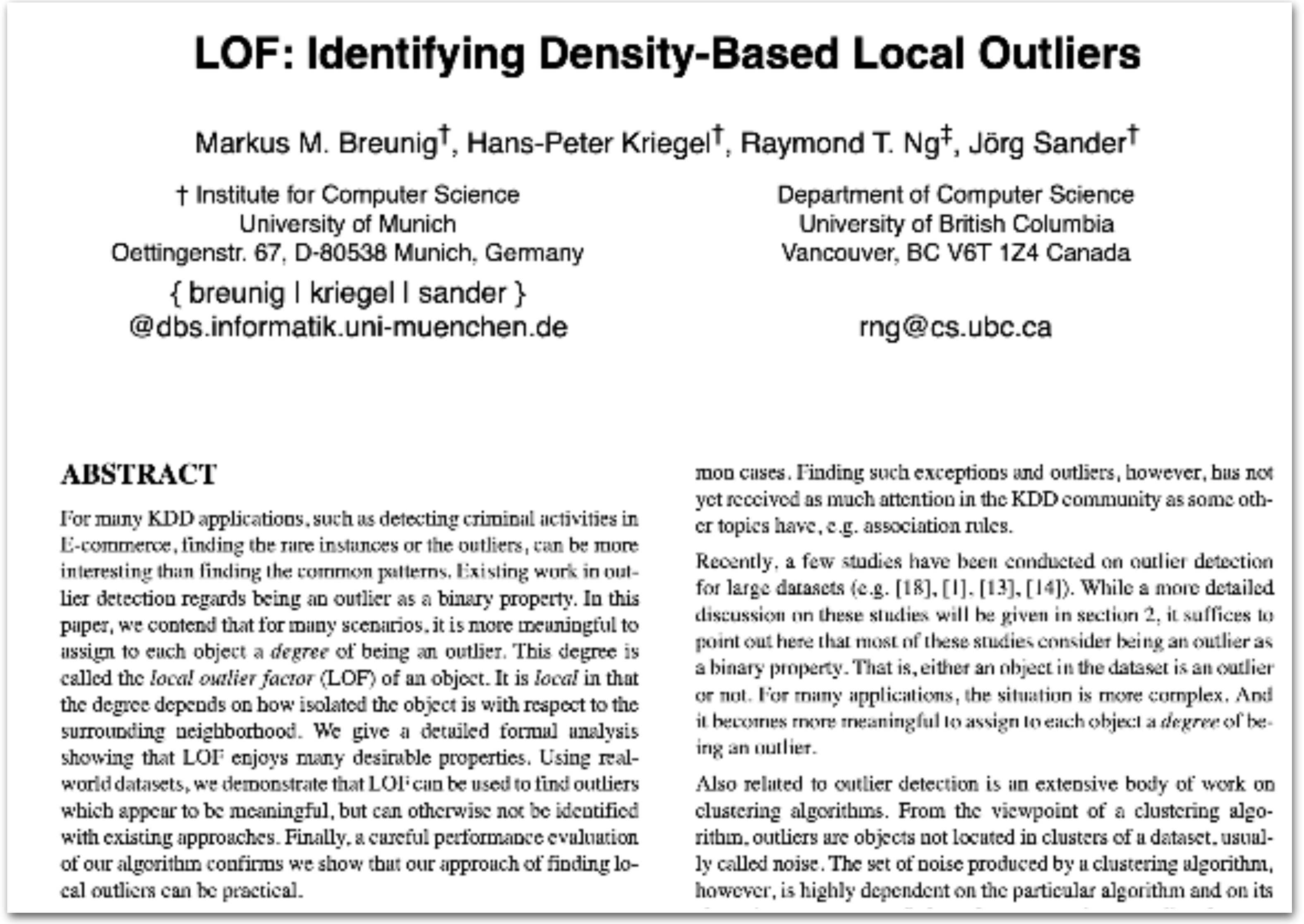
unique.methods	freq.methods	percent.methods
entrySet	6	100.00
Entry	4	100.00
getCount	4	100.00
iterator	3	100.00
copyHighestCountFirst	3	100.00
getElement	2	67.00

[4] - Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104. 2000.
[5] - C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *The 52nd Annual Meeting of the Association for Compu- tational Linguistics: System Demonstrations*, pp. 55–60. 2014.

Approach



Formed clusters would ideally have frequent API calls, characterising the code-part of the cluster.



LOF: Identifying Density-Based Local Outliers

Markus M. Breunig†, Hans-Peter Kriegel†, Raymond T. Ng‡, Jörg Sander†

† Institute for Computer Science
University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
{ breunig | kriegel | sander }
@dbs.informatik.uni-muenchen.de

‡ Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada
rng@cs.ubc.ca

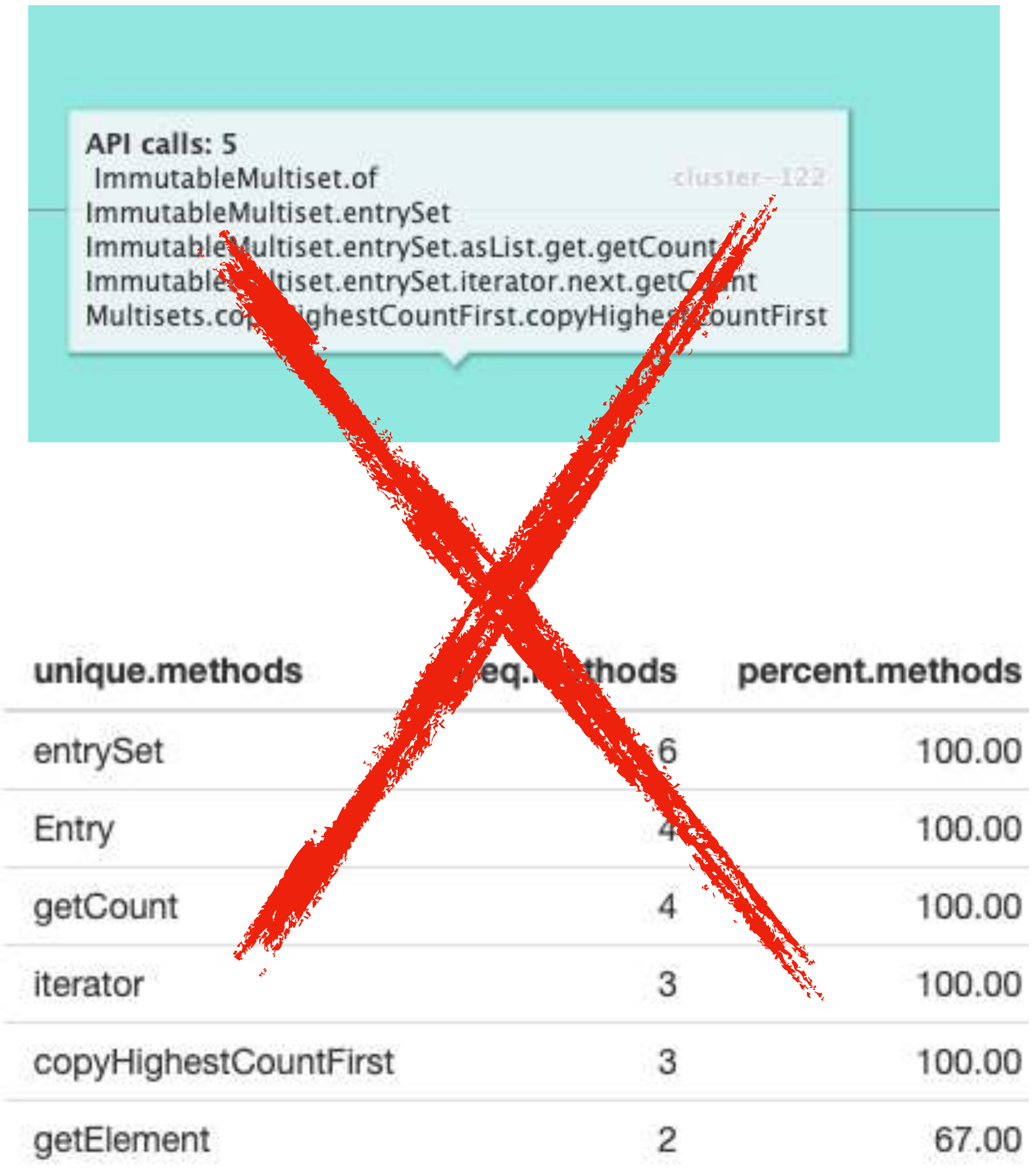
ABSTRACT

For many KDD applications, such as detecting criminal activities in E-commerce, finding the rare instances or the outliers, can be more interesting than finding the common patterns. Existing work in outlier detection regards being an outlier as a binary property. In this paper, we contend that for many scenarios, it is more meaningful to assign to each object a *degree* of being an outlier. This degree is called the *local outlier factor* (LOF) of an object. It is *local* in that the degree depends on how isolated the object is with respect to the surrounding neighborhood. We give a detailed formal analysis showing that LOF enjoys many desirable properties. Using real-world datasets, we demonstrate that LOF can be used to find outliers which appear to be meaningful, but can otherwise not be identified with existing approaches. Finally, a careful performance evaluation of our algorithm confirms we show that our approach of finding local outliers can be practical.

mon cases. Finding such exceptions and outliers, however, has not yet received as much attention in the KDD community as some other topics have, e.g. association rules.

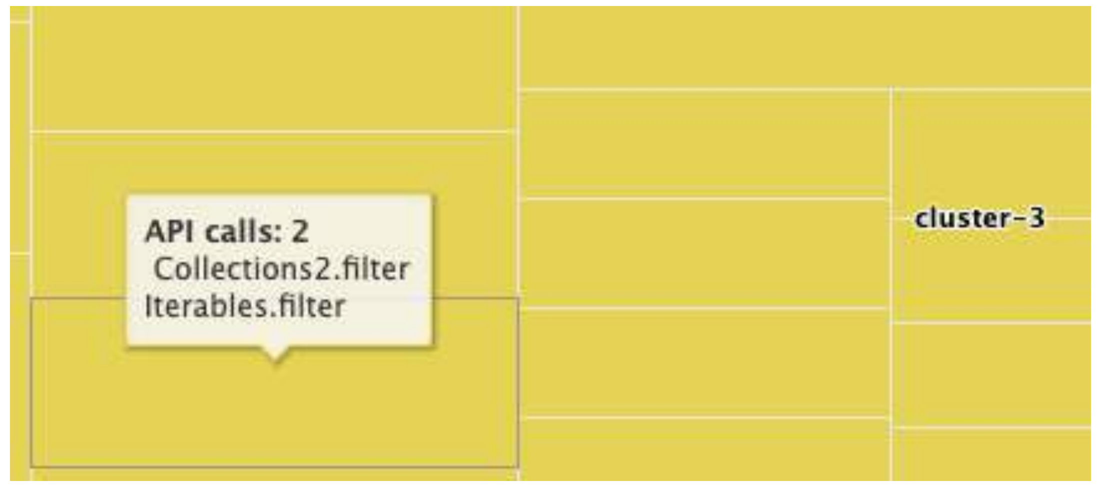
Recently, a few studies have been conducted on outlier detection for large datasets (e.g. [18], [1], [13], [14]). While a more detailed discussion on these studies will be given in section 2, it suffices to point out here that most of these studies consider being an outlier as a binary property. That is, either an object in the dataset is an outlier or not. For many applications, the situation is more complex. And it becomes more meaningful to assign to each object a *degree* of being an outlier.

Also related to outlier detection is an extensive body of work on clustering algorithms. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of a dataset, usually called noise. The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its

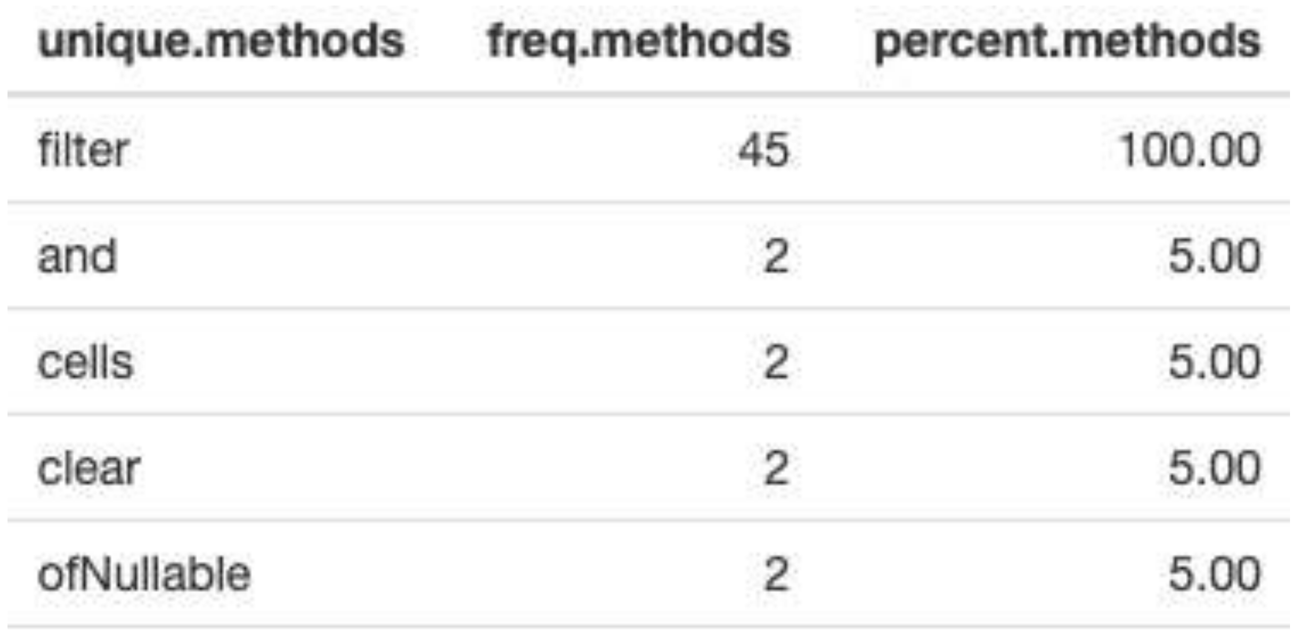


API calls: 5
ImmutableMultiset.of
ImmutableMultiset.entrySet
ImmutableMultiset.entrySet.asList.get.getCount
ImmutableMultiset.entrySet.iterator.next.getCount
Multisets.copyHighestCountFirst.copyHighestCountFirst

unique.methods	freq.methods	percent.methods
entrySet	6	100.00
Entry	4	100.00
getCount	4	100.00
iterator	3	100.00
copyHighestCountFirst	3	100.00
getElement	2	67.00



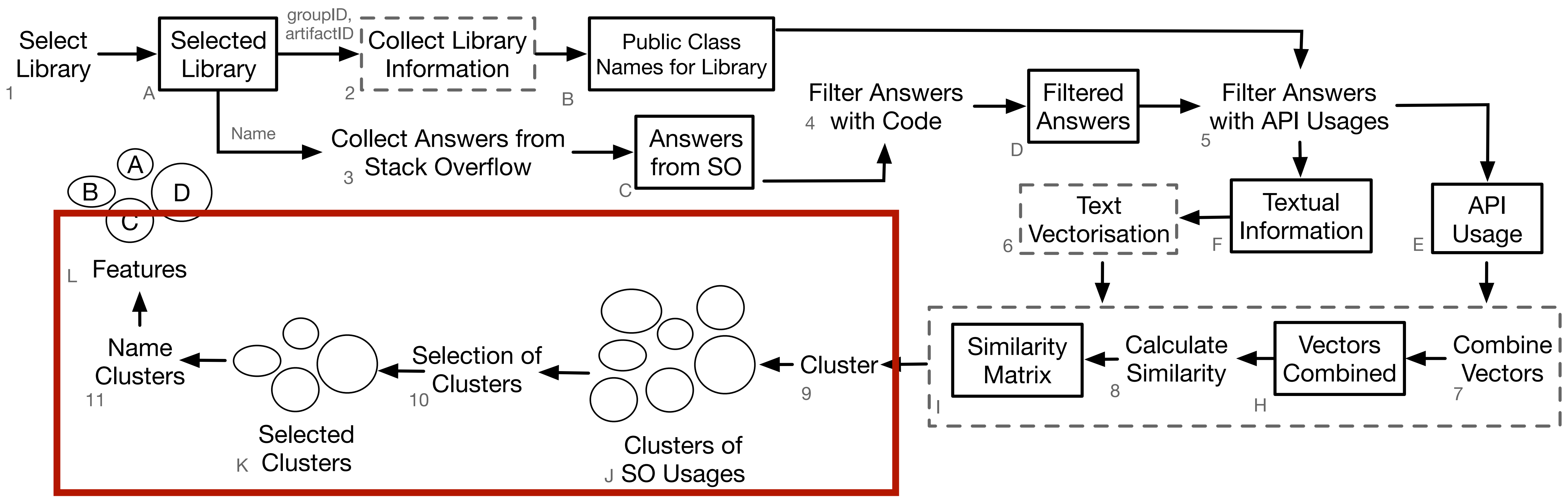
API calls: 2
Collections2.filter
Iterables.filter



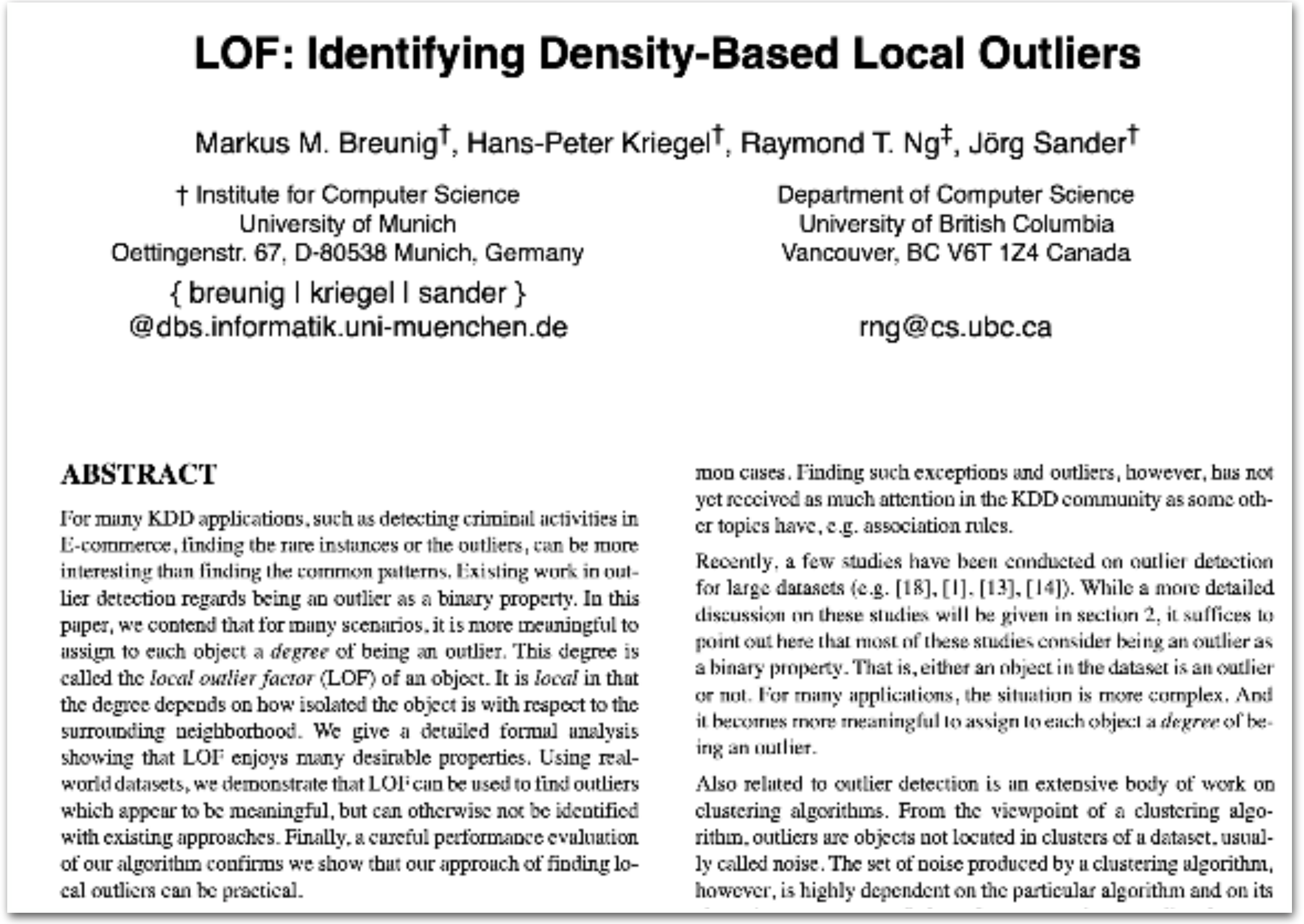
unique.methods	freq.methods	percent.methods
filter	45	100.00
and	2	5.00
cells	2	5.00
clear	2	5.00
ofNullable	2	5.00

[4] - Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104. 2000.
[5] - C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *The 52nd Annual Meeting of the Association for Compu- tational Linguistics: System Demonstrations*, pp. 55–60. 2014.

Approach



Formed clusters would ideally have frequent API calls, characterising the code-part of the cluster.



LOF: Identifying Density-Based Local Outliers

Markus M. Breunig[†], Hans-Peter Kriegel[†], Raymond T. Ng[‡], Jörg Sander[†]

† Institute for Computer Science
University of Munich
Oettingenstr. 67, D-80538 Munich, Germany
{ breunig | kriegel | sander }
@dbs.informatik.uni-muenchen.de

‡ Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4 Canada
rng@cs.ubc.ca

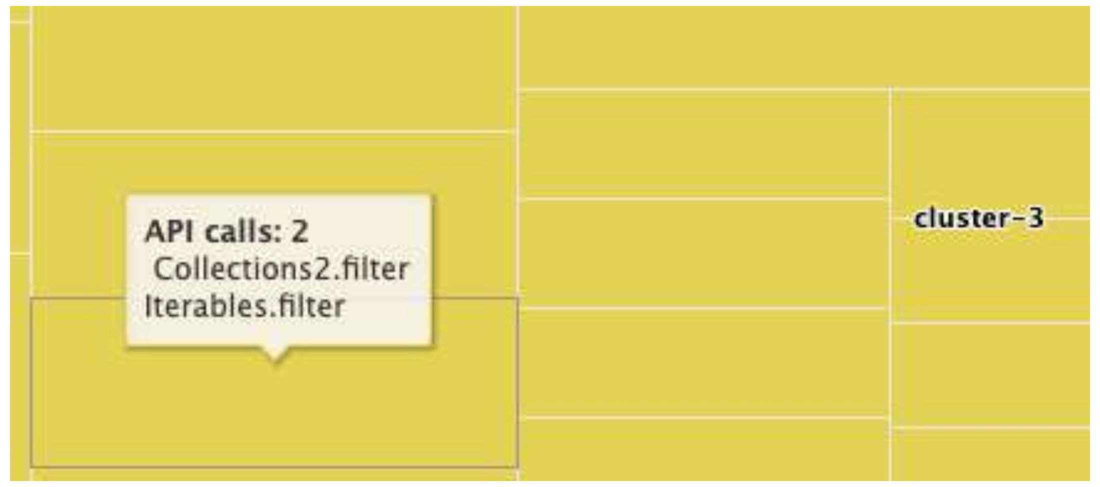
ABSTRACT

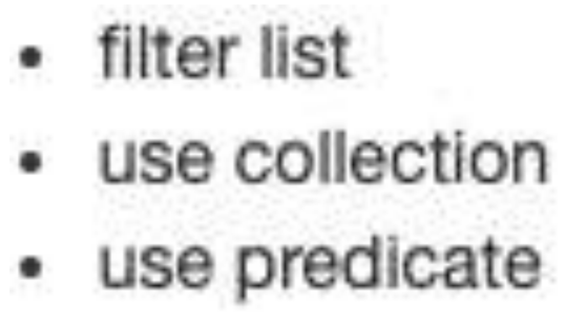
For many KDD applications, such as detecting criminal activities in E-commerce, finding the rare instances or the outliers, can be more interesting than finding the common patterns. Existing work in outlier detection regards being an outlier as a binary property. In this paper, we contend that for many scenarios, it is more meaningful to assign to each object a *degree* of being an outlier. This degree is called the *local outlier factor* (LOF) of an object. It is *local* in that the degree depends on how isolated the object is with respect to the surrounding neighborhood. We give a detailed formal analysis showing that LOF enjoys many desirable properties. Using real-world datasets, we demonstrate that LOF can be used to find outliers which appear to be meaningful, but can otherwise not be identified with existing approaches. Finally, a careful performance evaluation of our algorithm confirms we show that our approach of finding local outliers can be practical.

mon cases. Finding such exceptions and outliers, however, has not yet received as much attention in the KDD community as some other topics have, e.g. association rules.

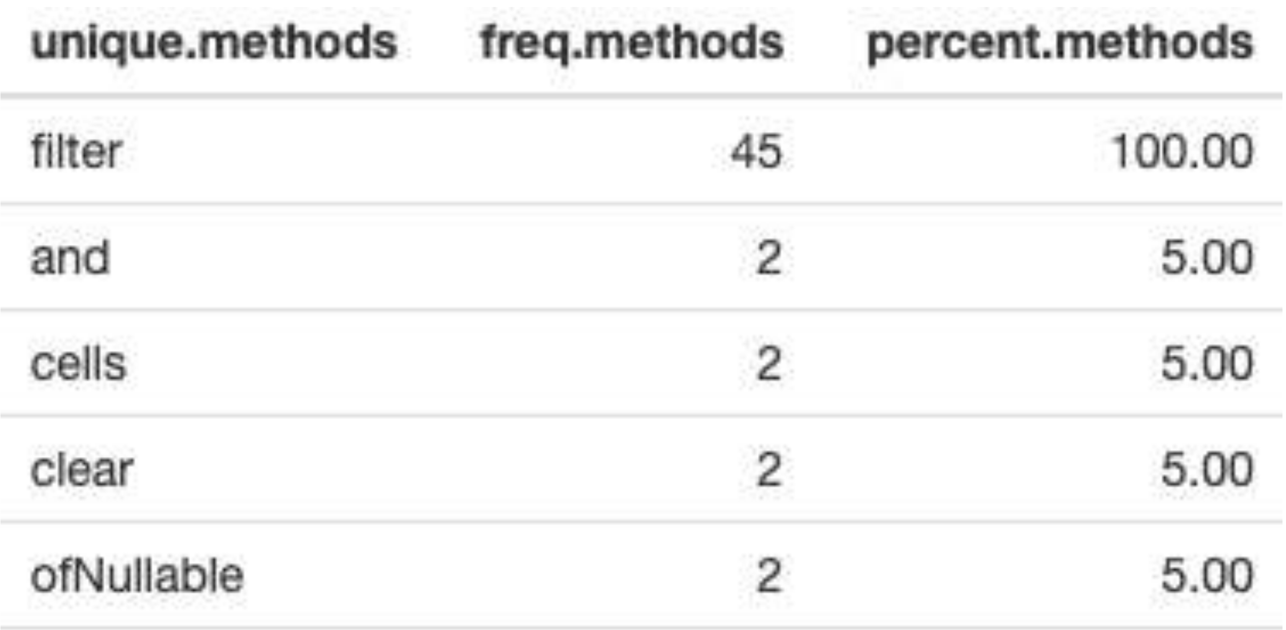
Recently, a few studies have been conducted on outlier detection for large datasets (e.g. [18], [1], [13], [14]). While a more detailed discussion on these studies will be given in section 2, it suffices to point out here that most of these studies consider being an outlier as a binary property. That is, either an object in the dataset is an outlier or not. For many applications, the situation is more complex. And it becomes more meaningful to assign to each object a *degree* of being an outlier.

Also related to outlier detection is an extensive body of work on clustering algorithms. From the viewpoint of a clustering algorithm, outliers are objects not located in clusters of a dataset, usually called noise. The set of noise produced by a clustering algorithm, however, is highly dependent on the particular algorithm and on its





- filter list
- use collection
- use predicate

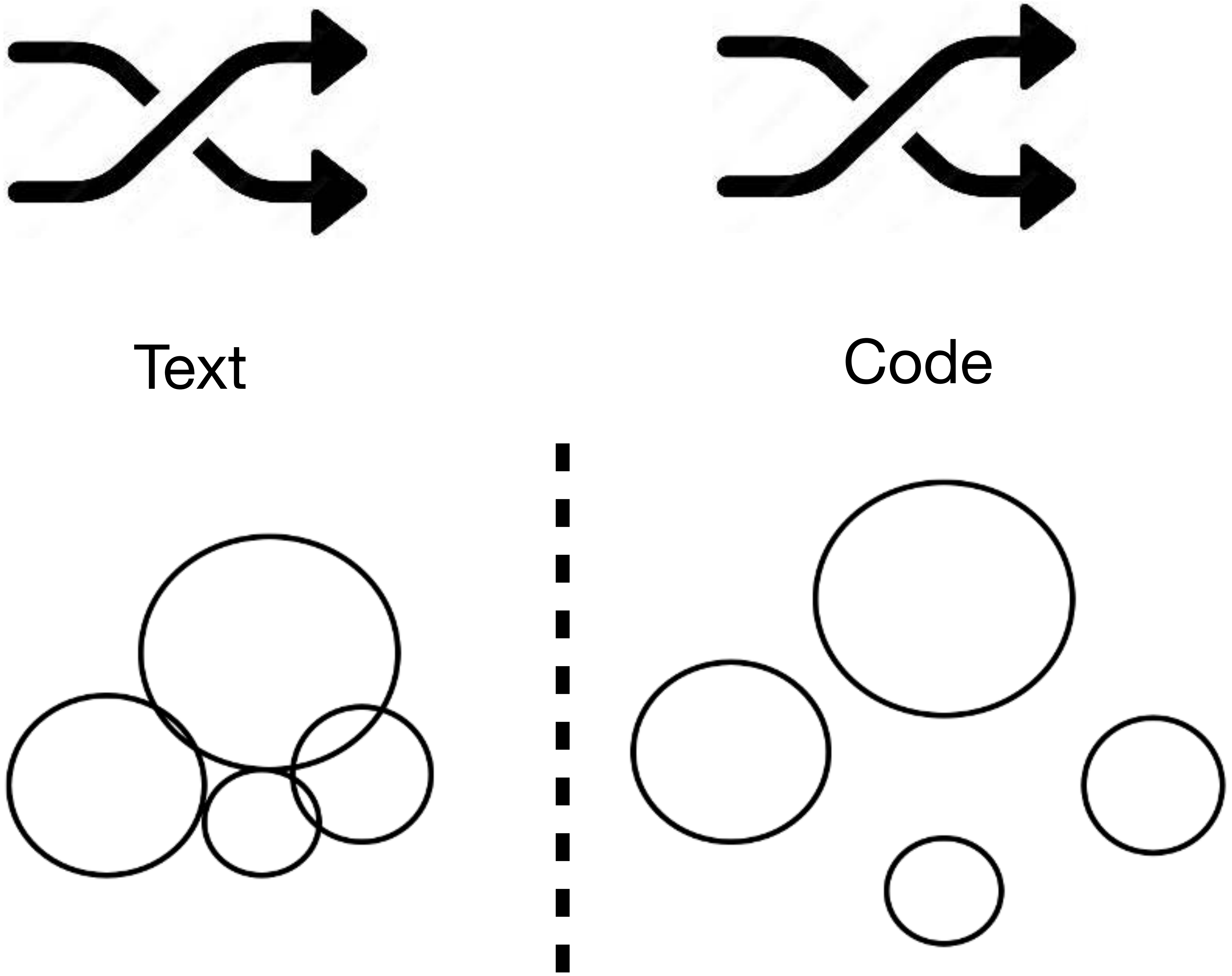


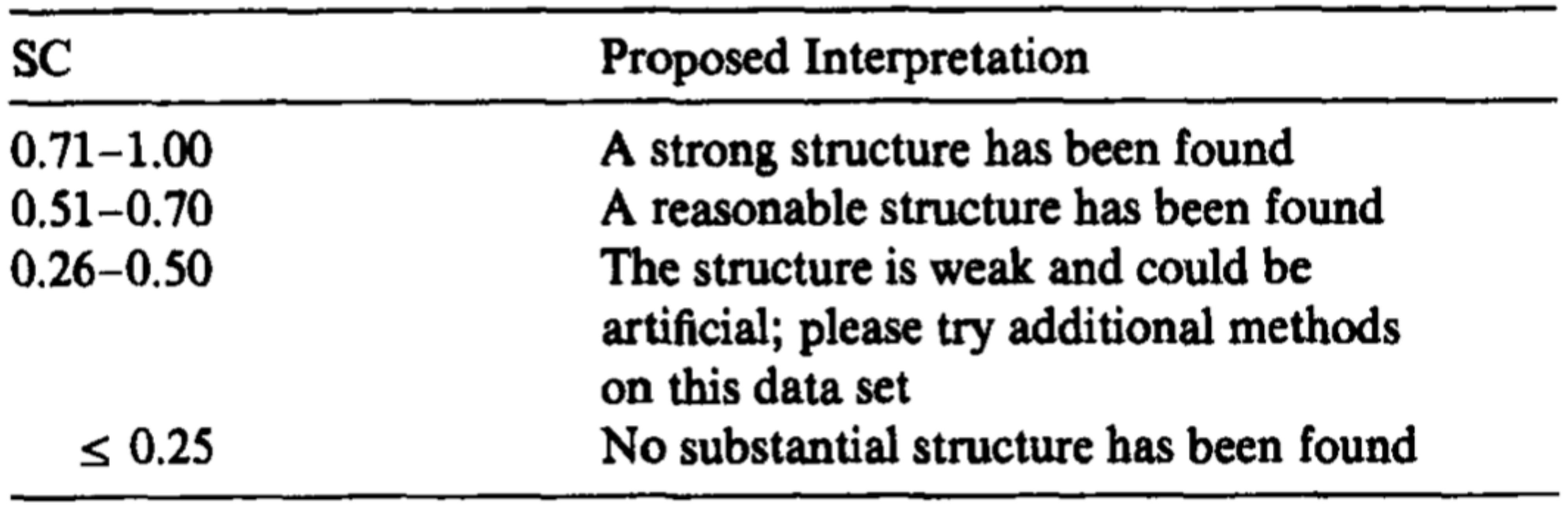
unique.methods	freq.methods	percent.methods
filter	45	100.00
and	2	5.00
cells	2	5.00
clear	2	5.00
ofNullable	2	5.00

[4] - Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: identifying density-based local outliers." In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93-104. 2000.
[5] - C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *The 52nd Annual Meeting of the Association for Compu- tational Linguistics: System Demonstrations*, pp. 55–60. 2014.

Evaluation

RQ1. Which combination of SO answer attributes produces the most cohesive clusters?





SC	Proposed Interpretation
0.71–1.00	A strong structure has been found
0.51–0.70	A reasonable structure has been found
0.26–0.50	The structure is weak and could be artificial; please try additional methods on this data set
≤ 0.25	No substantial structure has been found

[6] - L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990, vol. 344.

Evaluation

RQ1. Which combination of SO answer attributes produces the most cohesive clusters?

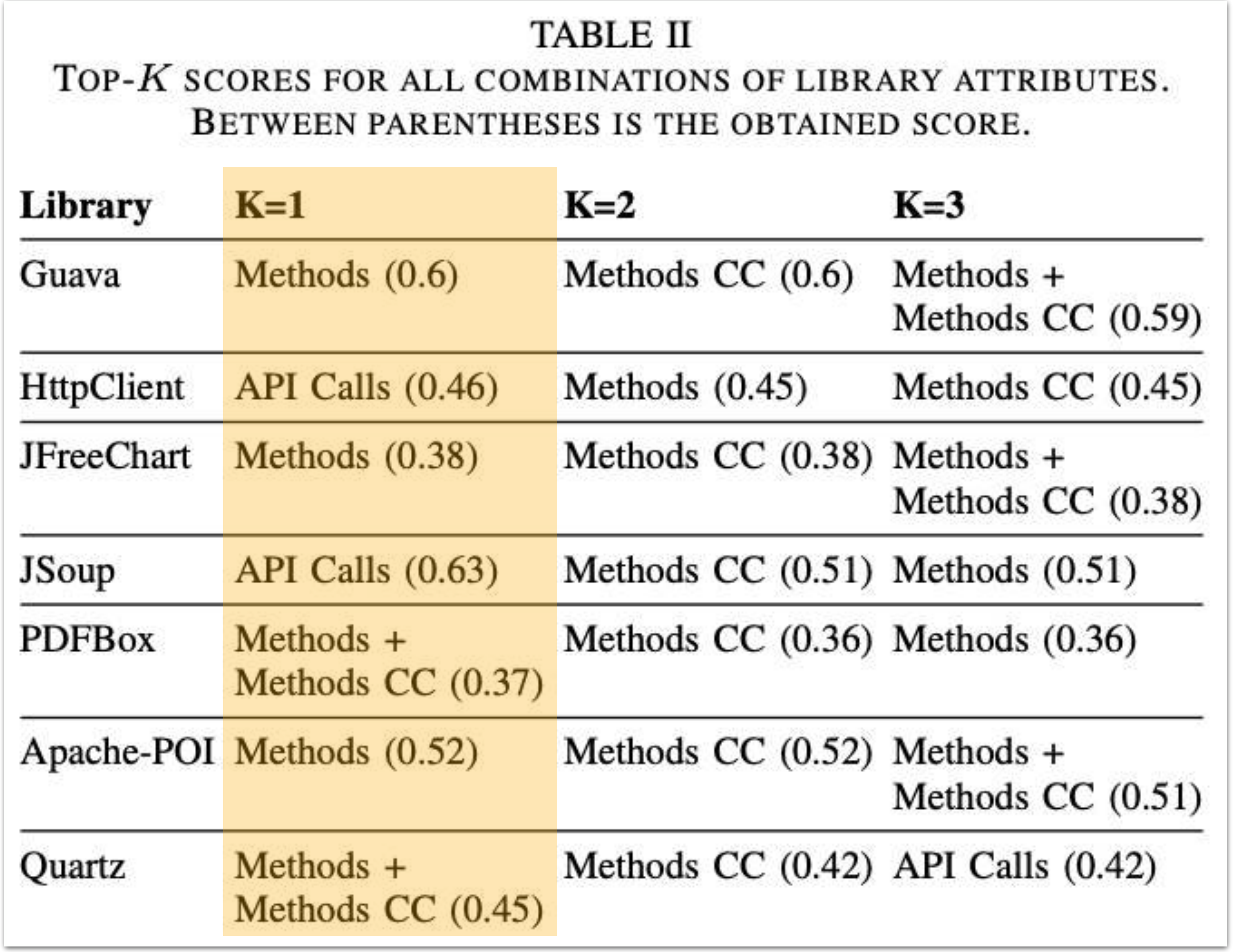


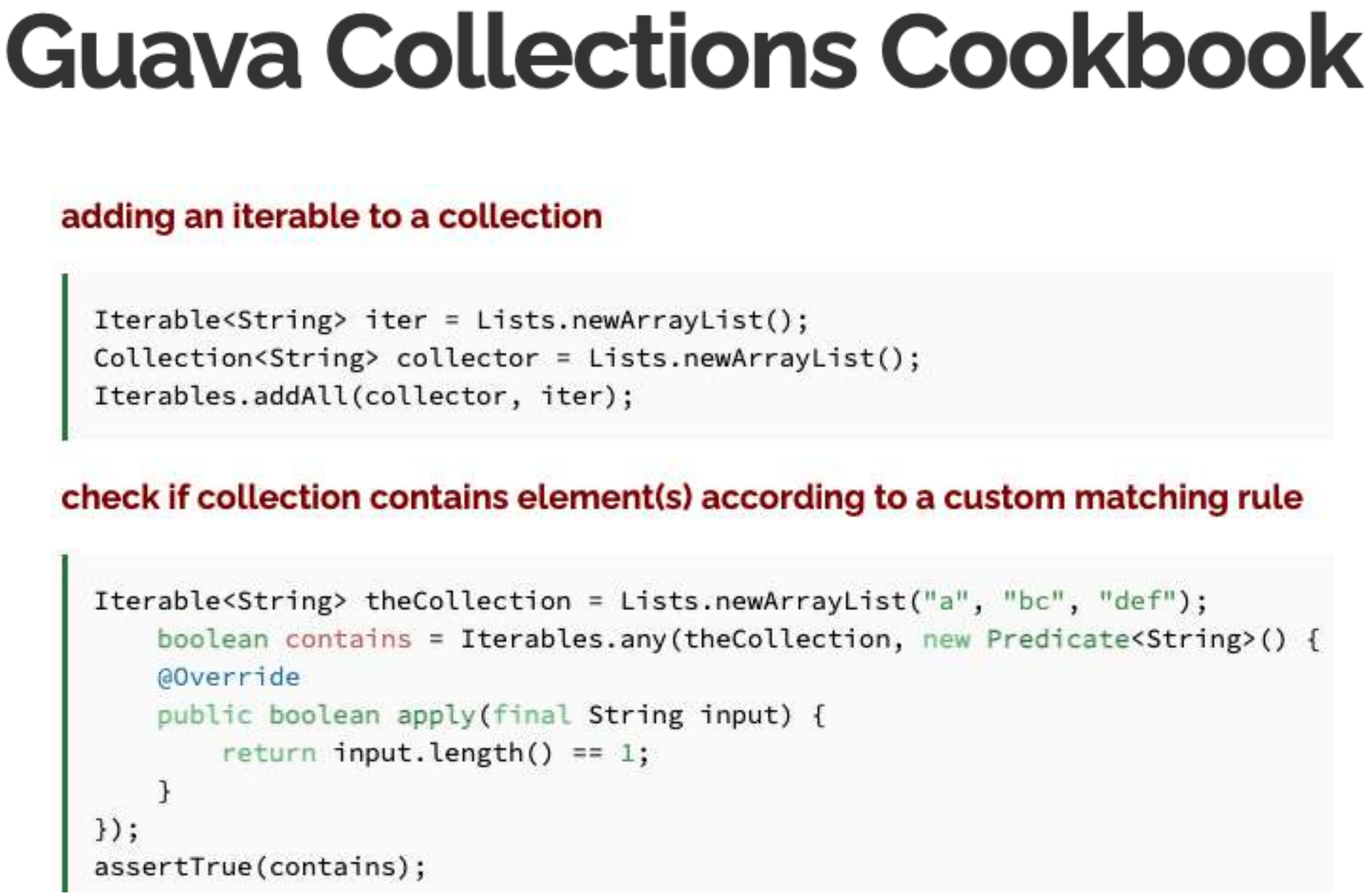
TABLE II
TOP-*K* SCORES FOR ALL COMBINATIONS OF LIBRARY ATTRIBUTES. BETWEEN PARENTHESES IS THE OBTAINED SCORE.

Library	K=1	K=2	K=3
Guava	Methods (0.6)	Methods CC (0.6)	Methods + Methods CC (0.59)
HttpClient	API Calls (0.46)	Methods (0.45)	Methods CC (0.45)
JFreeChart	Methods (0.38)	Methods CC (0.38)	Methods + Methods CC (0.38)
JSoup	API Calls (0.63)	Methods CC (0.51)	Methods (0.51)
PDFBox	Methods + Methods CC (0.37)	Methods CC (0.36)	Methods (0.36)
Apache-POI	Methods (0.52)	Methods CC (0.52)	Methods + Methods CC (0.51)
Quartz	Methods + Methods CC (0.45)	Methods CC (0.42)	API Calls (0.42)

The **Methods** combination from the API usage information attributes produced the most cohesive clusters.

Evaluation

RQ2. How similar are the automatically uncovered features to documented tutorial features?



Guava Collections Cookbook

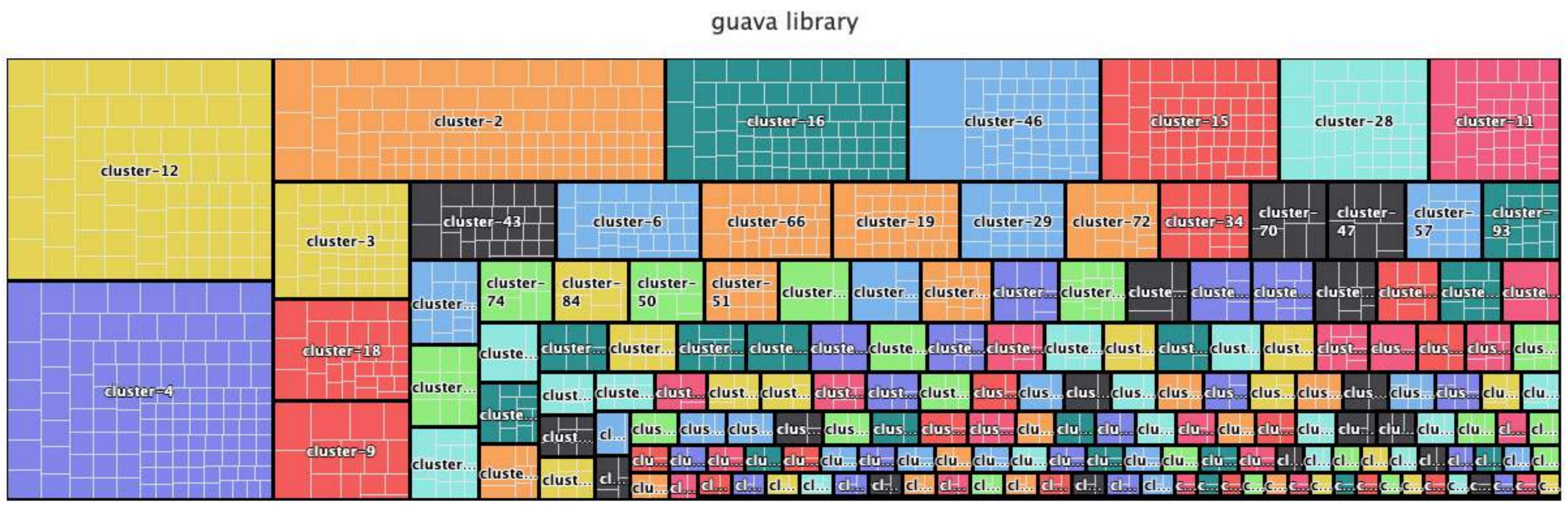
adding an iterable to a collection

```
Iterable<String> iter = Lists.newArrayList();
Collection<String> collector = Lists.newArrayList();
Iterables.addAll(collector, iter);
```

check if collection contains element(s) according to a custom matching rule

```
Iterable<String> theCollection = Lists.newArrayList("a", "bc", "def");
    boolean contains = Iterables.any(theCollection, new Predicate<String>() {
    @Override
    public boolean apply(final String input) {
        return input.length() == 1;
    }
});
assertTrue(contains);
```

Generated features



Evaluation

RQ2. How similar are the automatically uncovered features to documented tutorial features?

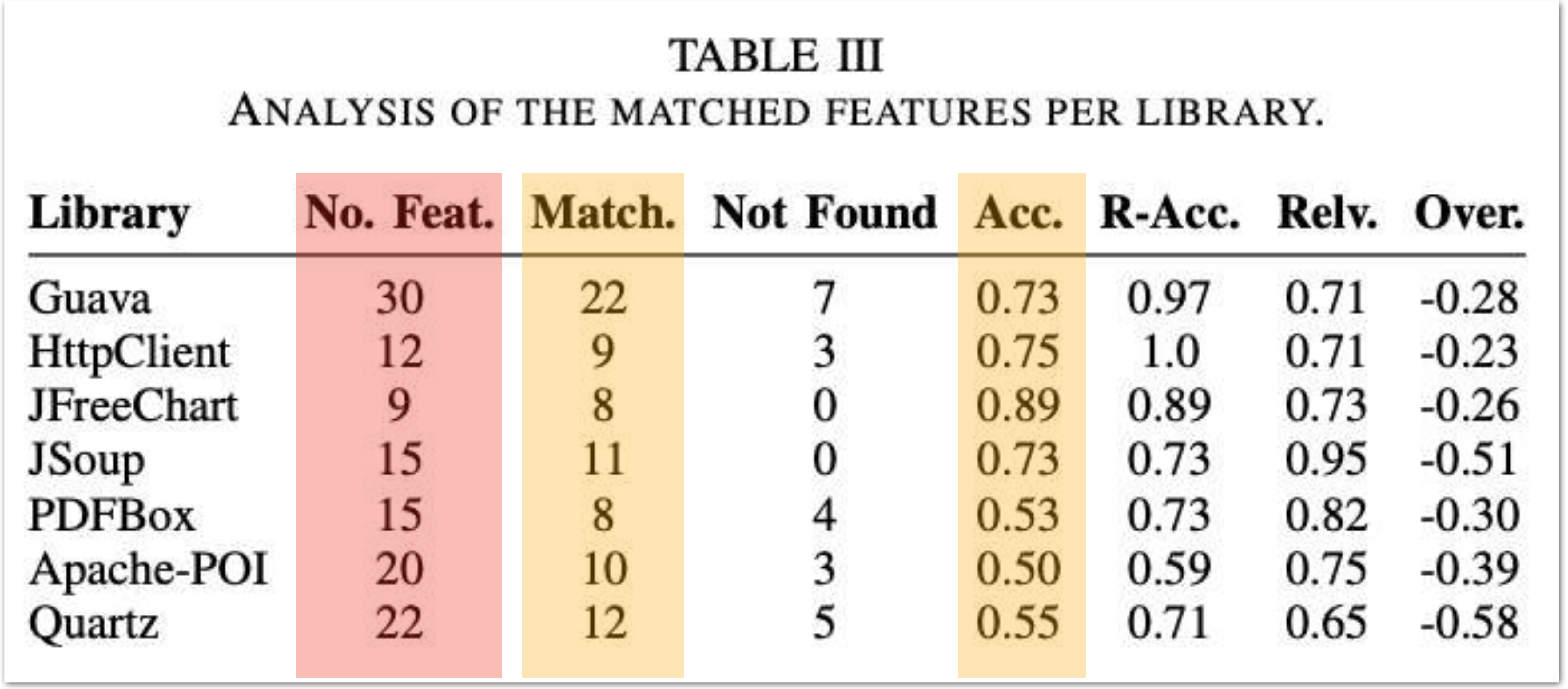


TABLE III
ANALYSIS OF THE MATCHED FEATURES PER LIBRARY.

Library	No. Feat.	Match.	Not Found	Acc.	R-Acc.	Relv.	Over.
Guava	30	22	7	0.73	0.97	0.71	-0.28
HttpClient	12	9	3	0.75	1.0	0.71	-0.23
JFreeChart	9	8	0	0.89	0.89	0.73	-0.26
JSoup	15	11	0	0.73	0.73	0.95	-0.51
PDFBox	15	8	4	0.53	0.73	0.82	-0.30
Apache-POI	20	10	3	0.50	0.59	0.75	-0.39
Quartz	22	12	5	0.55	0.71	0.65	-0.58

The majority of the tutorial features is part of our uncovered features.

Evaluation

RQ2. How similar are the automatically uncovered features to documented tutorial features?

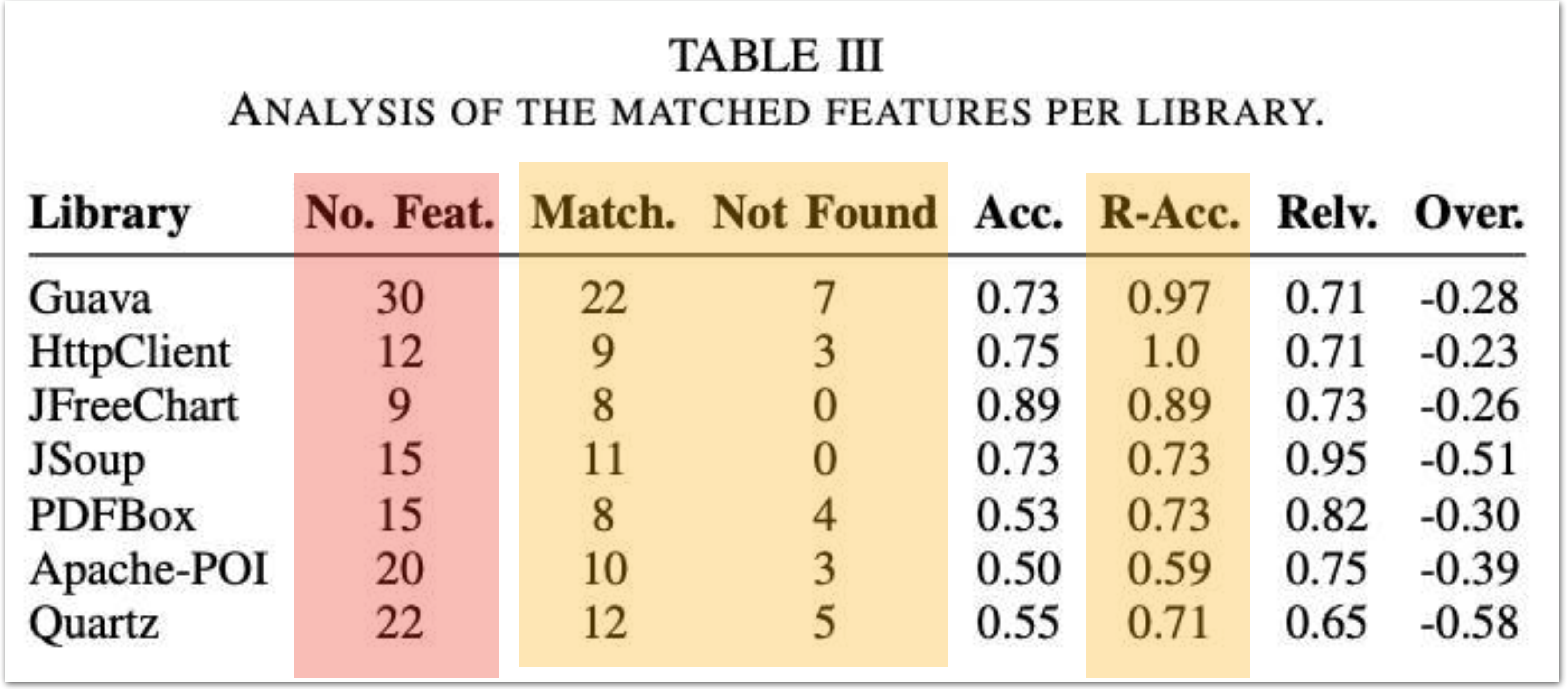


TABLE III
ANALYSIS OF THE MATCHED FEATURES PER LIBRARY.

Library	No. Feat.	Match.	Not Found	Acc.	R-Acc.	Relv.	Over.
Guava	30	22	7	0.73	0.97	0.71	-0.28
HttpClient	12	9	3	0.75	1.0	0.71	-0.23
JFreeChart	9	8	0	0.89	0.89	0.73	-0.26
JSoup	15	11	0	0.73	0.73	0.95	-0.51
PDFBox	15	8	4	0.53	0.73	0.82	-0.30
Apache-POI	20	10	3	0.50	0.59	0.75	-0.39
Quartz	22	12	5	0.55	0.71	0.65	-0.58

When considered against the usages in the SO data, the matched accuracy increases.

Evaluation

RQ2. How similar are the automatically uncovered features to documented tutorial features?

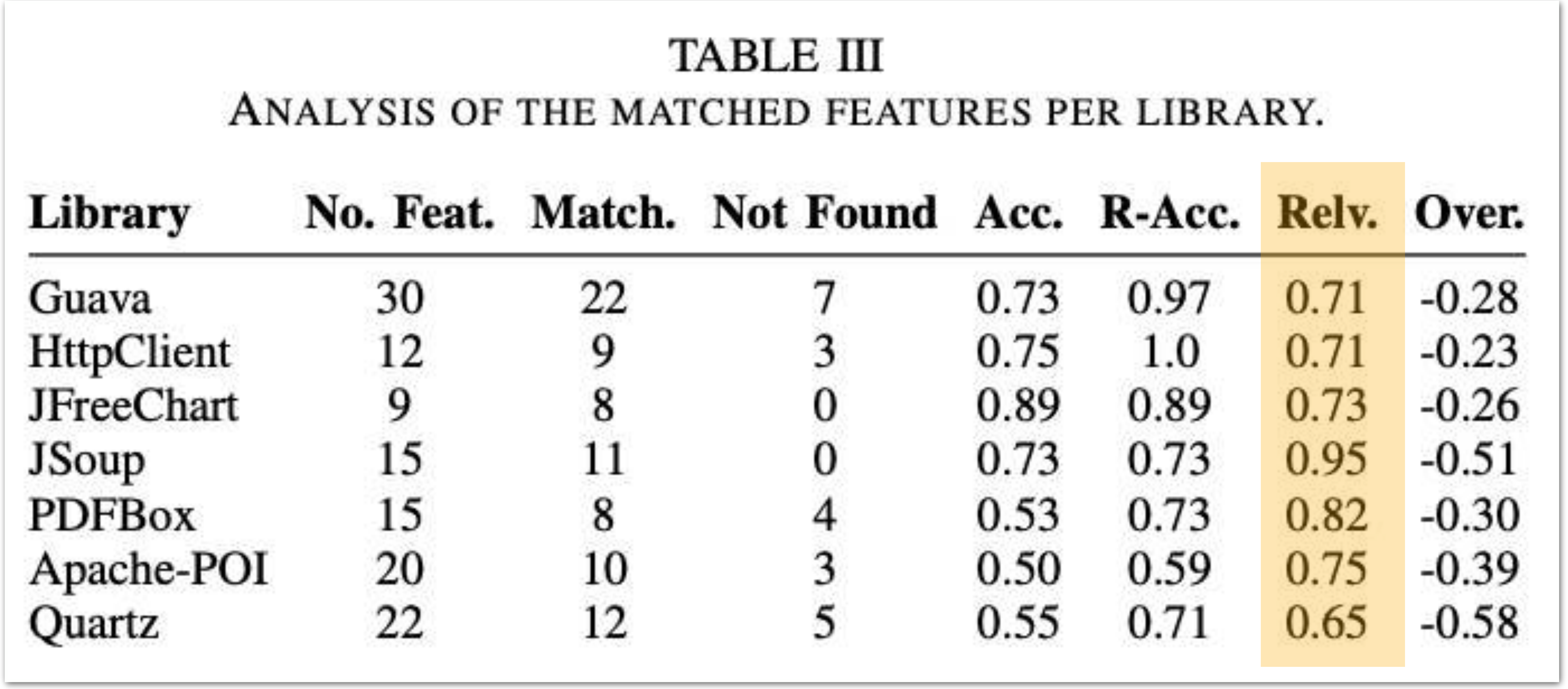


TABLE III

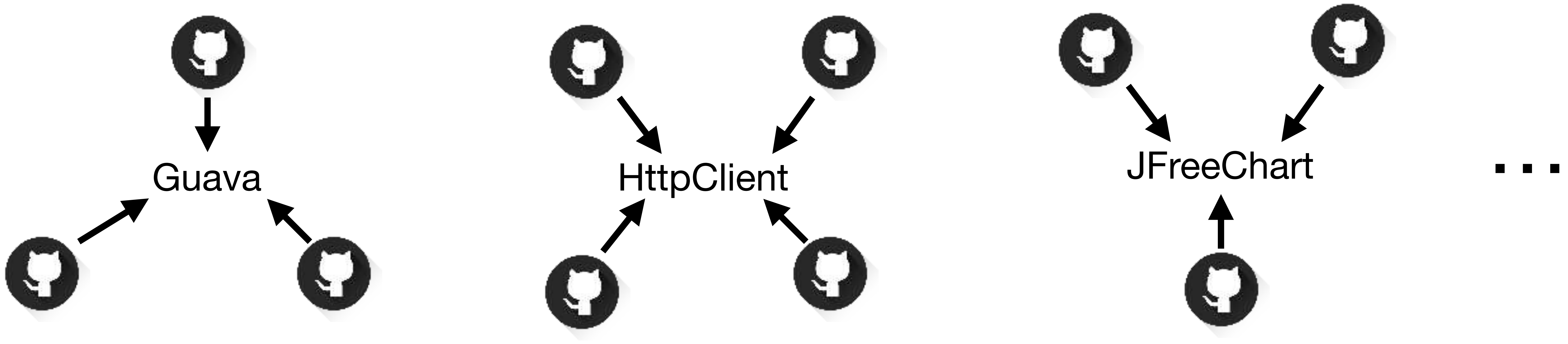
ANALYSIS OF THE MATCHED FEATURES PER LIBRARY.

Library	No. Feat.	Match.	Not Found	Acc.	R-Acc.	Relv.	Over.
Guava	30	22	7	0.73	0.97	0.71	-0.28
HttpClient	12	9	3	0.75	1.0	0.71	-0.23
JFreeChart	9	8	0	0.89	0.89	0.73	-0.26
JSoup	15	11	0	0.73	0.73	0.95	-0.51
PDFBox	15	8	4	0.53	0.73	0.82	-0.30
Apache-POI	20	10	3	0.50	0.59	0.75	-0.39
Quartz	22	12	5	0.55	0.71	0.65	-0.58

On average, our uncovered features are very similar to those in tutorials denoted by the relevance score.

Evaluation

RQ3. To which extent do the uncovered features that do not match documented tutorial features correspond to actual API usage in client projects?



Evaluation

RQ3. To which extent do the uncovered features that do not match documented tutorial features correspond to actual API usage in client projects?

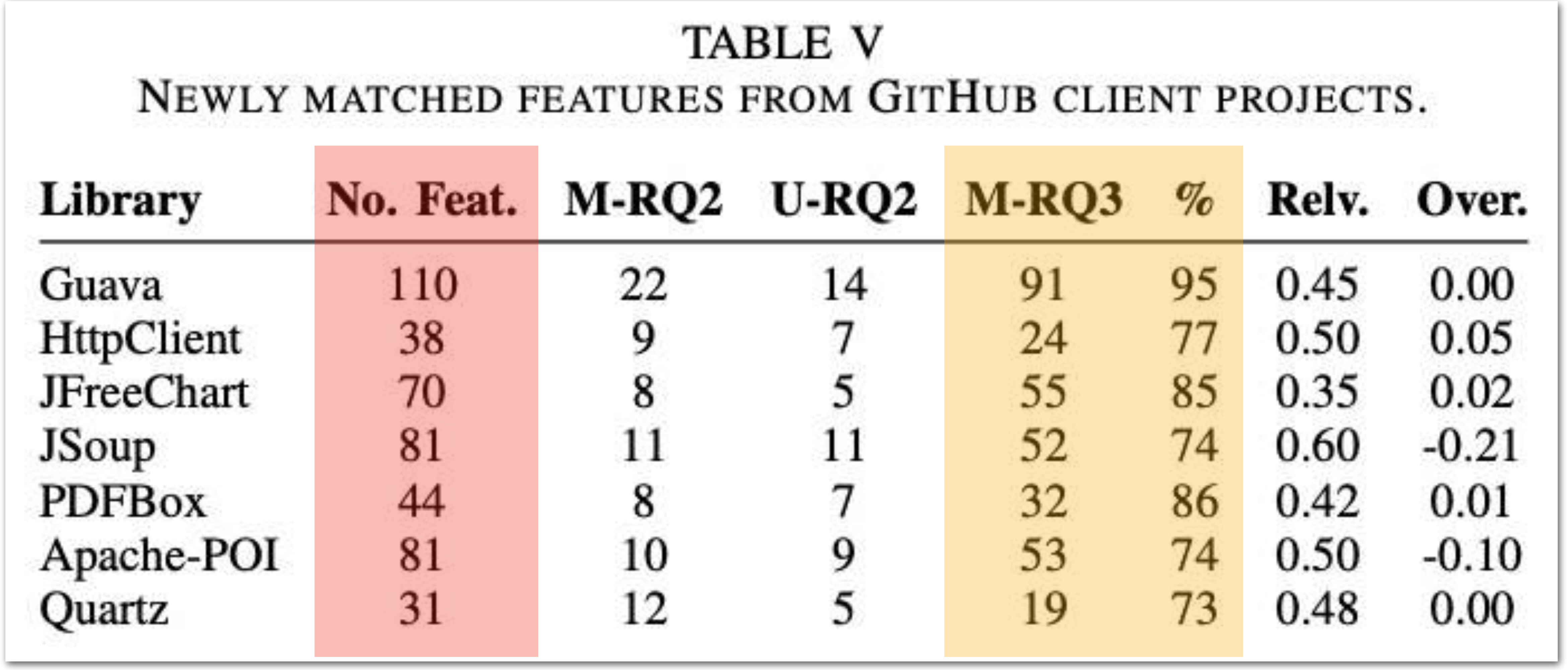


TABLE V
NEWLY MATCHED FEATURES FROM GITHUB CLIENT PROJECTS.

Library	No. Feat.	M-RQ2	U-RQ2	M-RQ3	%	Relv.	Over.
Guava	110	22	14	91	95	0.45	0.00
HttpClient	38	9	7	24	77	0.50	0.05
JFreeChart	70	8	5	55	85	0.35	0.02
JSoup	81	11	11	52	74	0.60	-0.21
PDFBox	44	8	7	32	86	0.42	0.01
Apache-POI	81	10	9	53	74	0.50	-0.10
Quartz	31	12	5	19	73	0.48	0.00

The majority of our uncovered features are also used in GitHub client projects.

Evaluation

RQ3. To which extent do the uncovered features that do not match documented tutorial features correspond to actual API usage in client projects?

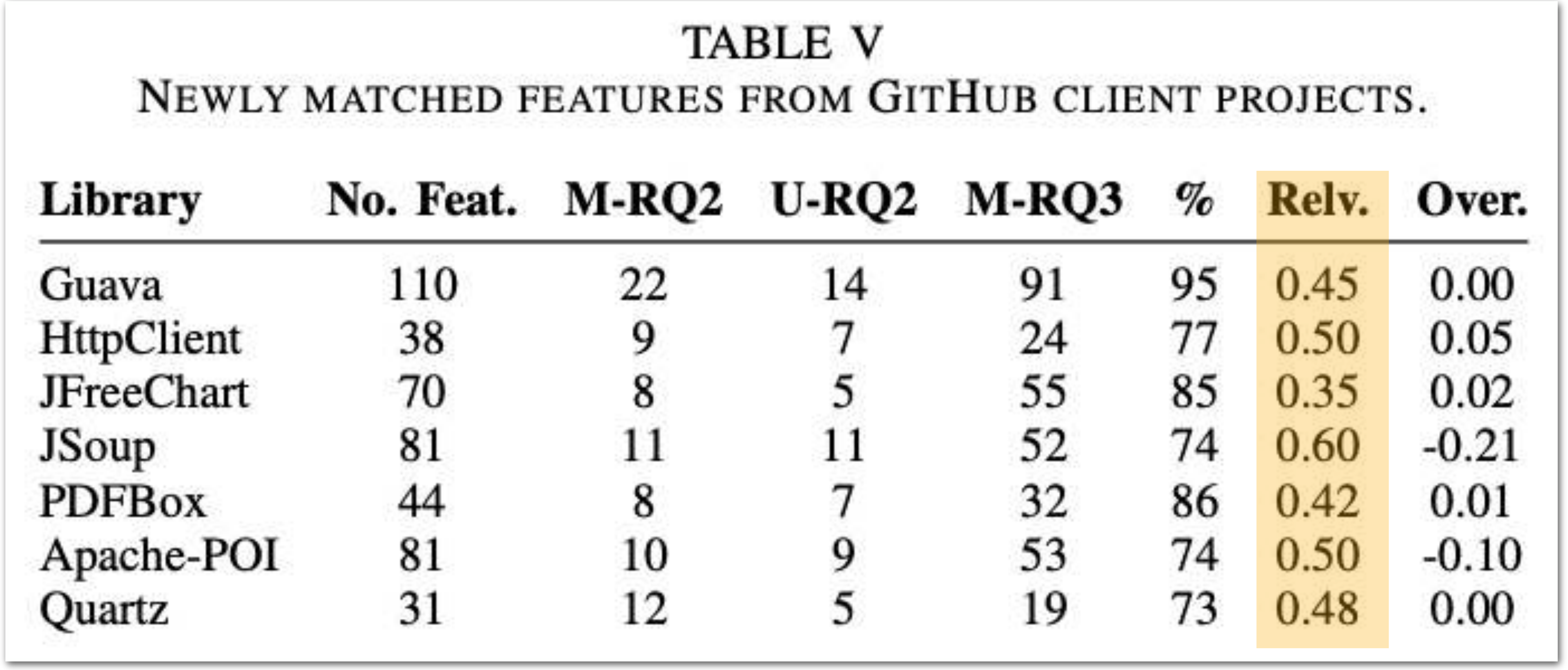


TABLE V

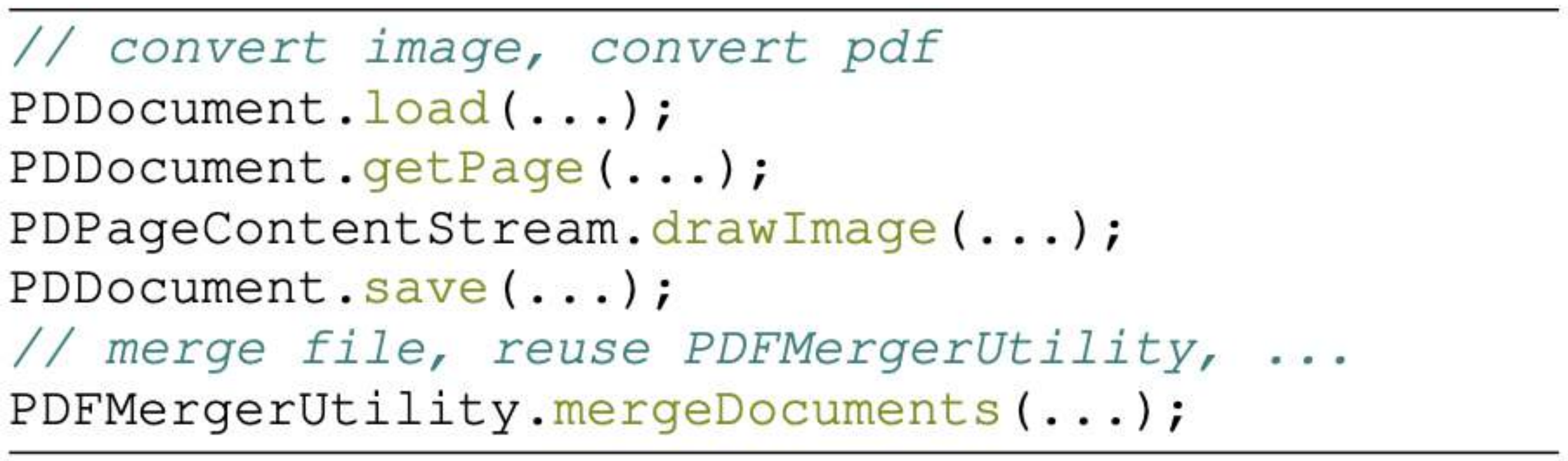
NEWLY MATCHED FEATURES FROM GITHUB CLIENT PROJECTS.

Library	No. Feat.	M-RQ2	U-RQ2	M-RQ3	%	Relv.	Over.
Guava	110	22	14	91	95	0.45	0.00
HttpClient	38	9	7	24	77	0.50	0.05
JFreeChart	70	8	5	55	85	0.35	0.02
JSoup	81	11	11	52	74	0.60	-0.21
PDFBox	44	8	7	32	86	0.42	0.01
Apache-POI	81	10	9	53	74	0.50	-0.10
Quartz	31	12	5	19	73	0.48	0.00

The decrease in the relevance might be caused by a less-focused usage of the features in GitHub.

Examples of uncovered features

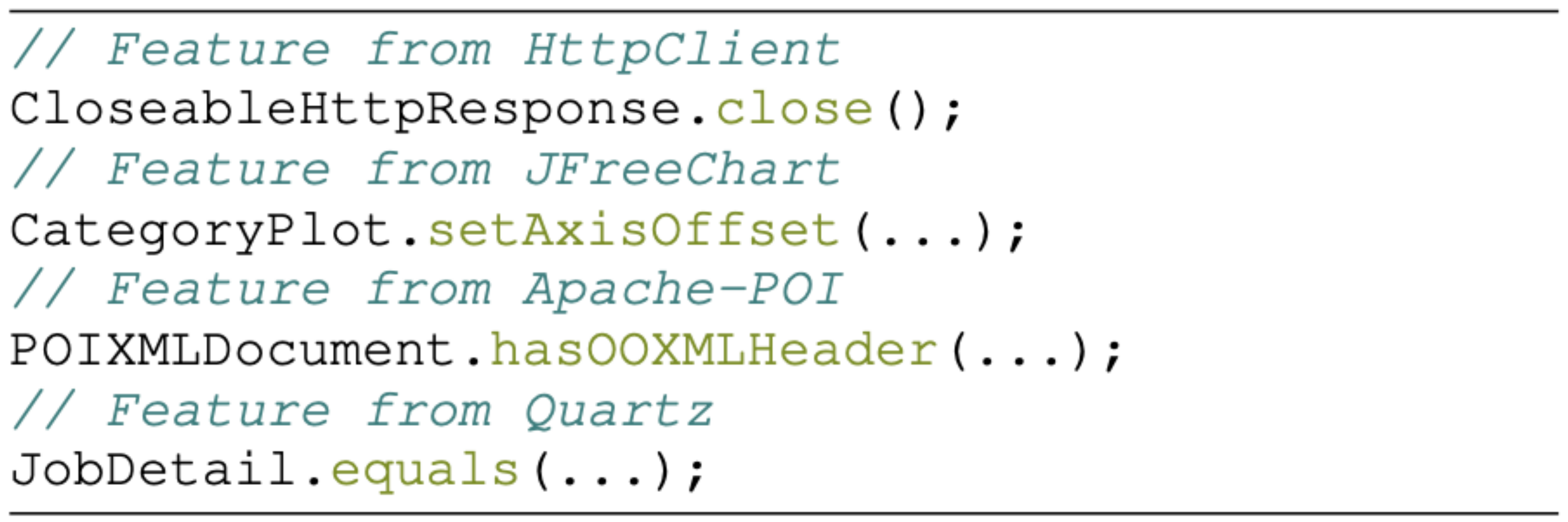
Good features



```
// convert image, convert pdf
PDDocument.load(...);
PDDocument.getPage(...);
PDPageContentStream.drawImage(...);
PDDocument.save(...);
// merge file, reuse PDFMergerUtility, ...
PDFMergerUtility.mergeDocuments(...);
```

PDFBox

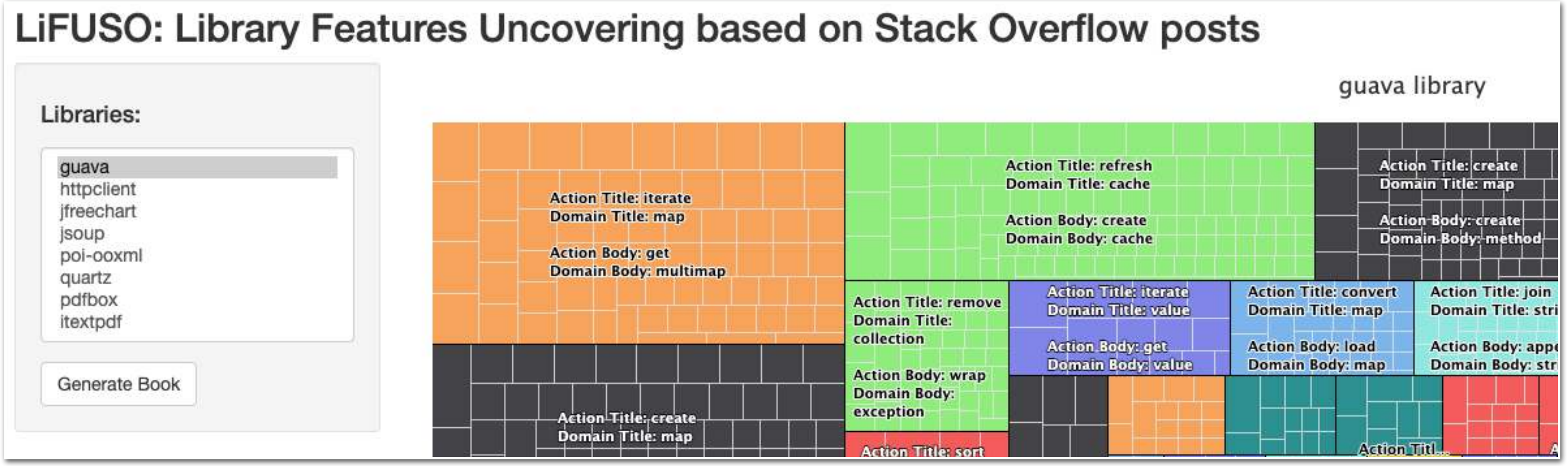
Bad features



```
// Feature from HttpClient
CloseableHttpResponse.close();
// Feature from JFreeChart
CategoryPlot.setAxisOffset(...);
// Feature from Apache-POI
POIXMLDocument.hasOOXMLHeader(...);
// Feature from Quartz
JobDetail.equals(...);
```

Several libraries

Tool LiFUSO



LiFUSO: Library Features Uncovering based on Stack Overflow posts

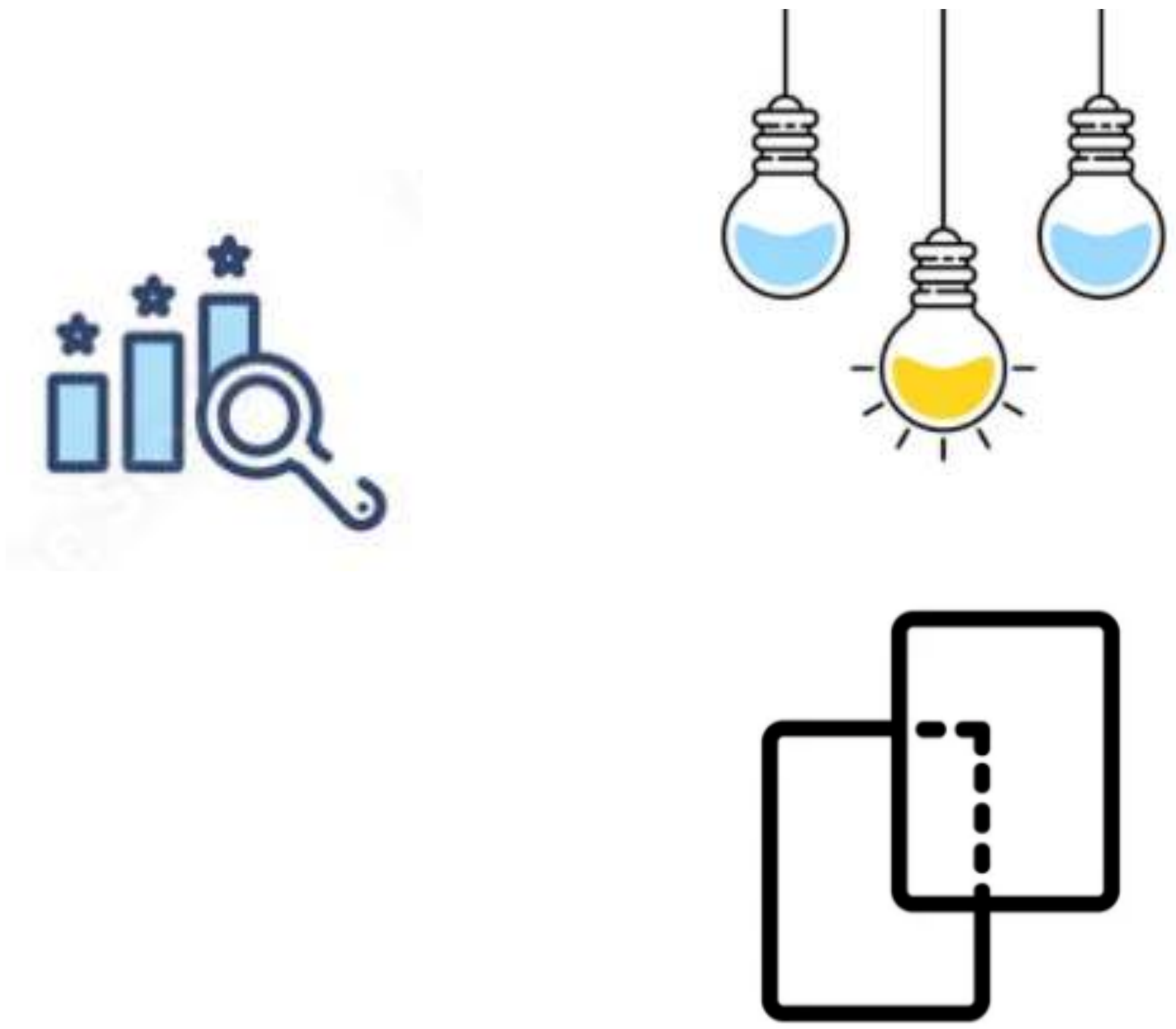
https://github.com/softwarelanguageslab/lifuso

Potential application

Maintainers

Potential application

Maintainers

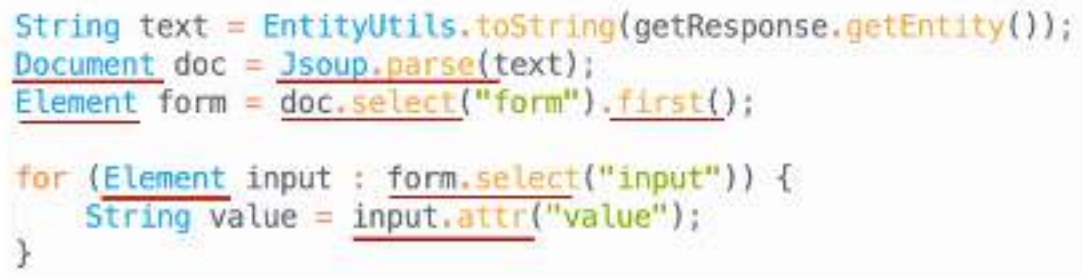


Users

Conclusion

Introduction

JSoup library



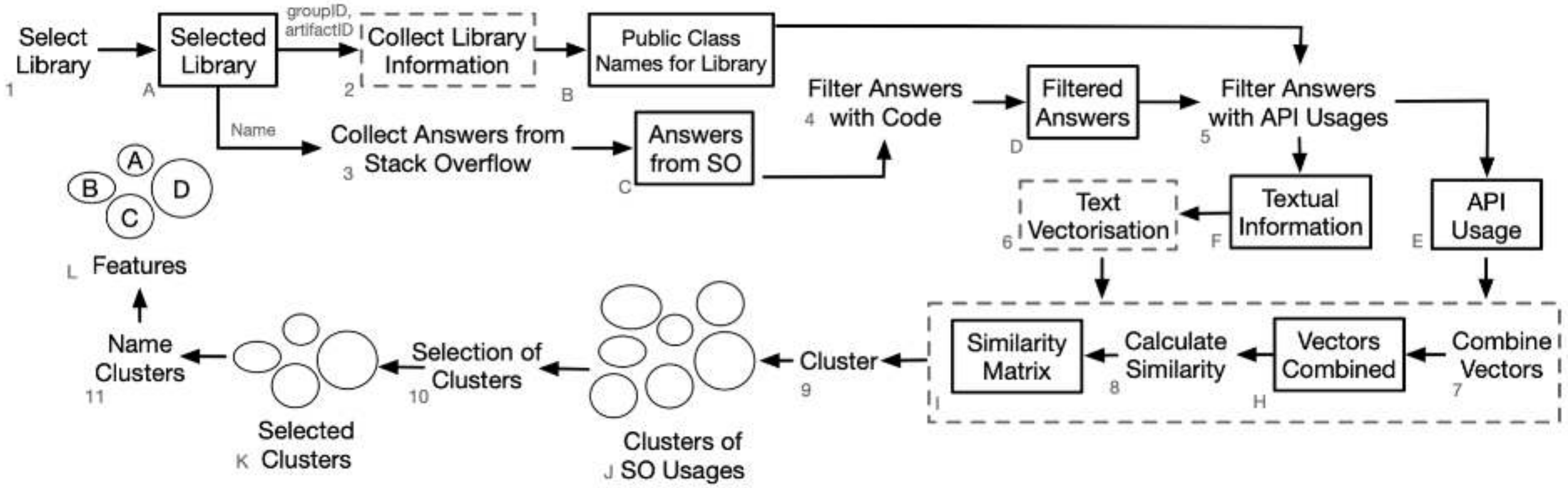
```
String text = EntityUtils.toString(getResponse.getEntity());
Document doc = Jsoup.parse(text);
Element form = doc.select("form").first();

for (Element input : form.select("input")) {
    String value = input.attr("value");
}
```

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities

⋮

Approach



Evaluation

RQ1. Which combination of SO answer attributes produces the most cohesive clusters?

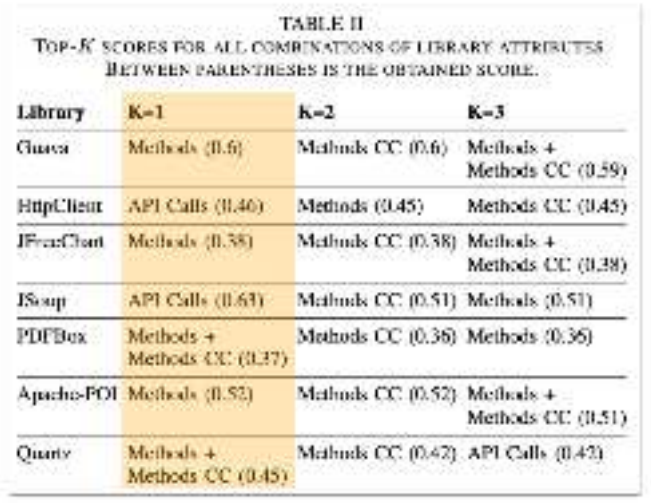


TABLE II
TOP-K SCORES FOR ALL COMBINATIONS OF LIBRARY ATTRIBUTES BETWEEN PARENTHESES IS THE OBTAINED SCORE.

Library	K=1	K=2	K=3
Guava	Methods (0.6)	Methods CC (0.6)	Methods + Methods CC (0.59)
HttpClient	API Calls (0.46)	Methods (0.45)	Methods CC (0.45)
JFreeChart	Methods (0.38)	Methods CC (0.38)	Methods + Methods CC (0.38)
JSoup	API Calls (0.63)	Methods CC (0.51)	Methods (0.51)
PDFBox	Methods + Methods CC (0.37)	Methods CC (0.36)	Methods (0.36)
Apache-POI	Methods (0.53)	Methods CC (0.52)	Methods + Methods CC (0.51)
Quartz	Methods + Methods CC (0.45)	Methods CC (0.42)	API Calls (0.42)

The **Methods** combination from the API usage information attributes produced the most cohesive clusters.

RQ2. How similar are the automatically uncovered features to documented tutorial features?

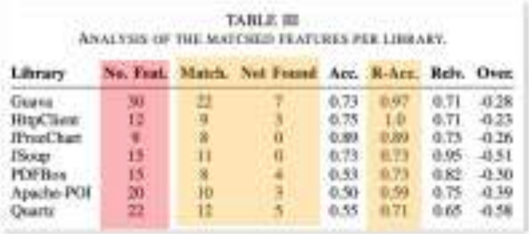


TABLE III
ANALYSIS OF THE MATCHED FEATURES PER LIBRARY.

Library	No. Feat.	Match.	Not Found	Acc.	R-Acc.	Relv.	Over.
Guava	30	22	7	0.73	0.97	0.71	-0.28
HttpClient	12	9	3	0.75	1.0	0.71	-0.23
JFreeChart	9	8	0	0.89	0.89	0.73	-0.26
JSoup	15	11	0	0.73	0.73	0.95	-0.51
PDFBox	15	8	4	0.53	0.73	0.82	-0.30
Apache-POI	20	10	3	0.50	0.59	0.75	-0.39
Quartz	22	12	5	0.55	0.71	0.65	-0.58

When considered against the usages in the SO data, the matched accuracy increases.

RQ3. To which extent do the uncovered features that do not match documented tutorial features correspond to actual API usage in client projects?

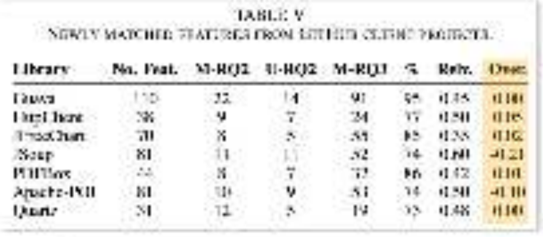


TABLE V
NEWLY MATCHED FEATURES FROM GITHUB CLIENT PROJECTS.

Library	No. Feat.	M-RQ2	U-RQ2	M-RQ3	%	Relv.	Over.
Guava	[illegible]	22	14	91	95	0.45	0.00
HttpClient	36	9	7	24	77	0.50	0.16
JFreeChart	30	8	5	34	85	0.55	0.12
JSoup	81	11	11	52	74	0.60	-0.21
PDFBox	[illegible]	8	7	37	86	0.42	0.01
Apache-POI	81	10	9	51	74	0.50	-0.10
Quartz	31	12	5	19	55	0.48	0.00

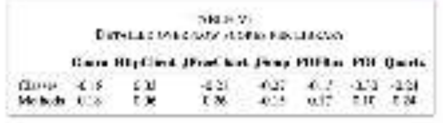


TABLE VI
[illegible]

	Guava	HttpClient	JFreeChart	JSoup	PDFBox	POI	Quartz
Classes	-0.15	0.30	-0.21	-0.27	[illegible]	-0.23	-0.21
Methods	[illegible]	0.36	0.36	[illegible]	0.17	0.10	0.24

A closer look to the overflow metric indicates that GitHub usages might encompass more than one of our uncovered features.

Examples of uncovered features

Good features

```
// [illegible]
PDDocument.load(...);
[illegible]
[illegible]
PDDocument.save(...);
// [illegible]
[illegible]
```

PDFBox

Bad features

```
// Features from [illegible]
[illegible]
// Features from [illegible]
[illegible]
// Features from Apache POI
[illegible]
// Features from Quartz
[illegible]
```

Several libraries

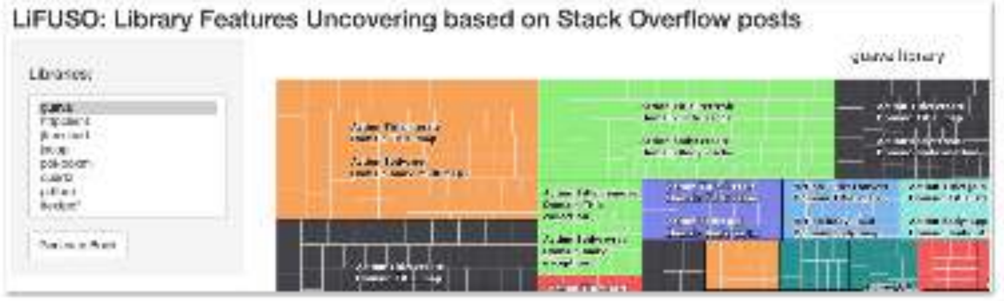
Potential application

Maintainers

Users

Search...

Tool LiFUSO



LiFUSO: Library Features Uncovering based on Stack Overflow posts