Received: 19 December 2019 | Revised: 8 August 2020 | Accepted: 2 September 2020

DOI: 10.1002/spe.2906

RESEARCH ARTICLE

WILEY

APISonar: Mining API usage examples

Andre Hora

Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, Brazil

Correspondence
Andre Hora, Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, Brazil.
Email: andrehora@dcc.ufmg.br

Abstract

Developers spend a significant part of their time searching for code examples on the web. Often, they look for Application Programming Interface (API) usage examples, that is, how to use APIs provided by libraries and frameworks. For this purpose, several programming websites are available. Some programming websites provide manually created examples: unfortunately, as millions of APIs are available nowadays, they do not cover the majority of the APIs. To alleviate this limitation, other programming websites focus on automatically mining API usage examples from code repositories. To the best of our knowledge, however, these solutions are still very limited: they often present poor, duplicated, and similar API usage examples. In this article, we propose an approach, APISonar, to automatically mine API usage examples from code repositories. Our approach aims to overcome the limitations of current solutions: we focus on presenting readable and reusable API usage examples. We analyze millions of source files provided by 4486 software projects hosted on GitHub. Based on this data, we extract 11 million API usage examples about 1.5 million distinct APIs. We evaluate APISonar by assessing its quality and usage. We show that APISonar is a competitive solution, providing the best API examples in terms of readability and reusability, as compared with popular programming websites. Moreover, despite being a novel website, APISonar attracted a significant amount of users in a short period (3.7K users from 119 countries during 5 months). APISonar is available at www.apisonar.com.

KEYWORDS

API usage examples, framework, library, mining software repositories, software maintenance

1 | INTRODUCTION

Developers often look for code examples to support their programming activities.[1-4] Prior work reports that developers may spend up to 20% of their time searching for code on the web.[5] Indeed, code examples provide several benefits: they improve learning, support reuse, and accelerate development.[6,7] Typically, developers are interested in Application Programming Interface (API) usage examples, that is, code examples about how to use APIs provided by frameworks and libraries.[8-10] Ideally, code examples should have some characteristics to be more useful for developers. For instance, they should be concise and without implementation details to be easier to understand and reuse.[11] For this purpose, several research studies are proposed to find, classify, and rank code examples.[10,12-18]

In practice, to support code search on the web, many programming websites are available,[19] such as Stack Overflow,[20] W3Schools,[21] ProgramCreek,[22] and Codota,[23] to name a few. These programming websites are very popular nowadays:

Stack Overflow, for instance, has more than 50 million users per month, W3Schools had 2.5 billion pageviews in 2018, and Codota has more than 1.5 million webpages indexed by Google. The code examples presented by them have a distinct origin. While some programming websites provide manually created code examples, others propose to automatically extract them from code repositories, such as GitHub. For instance, W3Schools presents examples manually created for didactic purposes explaining how to use certain APIs. By contrast, in Codota and ProgramCreek, the API examples are automatically extracted code repositories.

However, current solutions to find API usage examples on the web are still very limited. They often present poor, duplicated, and similar examples, covering a limited number of APIs. For instance, by running a state-of-the-art measure to assess code readability[24] on some examples returned by ProgramCreek and Codota, we find very low readability, which may hamper their adoption by developers (we provide more details in Section 2.1). Moreover, it is not uncommon to see duplicated and similar code examples returned by these tools. This is an issue because, in practice, developers often inspect multiple results of different usages to learn from,[18,25] thus, having multiple but irrelevant examples is not useful (we provide concrete cases in Section 2.2). Finally, due to the fact that some popular programming websites present manually created API examples, they may not scale to cover hundreds of thousands of existing APIs (see Section 2.3).

To overcome the limitations of current solutions, in this article, we propose APISonar, an approach to automatically mine API usage examples from code repositories and present API examples based on the API name. We focus on presenting readable, reusable, and unique API usage examples. Furthermore, our mining process looks for examples of present and past versions of code repositories, so we cover both actual and legacy APIs. In this study, we analyze 11.5 million source files provided by 4486 software projects hosted on GitHub. Based on this data, we extract 11 million API usage examples about 1.5 million distinct APIs. APISonar is available at http://apisonar.com.

Our approach takes as input a set of projects, extracts API usage examples from these projects, and returns as output the website hosting all code examples. In short, it includes four major steps: (1) collecting API usage examples, (2) filtering API usage examples, (3) ranking API usage examples, and (4) presenting API usage examples (Section 3). First, we collect examples from current and past versions of software projects (Section 4). Then, we remove duplicated and similar examples (Section 5.1), ranking them based on three quality measures: API similarity to the code example, code readability, and code reusability (Section 5.2). Finally, we automatically build the APISonar website to present the mined data (Section 6).

We evaluate the proposed approach in three scenarios to assess quality and coverage (Section 7). First, we analyze the quality of the API examples returned by APISonar as compared with the ones provided by three popular programming websites (ProgramCreek, Codota, and SearchCode[26]). Second, we analyze the quality of the API examples provided by APISonar as compared with the ones returned by API code searches on Google. Finally, we assess the coverage of the APIs provided by APISonar. Our results show that APISonar provides the best API examples in terms of readability and reusability, covering a large amount of relevant APIs. We also assess the usage of APISonar for 5 months (Section 8). Despite being a novel website, APISonar has attracted thousands of real users (3.7K users from 119 countries and 1193 cities). Moreover, APISonar has 25K webpages indexed by Google, suggesting that it provides relevant and useful content to the search engine. Therefore, the contributions of this work are threefold:

- We provide a new programming website that hosts millions of API usage examples (http://apisonar.com).
- We propose a novel approach to extract, filter, and rank API usage examples from code repositories.
- We provide an empirical study to assess the quality and usage of APISonar.

The remainder of this article is organized as follows. Section 9 details the threats to validity, Section 10presents the related work, and Section 11 concludes the article.

2 | MOTIVATION

Developers often rely on API usage examples to support development.[8-10] For this purpose, many programming websites are available. However, we find that API examples available on the web present several limitations. First, it is not uncommon to spot programming websites with poor examples, that is, with low readability and reusability. Second, often, these websites provide duplicated, similar, and irrelevant examples. Finally, the examples are more likely to cover popular APIs, lacking older and legacy ones. In the following subsections, we detail these limitations in the light of popular programming websites.

2.1 | Examples with poor quality

ProgramCreek[22] and Codota[23] are two popular programming websites that focus on presenting API usage examples. In common, they provide hundreds of thousands of examples extracted from code repositories. ProgramCreek has about 500K webpages indexed by Google, while Codota has over 1.5 million, thus, Java API searches on the web are very likely to find these websites. For example, by querying *"javax.swing.JFrame example"* on Google, both programming websites appear in the top 10 search results.

As an illustrative assessment, we analyze code examples for two highly popular APIs in the Java ecosystem: `java.util.List.add` and `java.util.List.addAll`. Figure 1(A) presents the first example provided by ProgramCreek for the API `java.util.List.add`. We verify two aspects in this example: it is large (25 lines) and it is mixed with external code (notice the references to `ButterKnife`, `GridLayoutManager`, `LinearLayoutManager`, and so forth). In fact, those two aspects are against the characteristics of good code examples, which say they

Project: *GitHub* File: *RecyclerViewFragment.java* View Source Code

```
@Override
public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    ButterKnife.bind(this, view);

    final List<Object> items = new ArrayList<>();

    for (int i = 0; i < ITEM_COUNT; ++i) {
        items.add(new Object());
    }


    //setup materialviewpager

    if (GRID_LAYOUT) {
        mRecyclerView.setLayoutManager(new GridLayoutManager(getActivity(), 2));
    } else {
        mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
    }
    mRecyclerView.setHasFixedSize(true);

    //Use this now
    mRecyclerView.addItemDecoration(new MaterialViewPagerHeaderDecorator());
    mRecyclerView.setAdapter(new TestRecyclerViewAdapter(items));
}
```

(A) First example for java.util.List.add in ProgramCreek.

WebSocketTransportRegistration.setDecoratorFactories(...) origin: spring-projects/spring-framework

```
/**
 * Configure one or more factories to decorate the handler used to process
 * WebSocket messages. This may be useful in some advanced use cases, for
 * example to allow Spring Security to forcibly close the WebSocket session
 * when the corresponding HTTP session expires.
 * @since 4.1.2
 */
public WebSocketTransportRegistration setDecoratorFactories(WebSocketHandlerDecoratorFactory... factories) {
  this.decoratorFactories.addAll(Arrays.asList(factories));
  return this;
}
```

(B) First example for java.util.List.addAll in Codota.

FIGURE 1 Poor quality code examples [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 1 Code readability for popular APIs

APIs	Readability of the first example in …	
	ProgramCreek	Codota
`java.util.List.add`	0.02	0.38
`java.util.List.addAll`	0.00	0.03
`java.io.File.exists`	0.01	0.03
`java.io.File.mkdirs`	0.00	0.95
`org.apache.commons.io.FilenameUtils.getExtension`	0.02	0.03
`org.apache.commons.io.FilenameUtils.concat`	0.03	0.00
`com.google.common.io.Files.write`	0.00	0.03
`com.google.common.io.Files.createTempDir`	0.00	0.89
`org.springframework.context.ApplicationContext.getBean`	0.00	0.03
`org.springframework.context.ApplicationContext.publishEvent`	0.85	0.11

Abbreviation: API, application programming interfaces.

should be concise and without implementation details.[11] Figure 1(B) shows the first example provided by Codota for the API `java.util.List.addAll`. By contrast to the previous one, this example is much smaller (only 4 lines of code), however, it still has references to external code (i.e., `WebSocketTransportRegistration`).

Altogether, these problems make those examples harder to understand and reuse. Indeed, by running a state-of-the-art measure on code readability,[24] both examples present very low readability: the former has a readability of 2% and the latter of 3% (in a scale of 0% to 100%, in which 0 represents the lowest readability and 100% means the highest). To further investigate such low readability, we run the same metric on the first example returned by both ProgramCreek and Codota for 10 popular APIs provided by Java, Apache Commons, Google Guava, and Spring Framework (Table 1). From the 20 measures presented in the table, there are only three cases with high readability (≥ 0.85), while the majority has readability close to zero.

> Popular programming websites that focus on API usage examples, such as ProgramCreek and Codota, may provide poor examples to the users in terms of readability and reusability, making them harder to understand and reuse.

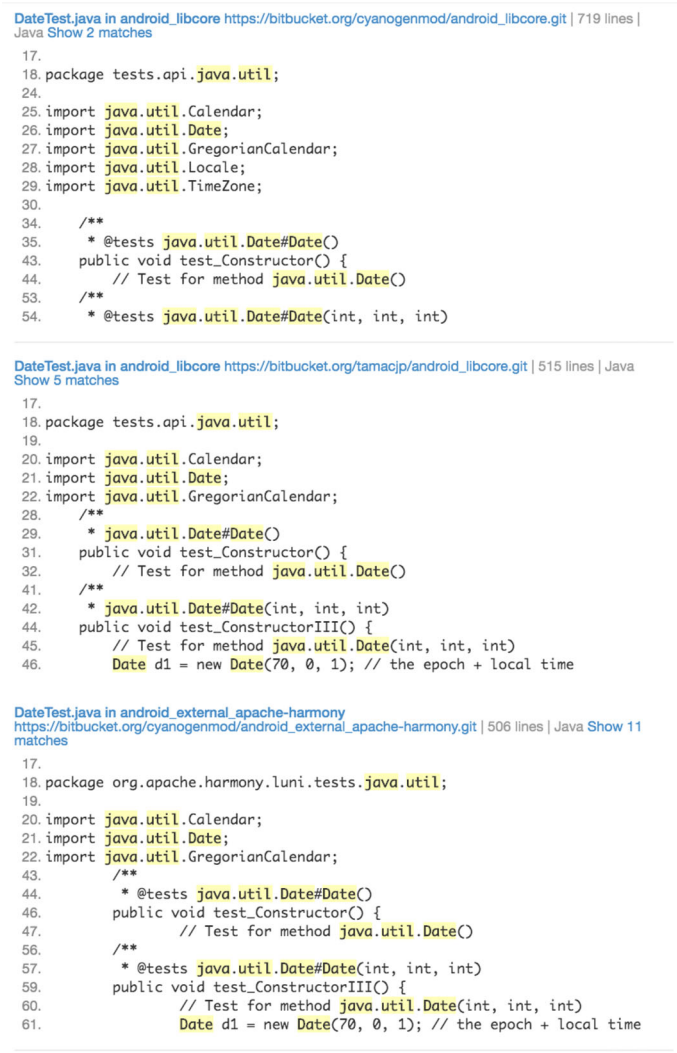
2.2 | Duplicated, similar, and irrelevant examples

In practice, developers often inspect multiple results of different usages to learn from,[18,25] thus, having a webpage with several relevant examples may facilitate this task. In this context, consider now the popular code search engine SearchCode.[26] Different from ProgramCreek and Codota, which focus on APIs, SearchCode is a search engine in which developers can perform any code query. For example, one may query for "*public int a*," "*x == null*," or simply look for an API example. We performed queries for the popular Java API `java.util.Date.getYear` on SearchCode and ProgramCreek, and inspected their top 3 results. As shown in Figure 2, both websites returned noisy examples. Among the top 3 results, SearchCode returned one irrelevant and two duplicated examples (Figure 2(A)). By inspecting more results, we notice that the duplicated examples continue until the top 5.[1] In ProgramCreek, there are two duplications in the top 3 results (Figure 2(B)). Moreover, by inspecting the top 5, the examples are very similar to each other, therefore, they do not bring new solutions to the users.[2]

[1] www.searchcode.com/?q=java.util.Date%2BgetYear%26lan=23
[2] www.programcreek.com/java-api-examples/?class=java.util.Date%26method=getYear

In fact, the presented issues are not rare: we found dozens of similar cases by looking for examples for popular APIs on SearchCode,[3] ProgramCreek,[4] and also Codota.[5]

> Popular programming websites, such as SearchCode, ProgramCreek, and Codota, may provide duplicated, similar, and irrelevant examples, making the inspection of the returned results repetitive and less effective.



DateTest.java in android_libcore https://bitbucket.org/cyanogenmod/android_libcore.git | 719 lines | Java Show 2 matches

```
17.
18. package tests.api.java.util;
24.
25. import java.util.Calendar;
26. import java.util.Date;
27. import java.util.GregorianCalendar;
28. import java.util.Locale;
29. import java.util.TimeZone;
30.
34.     /**
35.      * @tests java.util.Date#Date()
43.     public void test_Constructor() {
44.         // Test for method java.util.Date()
53.     /**
54.      * @tests java.util.Date#Date(int, int, int)
```

DateTest.java in android_libcore https://bitbucket.org/tamacjp/android_libcore.git | 515 lines | Java Show 5 matches

```
17.
18. package tests.api.java.util;
19.
20. import java.util.Calendar;
21. import java.util.Date;
22. import java.util.GregorianCalendar;
28.     /**
29.      * java.util.Date#Date()
31.     public void test_Constructor() {
32.         // Test for method java.util.Date()
41.     /**
42.      * java.util.Date#Date(int, int, int)
44.     public void test_ConstructorIII() {
45.         // Test for method java.util.Date(int, int, int)
46.         Date d1 = new Date(70, 0, 1); // the epoch + local time
```

DateTest.java in android_external_apache-harmony https://bitbucket.org/cyanogenmod/android_external_apache-harmony.git | 506 lines | Java Show 11 matches

```
17.
18. package org.apache.harmony.luni.tests.java.util;
19.
20. import java.util.Calendar;
21. import java.util.Date;
22. import java.util.GregorianCalendar;
43.         /**
44.          * @tests java.util.Date#Date()
46.         public void test_Constructor() {
47.                 // Test for method java.util.Date()
56.         /**
57.          * @tests java.util.Date#Date(int, int, int)
59.         public void test_ConstructorIII() {
60.                 // Test for method java.util.Date(int, int, int)
61.                 Date d1 = new Date(70, 0, 1); // the epoch + local time
```

(A) Top 3 examples for java.util.Date.getYear in SearchCode.

Example 1

Project: javaide File: ZioEntry.java View Source Code — Vote up 8 votes

```
public void setTime(long time) {
    Date d = new Date(time);
    long dtime;
    int year = d.getYear() + 1900;
    if (year < 1980) {
        dtime = (1 << 21) | (1 << 16);
    }
    else {
        dtime = (year - 1980) << 25 | (d.getMonth() + 1) << 21 |
        d.getDate() << 16 | d.getHours() << 11 | d.getMinutes() << 5 |
        d.getSeconds() >> 1;
    }

    modificationDate = (short)(dtime >> 16);
    modificationTime = (short)(dtime & 0xFFFF);
}
```

Example 2

Project: mobile-store File: ZioEntry.java View Source Code — Vote up 6 votes

```
public void setTime(long time) {
    Date d = new Date(time);
    long dtime;
    int year = d.getYear() + 1900;
    if (year < 1980) {
        dtime = (1 << 21) | (1 << 16);
    }
    else {
        dtime = (year - 1980) << 25 | (d.getMonth() + 1) << 21 |
        d.getDate() << 16 | d.getHours() << 11 | d.getMinutes() << 5 |
        d.getSeconds() >> 1;
    }

    modificationDate = (short)(dtime >> 16);
    modificationTime = (short)(dtime & 0xFFFF);
}
```

Example 3

Project: GitHub File: TimeUtil.java View Source Code — Vote up 5 votes

```
/**根据生日获取年龄
 * @param birthday
 * @return
 */
public static int getAge(Date birthday) {
        if (birthday == null) {
                return 0;
        }
        if (birthday.getYear() > getDateDetail(System.currentTimeMillis())
[0]) {
                birthday.setYear(birthday.getYear() -
 TimeUtil.SYSTEM_START_DATE[0]);
        }

        return getAge(new int[]{birthday.getYear(), birthday.getMonth(),
 birthday.getDay()});
}
```

(B) Top 3 examples for java.util.Date.getYear in ProgramCreek.

FIGURE 2 Duplicated, similar, and irrelevant code examples [Color figure can be viewed at wileyonlinelibrary.com]

2.3 | Limited examples

In addition to the programming websites aforementioned in this section, others focus on providing didactic code examples. In this case, the examples are not automatically extracted from real software systems, but manually created for

[3] For example, www.searchcode.com/?q=java.util.List%2Badd%26lan=23 and www.searchcode.com/?q=com.google.common.io.Files%2Bwrite%26lan=23

[4] For example, www.programcreek.com/java-api-examples/?class=ims.framework.enumerations.SortOrder%26method=ASCENDING

[5] For example, www.codota.com/code/java/methods/org.quartz.Scheduler/shutdown and www.codota.com/code/java/methods/android.app.Activity/onCreateOptionsMenu

How about this?

30

```
Integer[] ints = new Integer[] {1,2,3,4,5};
List<Integer> list = Arrays.asList(ints);
```

share improve this answer

(A) Didactic code example for Arrays.asList in Stack Overflow.

```
import java.util.ArrayList;

public class MyClass {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```

(B) Didactic code example for ArrayList.add in W3Schools.

```
class GFG {

    // Driver's code
    public static void main(String[] args)
    {

        // Creating 2 Integer arrays
        int[] arr1 = { 1, 2, 3, 4, 5 };
        int[] arr2 = { 6, 2, 7, 0, 8 };

        // Using Ints.concat() method to combine
        // elements from both arrays into a single array
        int[] res = Ints.concat(arr1, arr2);

        // Displaying the single combined array
        System.out.println("Combined Array: "
                           + Arrays.toString(res));
    }
}
```

(C) Didactic code example for Ints.concat in GeeksforGeeks.

FIGURE 3 Didactic code examples [Color figure can be viewed at wileyonlinelibrary.com]

didactic purposes, as shown in Figure 3. Programming websites as Stack Overflow, W3Schools, and GeeksforGeeks focus on this type of code example. In fact, these websites are very popular and grab the attention of developers. Stack Overflow, for instance, has more than 50 million users per month and is the 35th most accessed website worldwide. W3Schools, which claims to be the largest web developer site, had 2.5 billion pageviews in 2018 and is the 138th most accessed website. Although the didactic code examples can be cleaner than the ones extracted from real software systems, a key limitation is that they must be manually implemented. As millions of APIs may exist nowadays, manually creating code examples is a task that certainly does not scale. For instance, by assessing an ultra-large-scale dataset with thousands of Java projects,[27] we detect the usage of 682K distinct APIs. However, by inspecting W3Schools and GeeksforGeeks, we notice that the majority of the API examples they provide are about the Java Library itself,[6]not covering external libraries and frameworks.

> Programming websites as Stack Overflow, W3Schools, and GeeksforGeeks provide code examples that are created for didactic purposes. However, manually creating code examples is a task that does not scale since hundreds of thousands of APIs exist nowadays. Thus, this type of website may not include code examples for the vast majority of available APIs.

2.4 | Summary

We have seen several limitations of current programming websites that are largely adopted by developers. In short, these problems are related to the quality, duplication, and limitation of API usage examples. Our approach, APISonar, aims to

[6]For example: https://www.w3schools.com/java/java_examples.asp and https://www.geeksforgeeks.org/java

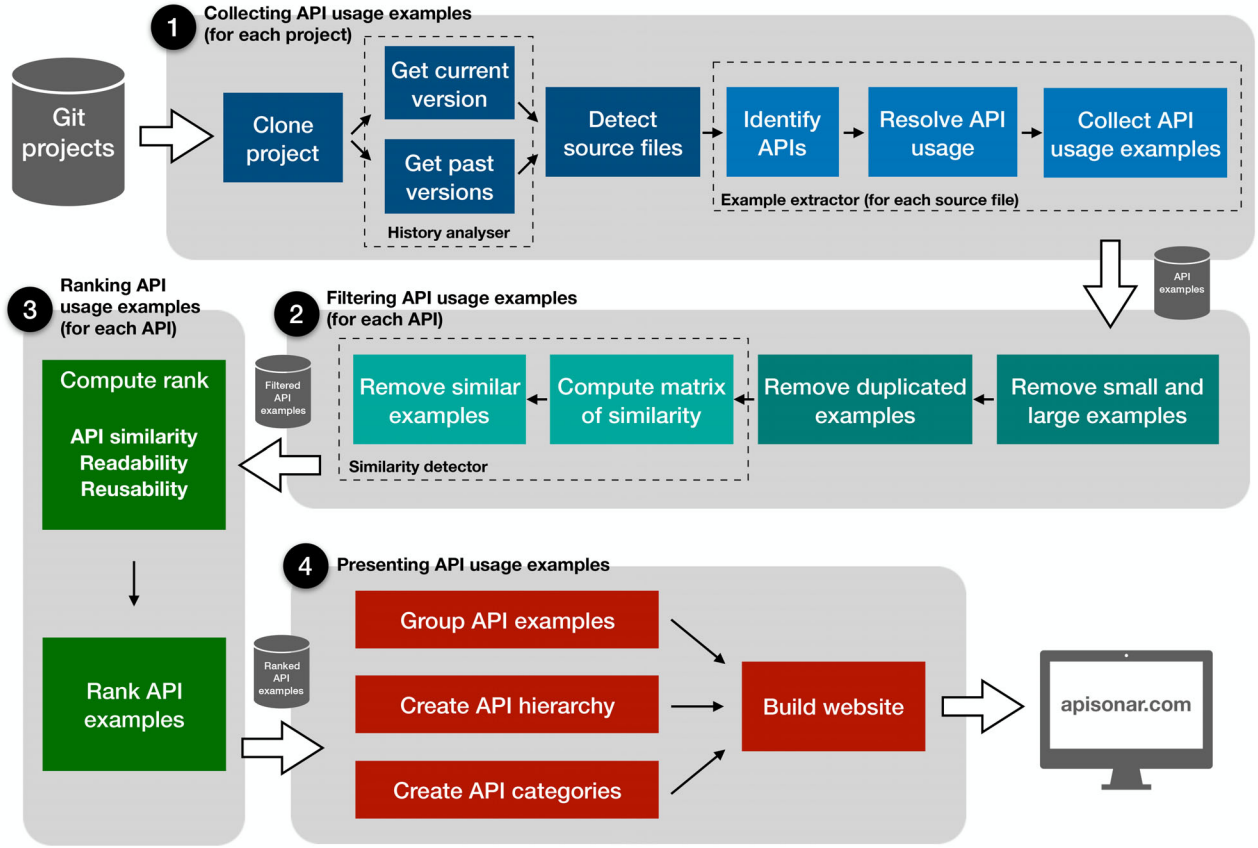


FIGURE 4 Overview of the proposed approach: APISonar [Color figure can be viewed at wileyonlinelibrary.com]

overcome these issues by properly collecting, filtering, and ranking code examples. To realistically help developers, we propose to make this curated data available on a new programming website that may compete with the ones presented in this section.

3 | OVERVIEW OF THE APPROACH

Our approach, APISonar, takes as input a set of projects to be analyzed, extracts API usage examples from these projects, and returns as output the website hosting all code examples. Figure 4 presents an overview of the proposed approach, which includes four major steps: (1) collecting API usage examples, (2) filtering API usage examples, (3) ranking API usage examples, and (4) presenting API usage examples. In the first step, we collect API usage examples from current and past versions of the analyzed projects (Section 4). In the second step, we remove duplicated and similar API usage examples by filtering out the ones that satisfy a set of conditions (Section 5.1). Next, in the third step, we rank the API usage examples. Here, we rely on three measures to ensure quality: API similarity to the code example, code readability, and code reusability (Section 5.2). Finally, in the fourth step, we automatically build the website to present the data mined in the previous steps (Section 6), in which users can find API examples for a given API.

In this study, we analyze millions of source files provided by 4486 software projects hosted on GitHub. Based on this data, we extract 11 million API usage examples about 1.5 million distinct APIs. APISonar is available at http://apisonar.com.

4 | COLLECTING API USAGE EXAMPLES

Figure 5 details the first step of our approach. It receives as input a set of git projects and produces as output a collection of API usage examples. This step is performed for each project, thus, we only move to the next step when we extract the API usage examples from all projects.

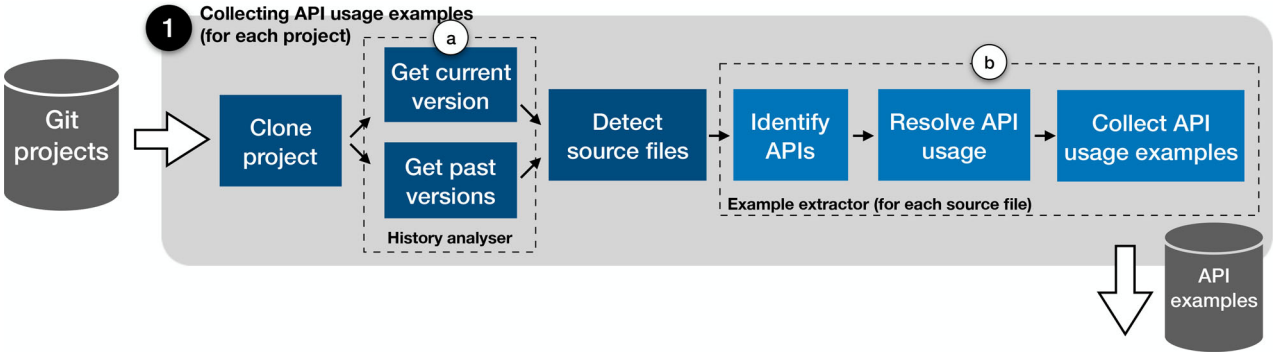


FIGURE 5 Collecting API usage examples (step 1). API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

Full API name: package.Class.method_or_field
Example 1: java.util.List.add
Example 2: com.google.common.base.Charsets.UTF_8

FIGURE 6 Format of an API in Java. API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

Our goal is to extract API usage examples from real and relevant software projects. Thus, we rely on GitHub to find software projects, as it is the most popular hosting repository nowadays. We select all Java projects hosted on GitHub with more than 500 stars, which represent 4486 projects. We adopt the star metric because it is often used in the software repository mining literature as a proxy of relevance and popularity.[28] The selected projects cover several domains, such as libraries, frameworks, tools, IDEs programming languages, databases, tutorials, and guides. Among them, we find: Elasticsearch, SpringBoot, Guava, Selenium, Gson, Hadoop, Arduino, Mockito, JUnit, and Clojure, to name a few. They come from 3043 distinct users and organizations. Apache is the organization with the most projects (87), followed by Google (46) and Alibaba (35). Other important organizations are Spring-projects (32 projects), Netflix (23), Linkedin (20), Facebook (15), and Eclipse (12).

4.1 | History analyser

As shown in Figure 5(A) (history analyzer), we clone the current version of each project. In the case a project has more than 2000 commits, we also get its past versions. Specifically, in these cases, we checkout one version for every 2000 commits. For example, if a project has 2500 commits, we get two versions: (1) version at commit 2500, that is, the current version, and (2) version at commit 500. This is intended to increase our search space, allowing us to access more APIs and examples. Moreover, this allows us to find API usage examples for current APIs and also for older ones to support developers working on legacy APIs.

Based on this approach, we collect 7201 versions. After collecting the project versions, we detect their source files (.java in our case). Considering all versions, we detect 11,509,061 java files. From these java files, 1,379,535 (12%) come from current versions, while 10,129,526 (88%) come from past versions.

4.2 | Example extractor

Next, we start the process to extract API usage examples. As shown in Figure 5(B) (example extractor), for each source file, we identify its used APIs with the support of an AST parser. In Java, APIs have the format shown in Figure 6: `package.Class.method_or_field`. Basically, an API has three parts: package, class, and method (or field). For example, the Java API `java.util.List.add` has package `java.util`, class `List`, and method `add`. The Google Guava API `com.google.common.base.Charsets.UTF_8` has package `com.google.common.base`, class `Charsets`, and field `UTF_8`.

(A) Usage of an API method

```
import java.util.List;
...
public class SortedMultisetTestSuiteBuilder {
    ...
    private List<String> getExtremeValues() {
        List<String> result = new ArrayList<>();
        result.add("!! a");
        result.add("!! b");
        result.add("~~ y");
        result.add("~~ z");
        return result;
    }
    ...
}
```

(B) Usage of an API field

```
import com.google.common.base.Charsets;
...
public class FilesTest extends IoTestCase {
    ...
    public void testCopyIdenticalFiles() throws IOException {
        File temp1 = createTempFile();
        Files.write(ASCII, temp1, Charsets.UTF_8);
        File temp2 = createTempFile();
        Files.write(ASCII, temp2, Charsets.UTF_8);
        Files.copy(temp1, temp2);
        assertEquals(ASCII, Files.toString(temp1, Charsets.UTF_8));
    }
    ...
}
```

FIGURE 7 API usage examples. API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

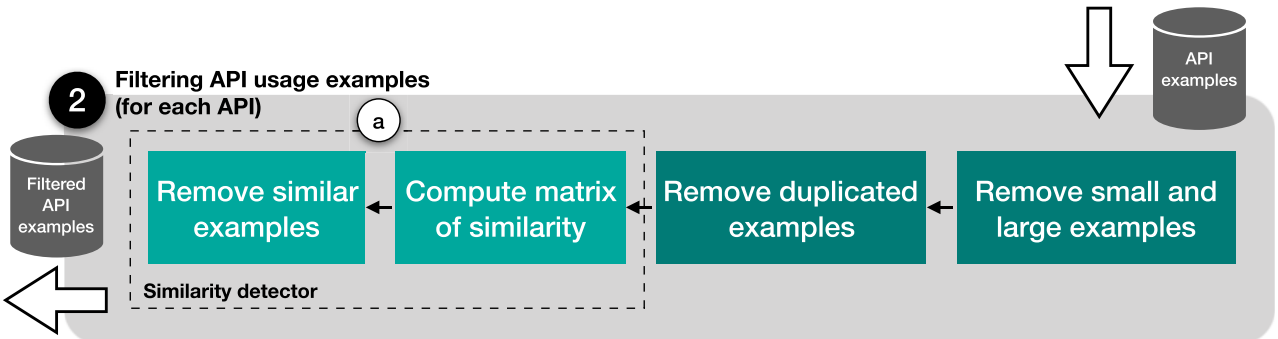


FIGURE 8 Filtering API usage examples (step 2). API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

To detect the methods in which the APIs are being used, we start by collecting all import statements in the analyzed java file. Then, we iterate over all methods looking invocations to API methods and accesses to API fields. We resolve the full API name by matching each API method and field usage to its import statement. This way, our approach does not require type resolution, only for the target APIs. Figure 7(A) presents an usage example for the API `java.util.List.add` provided by Java, while Figure 7(B) shows an usage example for the API `com.google.common.base.Charsets.UTF_8` provided by Guava. We only collect API usage examples in which both the API class and the API method (or field) explicitly appear in the code example. As we only collect a method in isolation, the examples are not required to be compilable.

As output, this step returns a list of detected and APIs with their respective examples. After processing the 11,509,061 java files, we detect 1,537,068 distinct APIs and 11,079,925 API examples.

5 | FILTERING AND RANKING API USAGE EXAMPLES

5.1 | Filtering

After collecting API usage examples, we move to the second step of our approach, filtering, as shown in Figure 8. It receives as input a collection of API usage examples, filters them, and returns a collection of filtered API usage examples.

First, we filter out large (> 15 lines of code), very small (≤3 lines of code), and duplicated code examples, so that we remain with relevant ones. Notice that two lines of code in an example refer to the signature and the closing "}," thus, with the minimum threshold 3 we ensure that the method body has at least 2 lines and we filter out examples with only one line of method body.

Furthermore, as shown in Figure 8(A) (similarity detector), we also remove similar code examples to avoid that different but close code examples are presented to developers. We measure the similarity between API examples with the metric Cosine similarity,[29] which is largely adopted in information retrieval and code search literature[13,14,18] to compare

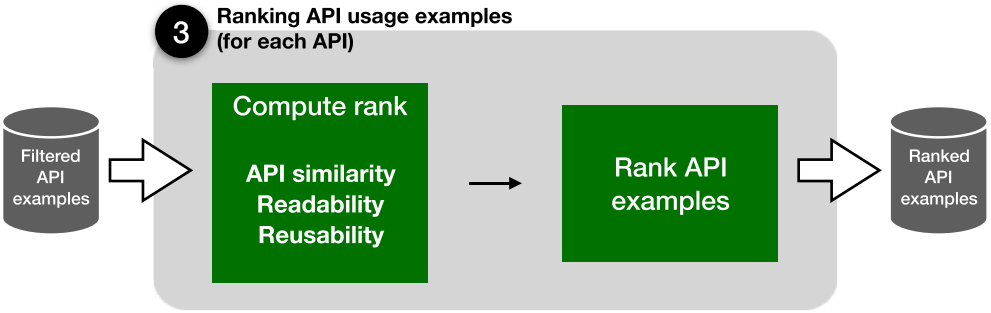


FIGURE 9 Ranking API usage examples (step 3). API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

the similarity between two documents (source code, in our case) independently of their size. We do not apply stemming to be more precise in the comparison nor identifier splitting for the same reason. For example, an identifier named `DateTimeZone` is different from `DateTime`, which is different from `Date`. Cosine similarity produces values in the range [0,1]: values close to 1 represent higher similarity, while values close to 0 represent lower similarity. For each API, we compare the similarity of its API examples and store this result in a matrix of similarity. Then, we adopt the threshold 0.70 to exclude similar API examples, that is, if two code examples have Cosine similarity ≥ 0.70, we remain with only one. This value was selected after a manual assessment of several thresholds (i.e., 0.50, 0.60, 0.70, 0.80, and 0.90) and reported the best results. We recognize, however, that other values can be adopted to ensure more or less flexibility in the removal of similar code examples.

Finally, as output, this step returns a collection of filtered API usage examples, which excludes large, small, duplicated, and similar code examples.

5.2 | Ranking

Figure 9 presents the third step of our approach: ranking API usage examples. For each API, it receives as input a collection of (filtered) API examples and returns a collection of ranked ones. Our goal in this step is to find and rank the best API usage examples. We rely on three metrics to rank the API examples: API similarity, code readability, and code reusability, as stated in the following subsections.

5.2.1 | Readability and reusability

Readability is a human judgment of how easy a text is to understand.[24] Buse and Weimer proposed a metric to evaluate the readability of a code example that was shown to be 80% effective in predicting developers' readability judgments.[24] It relies on low-level code features, such as identifiers length, number of assignments, number of loops, indentation, and so forth, to assess code readability. This metric is often referenced in the literature,[7]For instance, it was successfully adopted on a technique for ranking code examples[12] and on an approach to detect technical debit.[30] Given a code example, the readability metric produces values between 0 (lowest readability) and 1 (highest readability).

Reusability assesses the facility to reuse a given code example. Moreno et al.[12] presented a measure to evaluate reusability, which is defined as follows:

$$\text{Reusability} = \begin{cases} \frac{\#\text{native object types}}{\#\text{object types}} & \text{if } \#\text{object types} > 0 \\ 1.0 & \text{otherwise,} \end{cases} \quad (1)$$

where *#object types* is the total number of different object types used by the code example, and *#native object types* is the number of object types used by the code example and belonging to its own library or framework. The reusability metric varies from 0 to 1: 0 means that all object types in the code example are custom or external objects (low reusability), while 1 indicates that all object types in the code example belong to the target library or framework or that no object types are

[7]According to Google Scholar, to date, its article has over 200 citations.

present in the code example (high reusability).[12] The idea behind is that reusing a code example that relies on custom or external object types requires importing those objects into the code, which represents an additional work.[12]

The two aforementioned metrics to compute readability and reusability are very powerful and commonly adopted to assess code quality in the literature.[12,24,30] However, in practice, they have some limitations, which make them not suited for our approach. First, the provided implementation of the readability metric is language-dependent, that is, it only works for Java code examples. Although our focus on this study is Java, we plan to extend our approach to other popular programming languages, such as JavaScript and Python. Thus, we would need a language-independent measure of readability. The second problem is with the reusability metric. For this metric properly work, it needs to know beforehand the *native object types*. For example, to assess the reusability of a code example about an API of library X, we need to know all object types provided by library X. While this is possible to perform in a small analysis, this is impracticable in our approach due to the large size of our dataset, which includes millions of APIs. Therefore, we would need a library independent measure of reusability.

In summary, in the exact way they are implemented, unfortunately, these metrics are not suited for APISonar. However, the readability[24] metric is state-of-the-art, while the reusability[12] metric is a simple but powerful solution that objectively assesses code reuse. Thus, as presented next, we aim to find close replacements for these two measures.

5.2.2 | Assessing quality of API examples

In order to find readability and reusability metrics that are simple, language-independent, and, thus, better suited to the proposed approach, we perform the following analysis. First, we select four popular libraries and frameworks: Java Library, Google Guava, Apache Commons, and Spring Framework. For each system, we randomly select 2500 API examples, totalizing 10,000 examples. Then, we compute 8 language-independent code metrics, which are often adopted to assess code quality and documents in information retrieval: (1) lines of code, (2) number of comments, (3) number of tokens, (4) lines per token, (5) number of classes, (6) API references (aka, Term Frequency[31]), (7) API similarity (i.e., the Cosine similarity[29] between the code example and the API), and (8) reusability light (i.e., a lighter version of the original reusability metric in which the *native object types* are always the Java object types). In addition to those metrics, we also compute the original (9) readability and (10) reusability. Finally, we verify the correlation with a Spearman test on all metrics. The values of the Spearman coefficient ranges from -1 to 1. Values close to -1 or 1 represent high correlation, while values close to 0 indicates no correlation. Values less than 0.20 indicate very low correlation or no correlation, while values greater than 0.20 indicates at least a low correlation.[32]

Figure 10 presents the Spearman correlation structure of the 10 metrics when considering the 10,000 API examples. We notice that reusability is strongly correlated with reusability light (value: 0.62), while readability is moderately correlated with lines per token (value: 0.53). That is, reusability light and lines per token are a good replacement for the language-dependent measures reusability and readability, respectively.

To assess whether these correlations still hold at the system level, Figure 11 presents the structures for each system separately: (a) Java Library, (b) Google Guava, (c) Apache Commons, and (d) Spring Framework. For Java, we verify that reusability and reusability light are perfectly correlated (value 1.0); this is expected since in the lighter metric the *native object types* are always the Java object types. For the other analyses, the correlation of reusability and reusability light is 0.88 in Apache Commons (strong), 0.78 in Google Guava (strong), and 0.42 in Spring Framework (moderate). Regarding the correlation between readability and lines per token, we find the following values: 0.61 in Apache Commons (strong), 0.60 in Java (strong), 0.50 in Spring Framework (moderate), and 0.45 in Google Guava (moderate). Thus, as in the previous analysis, at the system level, we find correlations that vary from moderate to strong for both reusability and readability.

Thus, we adopt *reusability light* and *lines per token* as lightweight measures to assess the reusability and readability. Since both metrics are defined in the range [0,1], we linearly combine them to obtain the *quality score* of an API example ex_i:

$$QualityScore(ex_i) = 0.5 \times ReusabilityLight(ex_i) + 0.5 \times LinesPerToken(ex_i). \quad (2)$$

We also consider the similarity between the API and the API example (for short, API similarity) in our ranking schema. Ideally, when looking for API examples, developers may expect to see references to the desired API in the code. For this purpose, we rely on the Cosine similarity, which is often adopted in the code search literature.[13,14,18] It produces values in the range [0,1]: values close to 1 represent higher similarity, while values close to 0 represent lower similarity. This way,

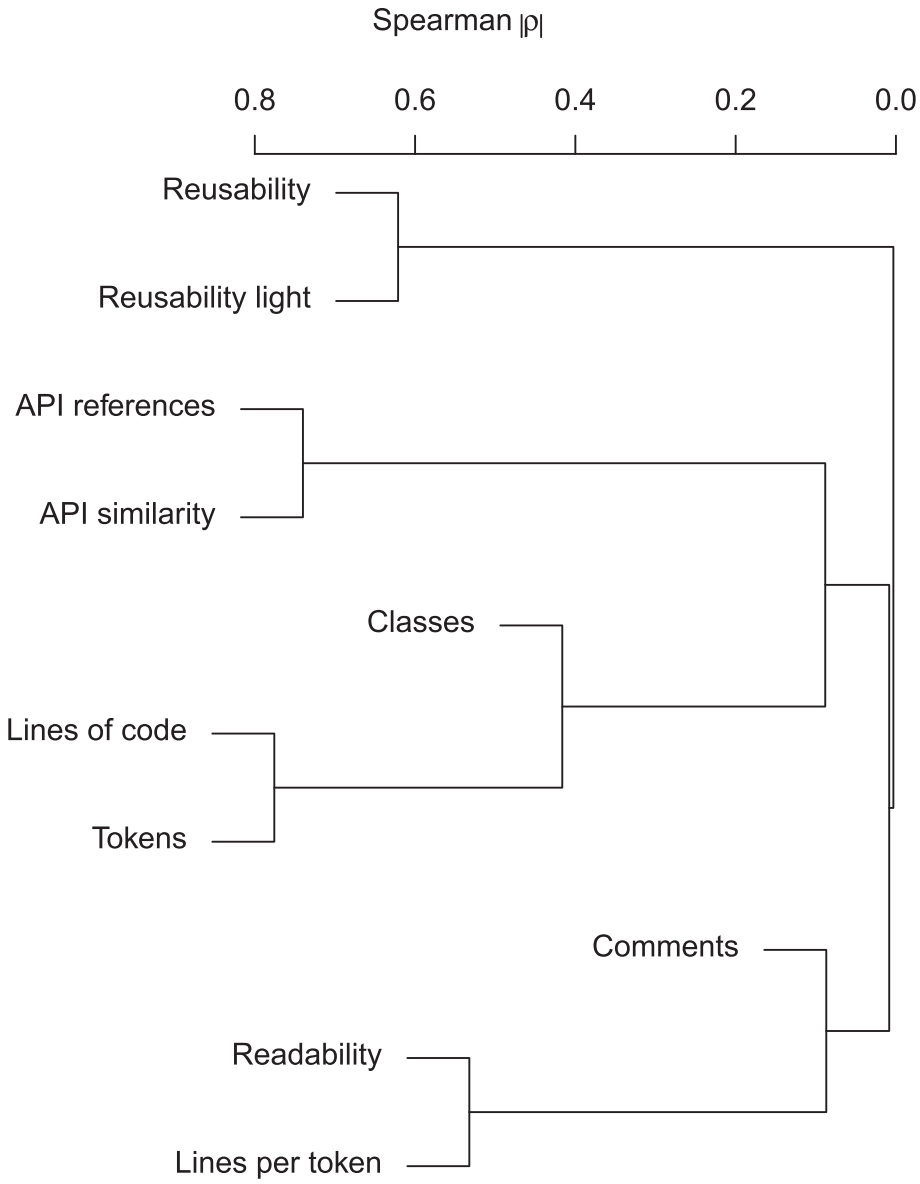


FIGURE 10 Correlation structure for all API examples. API, application programming interfaces

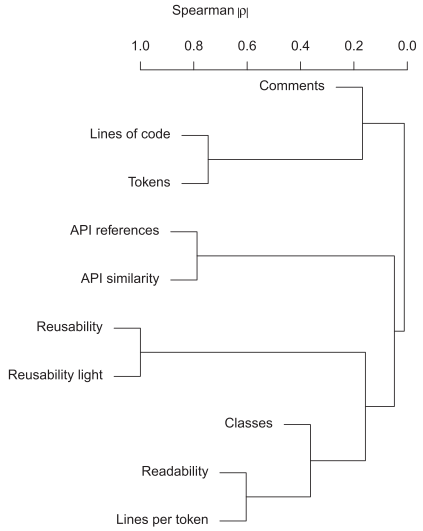
we obtain the final *score* of an API example ex_i by linearly combining its *API similarity* and *quality score*, as follows:

$$Score(ex_i) = 0.5 \times APISimilarity(ex_i) + 0.5 \times QualityScore(ex_i). \quad (3)$$

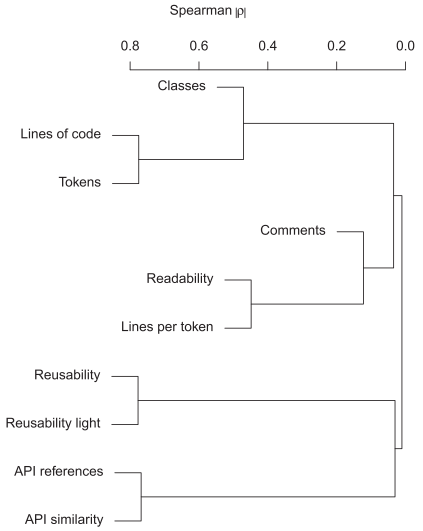
Score values close to 1 represent API examples with higher similarity and quality in terms of readability and reusability. By contrast, score values close to 0 indicate API examples with lower similarity and quality. **We rely on the *score metric* to rank the API examples**. In Section 7, we provide a detailed analysis of our ranking solution by comparing the quality of our API examples with the ones provided by popular programming websites.

5.2.3 | Examples

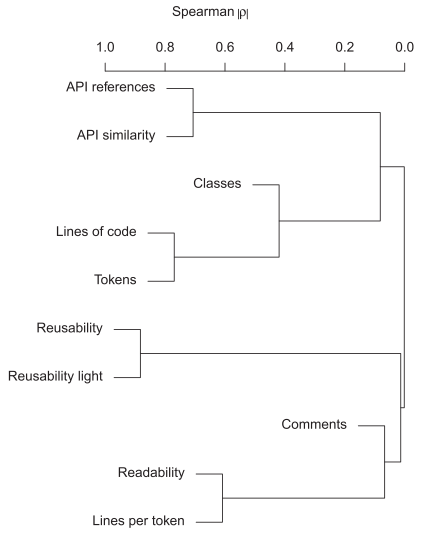
As illustration, Figure 12 presents top ranked (left side) and bottom ranked (right side) API usage examples for four APIs: (a) `IOUtils.copy`, (b) `BaseEncoding.base32`, (c) `Files.createTempDir`, and (d) `StringTokenizer.countTokens`. Overall, we notice that the top-ranked examples are smaller and more concise than the bottom-ranked ones. This occurs because smaller API examples are more similar to their target APIs, which is captured by the *API similarity* in the *score metric*. A deeper investigation about *readability* and *reusability* is presented in Section 7.



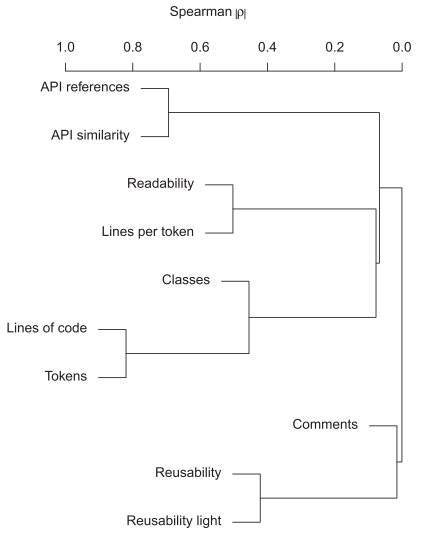
(A) Java Library API examples.



(B) Google Guava API examples.



(C) Apache Commons API examples.



(D) Spring Framework API examples.

FIGURE 11 Correlation structure per system

6 | PRESENTING API USAGE EXAMPLES

After collecting, filtering, and ranking the API examples, we start the process to present this data. Figure 13 illustrates the last step of our approach: presenting API examples. It receives as input a list of ranked API examples and creates HTML webpages to support their visualization and navigation.

6.1 | API example webpages

For each API, we generate a single webpage with its ranked examples. Therefore, we generate 1,537,068 webpages with at least one API example. Figure 14 presents a typical webpage containing API examples in APISonar; in this case, it refers to the API `org.joda.time.DateTime.now`.[8]The source code is highlighted for the Java syntax. The API usage, which

[8]http://apisonar.com/java-examples/org.joda.time.DateTime.now.html

Example 1

```
private static int copy(InputStream input, OutputStream output) {
        try {
            return IOUtils.copy(input, output);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

Example 25

```
private void copyResourceToPythonWorkDir(String srcResourceName,
                                         String dstFileName) throws IOException {
    FileOutputStream out = null;
    try {
      out = new FileOutputStream(pythonWorkDir.getAbsoluteFile() + "/" + dstFileName);
      IOUtils.copy(
          getClass().getClassLoader().getResourceAsStream(srcResourceName),
          out);
    } finally {
      if (out != null) {
        out.close();
      }
    }
  }
```

(A) API `IOUtils.copy` (http://apisonar.com/java-examples/org.apache.commons.io.IOUtils.copy.html).

Example 1

```
protected String getEncodedValue(String value, boolean omitPadding) {
        BaseEncoding encoding = BaseEncoding.base32();
        encoding = omitPadding ? encoding.omitPadding() : encoding;

        return encoding.encode(value.getBytes(StandardCharsets.UTF_8));
    }
```

Example 5

```
private String getBuiltProductsRelativeTargetOutputPath(TargetNode<?> targetNode) {
    if (targetNode.getDescription() instanceof AppleBinaryDescription
        || targetNode.getDescription() instanceof AppleTestDescription
        || (targetNode.getDescription() instanceof AppleBundleDescription
            && !isFrameworkBundle((AppleBundleDescriptionArg) targetNode.getConstructorArg()))) {
      // TODO(grp): These should be inside the path below. Right now, that causes issues with
      // bundle loader paths hardcoded in .xcconfig files that don't expect the full target path.
      // It also causes issues where Xcode doesn't know where to look for a final .app to run it.
      return ".";
    } else {
      return BaseEncoding.base32()
          .omitPadding()
          .encode(targetNode.getBuildTarget().getFullyQualifiedName().getBytes());
    }
  }
```

(B) API `BaseEncoding.base32` (http://apisonar.com/java-examples/com.google.common.io.BaseEncoding.base32.html).

Example 1

```
public static String createTempDir() {
    final File tmpDir = Files.createTempDir();
    tmpDir.deleteOnExit();
    return tmpDir.getAbsolutePath();
  }
```

Example 25

```
private static Properties loadConfig(int port, String zkConnect) {
        Properties properties = new Properties();
        properties.setProperty("port", String.valueOf(port));
        properties.setProperty("zookeeper.connect", zkConnect);
        properties.setProperty("broker.id", "0");
        properties.setProperty("log.dirs", Files.createTempDir().getAbsolutePath());
        properties.setProperty("delete.topic.enable", "true");
        properties.setProperty("offsets.topic.replication.factor", "1");
        properties.setProperty("group.initial.rebalance.delay.ms", "0");

        return properties;
    }
```

(C) API `Files.createTempDir` (http://apisonar.com/java-examples/com.google.common.io.Files.createTempDir.html).

Example 1

```
private String getLocalName(String qname) {
        StringTokenizer tok = new StringTokenizer(qname, ":");

        if (tok.countTokens() == 1) {
            return qname;
        }
        tok.nextToken();

        return tok.nextToken();
    }
```

Example 25

```
public static String[] stringToArray(String from, String separator) {
        if (from == null) {
            return null;
        }
        if (separator == null) {
            separator = " ";
        }
        StringTokenizer toks = new StringTokenizer(from, separator);
        String[] result = new String[toks.countTokens()];
        int i = 0;
        while (toks.hasMoreTokens()) {
            result[i++] = toks.nextToken().trim();
        }
        return result;
    }
```

(D) API `StringTokenizer.countTokens` (http://apisonar.com/java-examples/java.util.StringTokenizer.countTokens.html).

FIGURE 12 Top ranked (left side) and bottom ranked (right side) API usage examples in APISonar. API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

represents the key element of the solution, is also highlighted: the API class is presented in yellow, while the API method (or field) in green. Highlighting important elements of the solution is a good practice to present code examples.[11]

Below the API full name, we find two ways to navigate to other APIs: API hierarchy and API categories. In the former, one may move to the class or package level of the same API. For instance, in the case of the API `org.joda.time.DateTime.now`, we can directly move to the `org.joda.time.DateTime`, `org.joda.time`, `org.joda`, and `org` webpages. By exploring the API categories, one may find related APIs that belong to other libraries and frameworks. For instance, by clicking on the button *now APIs*, APISonar presents 126 APIs that include the string *now* in its full name. Also in Figure 14, we see that it is possible to search API examples by simply typing API names in the top right text input, which provides autocomplete suggestions. All webpages in APISonar are mobile responsive.

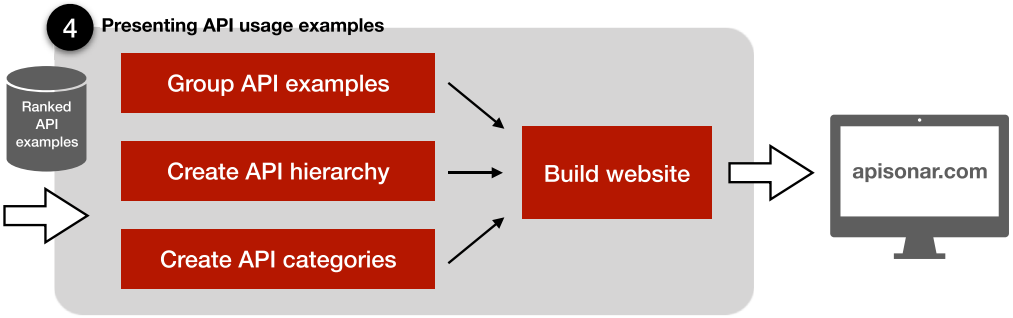


FIGURE 13 Presenting API usage examples (step 4). API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

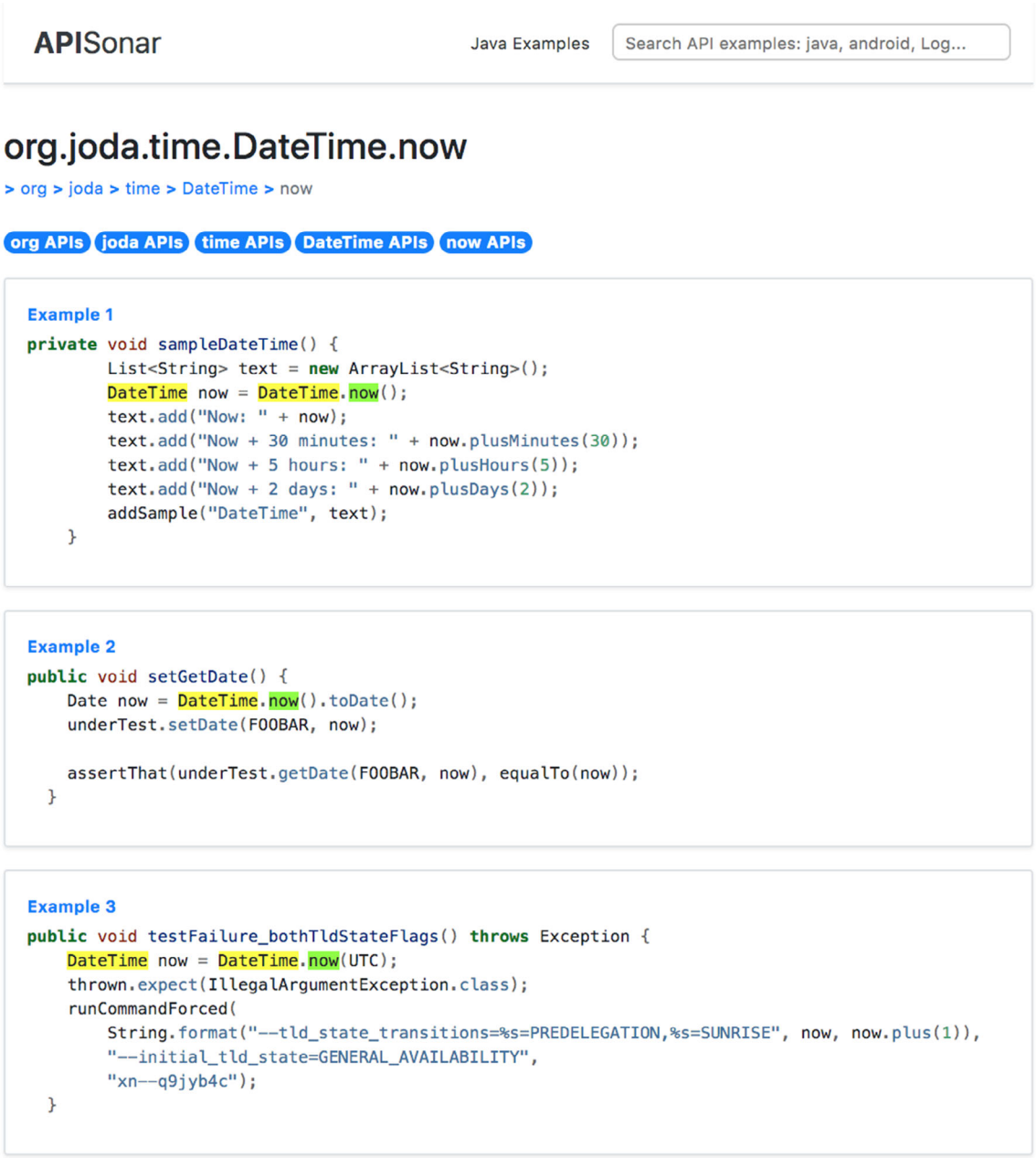


FIGURE 14 Typical API example webpage provided by APISonar. API, application programming interfaces [Color figure can be viewed at wileyonlinelibrary.com]

6.2 | API navigation and category webpages

In addition, to the webpages containing the API examples, we also generate auxiliary webpages to support navigation on API hierarchy and category. API hierarchy webpages are generated to group APIs in the same package, while API category webpages include APIs that are formed by the same substring. We export 675,730 API hierarchy and 817,849 API category webpages.

6.3 | APISonar use cases

APISonar hosts millions of API usage examples to help developers during their programming tasks. As stated in Section 2, there are several programming websites available nowadays for this goal. Many of them have high traffic and millions of pages indexed by the Google search engine, which is an indication of their popularity and importance. Due to our curated extraction, filtering, and ranking steps, to our knowledge, APISonar makes a competitive solution to the current programming websites. We foresee two use cases for APISonar:

- Finding out how to use an API: a developer may rely on APISonar to find API usage examples for a certain API to support its programming activities. That is, given the API name (i.e., class and method names), he can find usage examples for that API.
- Discovering related APIs: a developer may use APISonar to explore related APIs. For this purpose, he or she can easily access APIs in the same package (API hierarchy webpages) or jumping to APIs in other libraries and frameworks (API category webpages).

7 | EVALUATION 1: QUALITY AND COVERAGE

We perform three evaluations to assess the examples provided by APISonar. First, we analyze the quality of the API examples returned by APISonar as compared with the ones presented by popular programming websites (Section 7.1). Second, we compare the quality of the APISonar API examples, and the ones returned by API searches on Google (Section 7.2). Finally, we assess the coverage of the APIs present in APISonar (Section 7.3).

Notice that the websites selected for comparison in this evaluation (i.e., ProgramCreek, Codota, SearchCode, and Google) are code search engines; they are not API search/recommendation engines (see Section 10 for an overview of API search/recommendation tools). Indeed, as APISonar, those websites can be used to find code examples.

It is important to recall that our evaluation is performed with the original readability and reusability metrics, that is, not with *reusability light* and *lines per token*, which are only adopted in the ranking step of our approach.

7.1 | Quality assessment: APISonar vs. Programming websites

7.1.1 | Design

In our first evaluation, we compare the quality of the API examples provided by APISonar with API examples made available by three popular programming websites: SearchCode, ProgramCreek, and Codota. Our goal is to assess whether our API examples are as good as, better, or worse than the API examples provided by those websites. Ideally, code examples should be easy to understand and reuse,[11,24] thus, we measure quality in terms of readability and reusability, as presented in Section 5.2.1.

We select 100 APIs provided by four popular libraries and frameworks: Google Guava, Java Library, Apache Commons, and Spring Framework, which are presented in Table 2 with their respective class and method names. Then, we collect code examples about these APIs in APISonar and on the target websites (SearchCode, ProgramCreek, and Codota). Finally, for each API, we assess the readability and reusability of the top-ranked API examples in three scenarios, top 1, top 5, and top 10, as those are typical sizes that users would inspect.[18]

TABLE 2 Selected APIs from Google Guava, Java Library, Apache Commons, and Spring Framework

	API	
System	**Class name**	**Method name**
Google Guava	Preconditions	checkArgument, checkElementIndex, checkNotNull, checkPositionIndex, checkState
	Ints	asList, concat, saturatedCast, toArray, tryParse,
	Lists	asList, charactersOf, newArrayList, partition, reverse
	Sets	cartesianProduct, difference, intersection, powerSet, union
	Files	append, createTempDir, readLines, toByteArray, write
	BufferedReader	close, read, readLine, reset, skip
	File	exists, getAbsolutePath, isDirectory, listFiles, mkdirs
Java Library	ArrayList	add, contains, get, isEmpty, size
	HashMap	containsKey, containsValue, isEmpty, put, putAll
	Scanner	hasNextLine, match, nextLine, useDelimiter, useLocale,
	Base64	decode, decodeBase64, encodeBase64, encodeBase64URLSafeStrin,g, isArrayByteBase64
	CSVFormat	parse, withCommentMarker, withDelimiter, withHeader, withQuote
Apache Commons	FilenameUtils	concat, getExtension, isExtension, normalize, separatorsToUnix
	IOUtils	closeQuietly, copy, readLines, toByteArray, toInputStream
	StringUtils	isBlank, join, split, substringBefore, trimToNull
	SpringApplication	addListeners, exit, run, setAdditionalProfiles, setBannerModemidrule
	ApplicationContext	containsBean, getAutowireCapableBeanFactory, getBean, getBeanDefinitionNames, publishEventmidrule
Spring Framework	BindingResult	getFieldErrors, getGlobalErrors, hasErrors, hasFieldErrors, rejectmidrule
	RestTemplate	exchange, getForEntity, getForObject, postForObject, setMessageConvertersmidrule
	ModelAndView	addAllObjects, addObject, getModel, hasView, setViewName

Abbreviation: API, application programming interfaces.

We analyze the statistical significance of the difference between the comparisons by applying the Mann-Whitney test at p-value = .05. To show the effect size of the difference between them, we compute Cliff's Delta (or d). We interpret the effect size values as negligible for $d < 0.147$, small for $d < 0.33$, medium for $d < 0.474$, and large otherwise.[33]

7.1.2 | Results

Figure 15 presents the readability of the top 1, top 5, and top 10 returned API examples, for each programming website. We notice that APISonar provides the most readable API examples in the three scenarios, followed by Codota, ProgramCreek, and SearchCode. When considering the first returned API example (top 1), APISonar has a readability of 0.49, while Codota has 0.11, on the median. For the top 5 and top 10, APISonar also has better results than Codota (top 5: 0.351 vs. 0.14; top 10: 0.322 vs. 0.148). We analyze the statistical significance of the difference between APISonar and Codota examples: in the three comparisons, the difference is statistically significant (p-value < .001), with *small* and *medium* effect.

Figure 16 presents the reusability analysis. Overall, we notice that the API examples provided by APISonar are the most reusable. As in the previous analysis, APISonar is followed by Codota, ProgrammigCreek, and SearchCode. When considering the top 1, APISonar and Codota API examples have the highest reusability (i.e., 1), on the median. However, APISonar presents higher reusability than Codota in the other two scenarios (top 5: 0.971 vs. 0.842; top 10: 0.966 vs. 0.827). In both cases, the difference is statistically significant (p-value < .001), with *large* effect.

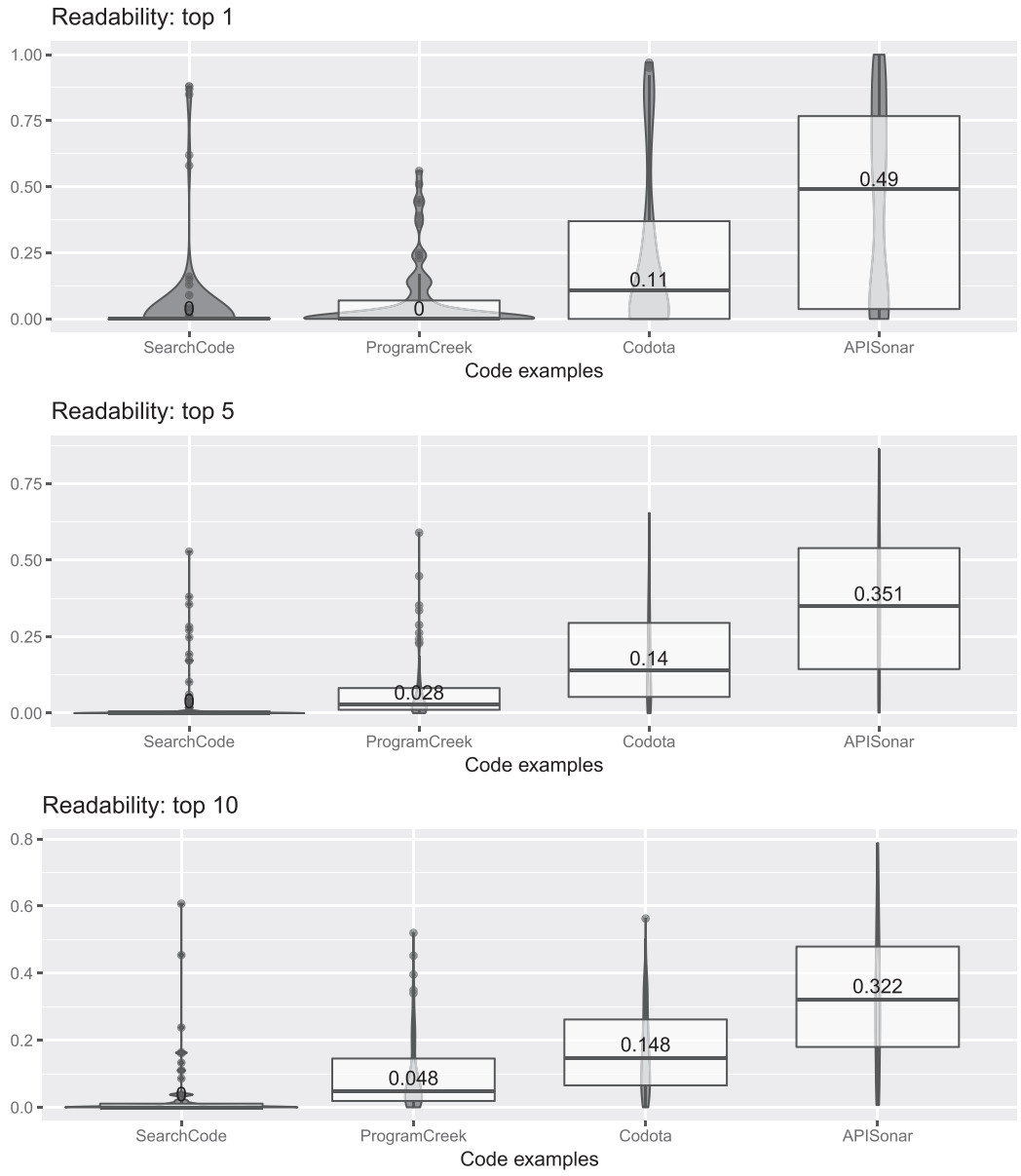


FIGURE 15 Readability: APISonar vs. API examples provided by Programming websites. API, application programming interfaces

Summary: API examples provided by APISonar are more readable and reusable than the ones made available by the popular programming websites SearchCode, ProgramCreek, and Codota.

7.2 | Quality assessment: APISonar vs. Google search

7.2.1 | Design

In this second evaluation, we go a step further in our quality assessment. Instead of directly comparing APISonar to the results provided the programming websites themselves (as in the previous analysis), we compare the API examples returned by Google search. Again, we focus on the APIs provided by Google Guava, Java Library, Apache Commons, and Spring Framework. We collect API examples about these APIs by manually inspecting the first Google search results.

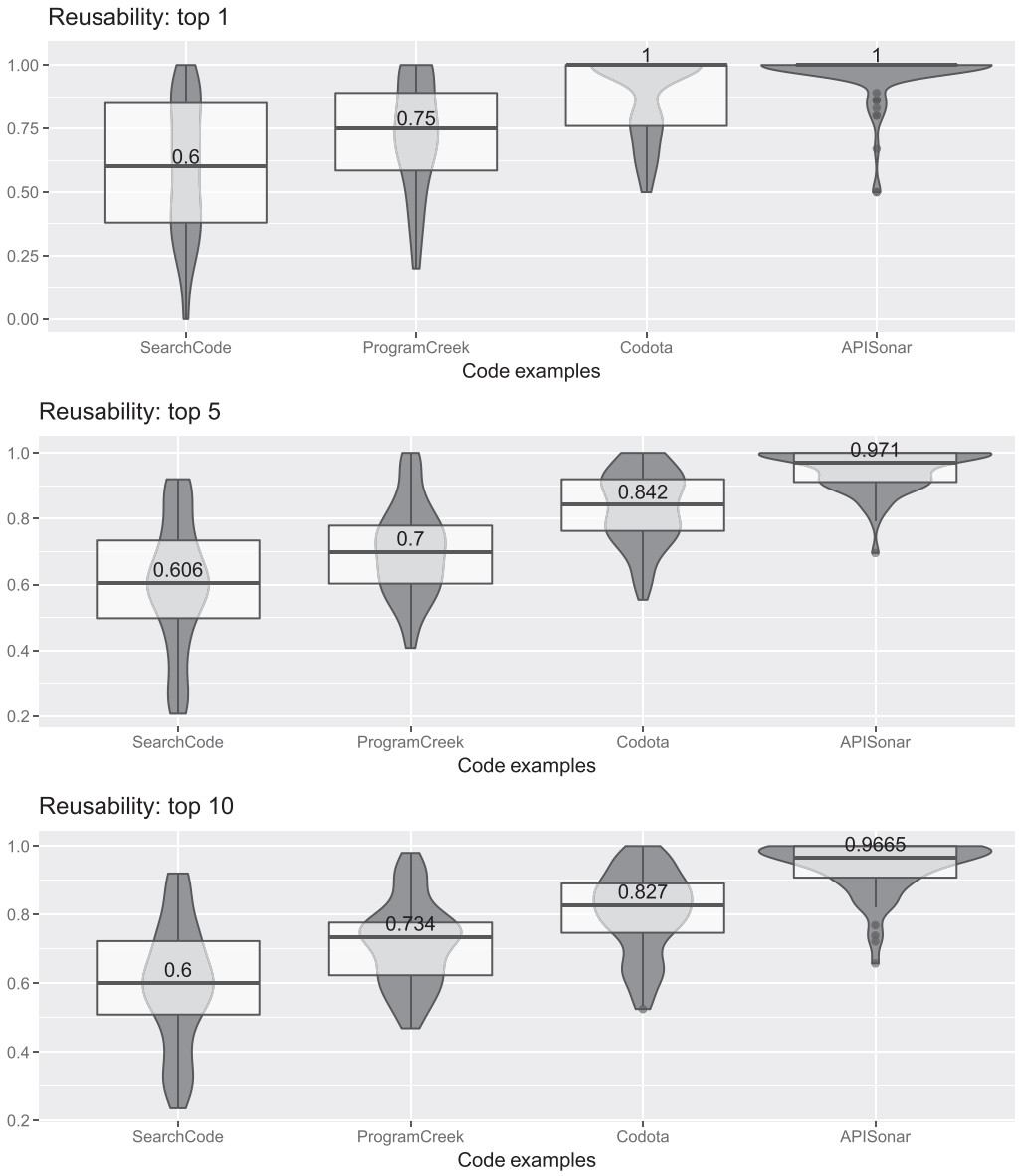


FIGURE 16 Reusability: APISonar vs. API examples provided by Programming websites. API, application programming interfaces

Specifically, we look for code examples on the web by querying on Google the API full name followed by the keyword "*example*." For instance, for the Java API `BufferedReader.close`, we query for *"java.io.BufferedReader.close example*." For each API, we manually collect six code examples, from which three are didactic and three are real software, thus, totalizing 600 code examples (300 didactic and 300 real software). These code examples come from popular programming websites, such as Stack Overflow, W3Schools, Baeldung, Tutorials Point, GeeksforGeeks, among many others, which are often top-ranked by the Google search engine.[19] As in the previous analysis, we rely on readability and reusability to assess the quality of the examples returned by Google.

In APISonar, the examples are selected as follows. First, we randomly select 10K API examples about Google Guava, Java Library, Apache Commons, and Spring Framework (2.5K per system). Then, from this balanced dataset, we randomly collect 300 API examples and compute their readability and reusability. Finally, we collect the top 300 most readable and the top 300 most reusable API examples to assess the best examples provided by APISonar.

7.2.2 | Results

Figure 17 presents the readability of the API examples returned by Google search (first three box plots) and APISonar (last two box plots). The real software and didactic code examples returned by Google search have low readability, 0.01 and

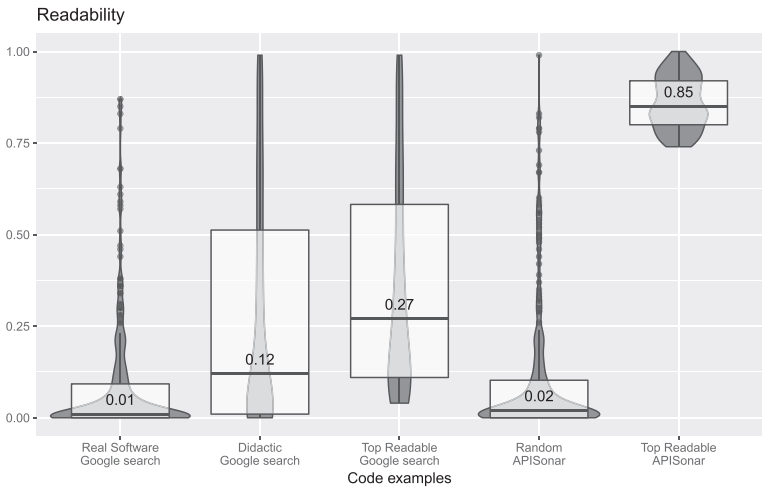


FIGURE 17 Readability: APISonar vs. API examples returned by Google search. API, application programming interfaces

0.12, respectively. However, not all API examples returned by Google have low readability: the top readable API examples achieve readability of 0.27. In APISonar, we have two distinct results. First, when randomly selecting API examples in our dataset, we find readability of 0.02; interestingly, this value is very close to the readability of the real software examples returned by Google (0.01). This low value is expected and this is the major reason we include the ranking step in the proposed approach. Thus, if they appear on our website, these API examples are likely to be lower ranked. The last box plot presents the top readable API examples in APISonar. In this case, our approach achieves readability of 0.85: the difference is statistically significant (p-value $< .001$), with *large* effect. This shows that API examples with high readability can be mined from code repositories. However, if randomly selected from code repositories, they may have low readability.

Figure 18 presents the same analysis for the reusability metric. Notice that the didactic and the top reusable API examples returned by Google and APISonar have the highest reusability (value: 1). API examples from real software and randomly selected in our dataset have lower reusability (0.8 and 0.67).

Summary: Overall, API examples provided by APISonar are more readable than the ones returned in Google search results. Both APISonar and Google results present the highest level of reusability.

7.3 | Coverage assessment

To assess the coverage of APIs provided by APISonar, we perform two analyses. We first compare the APIs detected by our approach with the ones detected at the ultra-large-scale software repository Boa.[27] Second, we compare our APIs with the ones provided by ProgramCreek, which is the sole programming website that makes their APIs publicly available.

7.3.1 | Design

Boa is a language and infrastructure that eases mining software repositories at an ultra-large-scale level.[27] We rely on Boa to discover APIs consumed by thousands of Java projects. Boa provides an online platform in which one may perform queries to assess their datasets. For this purpose, we select the dataset *2019 October/GitHub (medium)*, which includes 783,982 projects.[9] From those projects, we find that 203,499 use the Java programming language. We implemented and run a query to find all the APIs imported by these Java projects.[10] Then, we filter out the APIs imported via wildcard (i.e., `*`) to find the exact APIs consumed by the 203,499 Java projects. Finally, we compare those APIs with the ones provided by APISonar.

[9] http://boa.cs.iastate.edu/stats/index.php
[10] Query available at http://boa.cs.iastate.edu/boa/?q=boa/job/public/86090

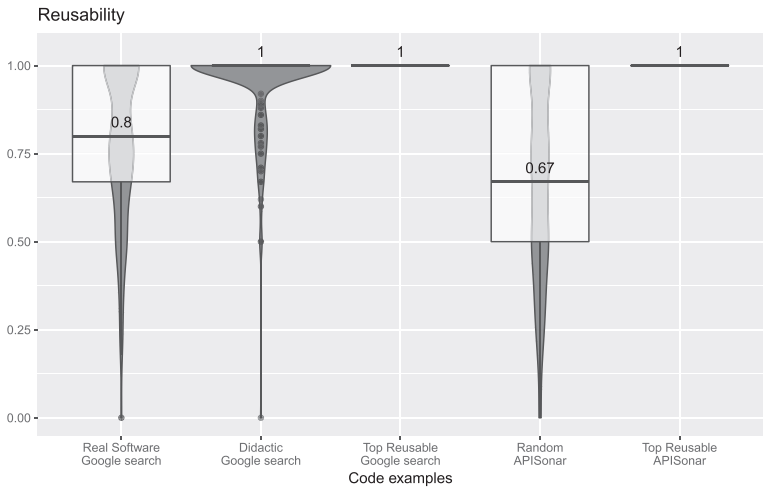


FIGURE 18 Reusability: APISonar vs. API examples returned by Google search. API, application programming interfaces

TABLE 3 APIs provided by Boa and APISonar (at class level)

APIs	Boa	APISonar
all	654,536	464,757
java.*	1496	1514
android.*	3075	4460
org.apache.*	32,927	67,665
org.springframework.*	6079	8229

Abbreviation: API, application programming interfaces.

In the second analysis, we assess the APIs provided by ProgramCreek, which is a popular programming website that developers rely on to find API examples in Java. For this purpose, we focus on the APIs suggested by ProgramCreek itself on its search webpage: [11] spark, google, hadoop, utils, android, FileChannel, jdbc, springframework, gson, reflect, graphics, kafka, sql, jsoup, wifi, bluetooth, swing, lucene, and hbase. Finally, we compare those APIs with the ones provided by APISonar.

It is important to recall that, in both analyses (i.e., Boa and ProgramCreek), the comparisons are performed at the level of API classes, that is, not at the level of API methods nor fields. This is done because the data returned by both Boa and ProgramCreek are at the class level.

7.3.2 | Results

APIs provided by Boa and APISonar. As presented in Table 3, 654,536 API classes are consumed by the Java projects in the Boa dataset. In APISonar, there are examples of 464,757 API classes. Boa has therefore 189,779 more APIs than APISonar. This difference is expected: the Boa dataset has 203,499 Java projects, while APISonar has only 4486. To focus on relevant APIs, we assess the APIs provided by Java, Android, Apache, and Spring Framework. In this case, APISonar provides more APIs than Boa. For instance, APISonar includes 4460 Android APIs, while Boa 3075. This may be explained the fact we mine project history, thus, more APIs are returned for those libraries and frameworks.

Table 4 presents set operations over the Boa and APISonar APIs. When considering all APIs, there is an intersection of only 57,780 (5.4%); 596,756 (56.2%) APIs are specific to Boa, while 406,977 (38.3%) are specific to APISonar. However, when considering the APIs provided by Java, Android, Apache, and Spring Framework, we notice that more APIs only happen in APISonar. For instance, in Android, 1704 (29.2%) APIs happen in both, 1371 (23.5%) are specific to Boa, and 2756 (47.3%) are specific to APISonar. In Apache, the differences are even larger: 22,274 (24.8%) are Boa specific, while 57,012 (63.4%) are APISonar specific. Thus, we can infer that adding more projects to the analysis will not necessarily

[11] Main search page of ProgramCreek: https://www.programcreek.com/java-api-examples/index.php

TABLE 4 Set operations of the APIs provided by Boa and APISonar (at class level)

	Union	Intersection		Boa specific		APISonar specific	
API	Boa ∪ APISonar	Boa ∩ APISonar	%	Boa–APISonar	%	APISonar–Boa	%
all	1,061,513	57,780	5.4	596,756	56.2	406,977	38.3
java.*	1792	1218	68.0	278	15.5	296	16.5
android.*	5831	1704	29.2	1371	23.5	2756	47.3
org.apache.*	89,939	10,653	11.8	22,274	24.8	57,012	63.4
org.springframework.*	11,789	2519	21.4	3560	30.2	5710	48.4

Abbreviation: API, application programming interfaces.

TABLE 5 APIs provided by ProgramCreek and APISonar (at class level)

APIs	ProgramCreek	APISonar	APIs	ProgramCreek	APISonar
spark	31	804	*graphics*	173	588
google	1160	13,980	*kafka*	101	1862
hadoop	443	11,424	*sql*	160	7416
utils	338	8666	*jsoup*	20	69
android	1914	16,834	*wifi*	18	169
FileChannel	4	21	*bluetooth*	26	211
jdbc	76	1642	*swing*	371	887
springframework	1119	8236	*lucene*	133	2164
gson	47	190	*hbase*	103	3367
reflect	88	976	all	6325	79,506

Abbreviation: API, application programming interfaces.

bring more relevant APIs. However, adding older project versions to the analysis—as performed by APISonar—may bring more relevant APIs. Indeed, at some point, having more projects will only bring noise to the analysis with toy, irrelevant, and inactive projects, as previously reported by the code mining literature.[34]

Summary: APISonar covers most APIs provided by relevant projects such as Java, Android, Apache, and Spring. This is explained by the fact APISonar assesses not only current project versions, but also older versions.

APIs provided by ProgramCreek and APISonar. In this second coverage analysis, we compare the APIs provided by APISonar against the ones made available by ProgramCreek, as presented in Table 5. Overall, APISonar provides many more APIs about the selected API patterns than ProgramCreek. For instance, for the pattern *google*, ProgramCreek has 1160 APIs and APISonar has 13,980. For all patterns, ProgramCreek provides 6325 APIs, while APISonar 79,506.

Summary: Considering the 19 API patterns, the difference is significant between the amount of APIs provided by APISonar and ProgramCreek: APISonar provides 12× more APIs than ProgramCreek.

8 | EVALUATION 2: AUDIENCE, ENGAGEMENT, INDEXING, AND SEARCH QUERIES

In this second evaluation, we focus on the usage of APISonar. To assess the relevance of APISonar, we analyze the audience and engagement of the users (Section 8.1), the indexing of our webpages by the Google search engine (Section 8.2), and the search queries users perform on Google find our website (Section 8.3).

8.1 | Audience and engagement

8.1.1 | Design

We assess the traffic logs generated by the users of our website when performing code search activities during the first 5 months the website is alive, from January 2020 to May 2020. The logs are tracked and analyzed with the support of Google Analytics, a tool that helps understanding how people use websites.[35,36] We focus on better understanding our audience and its engagement by assessing the number of users, the countries they come, and how long they navigate in our platform.

8.1.2 | Results

APISonar received 3744 users, which generated 4093 sessions and 7455 pageviews. From the 3744 users, 211 (5%) are returning visitors; from the 4093 sessions, 3740 (91%) are generated by new visitors, while 353 (9%) are generated by returning visitors. Figure 19 shows the world map colored by the frequency of users per country. As we can notice, our users came from all continents: Europe (36%), Asia (34%), Americas (25%), Africa (2%), and Oceania (1%). Specifically, the users came from 119 distinct countries and 1193 cities.

Table 6 details the top five countries with most users. The United States brought most users, 14.83%, followed by India (10.09%), South Korea (5.65%), Germany (4.37%), and Japan (4.26%); these five countries together represented 1472 (39%) users. The top 10 countries provided 2052 (54%) users, while the top 50 brought the majority of the traffic: 3481 (92%).

Next, we assess the engagement of our users. As suggested by Google Analytics,[36] we measure engagement in terms of session duration. The average amount of time users spent viewing a specific page in APISonar is 40 s. Table 7 breaks this analysis per session duration: 84% of the sessions have between 0 and 10 s and generated 49% of the pageviews. In addition, 16% of the sessions have over 10 s and generated 51% of the pageviews, which represents 647 sessions and 3788 pageviews.

Table 8 presents the pages in APISonar with the highest average time. The API `FragmentTransaction.commit` has the highest average time, that is, 28 min and 28 s. In second, we have the API `MoneyConverters.asWords` (28 min and 14 s) and third one is `Cache.getAdvancedCache` (25 min and 5 s). On average, the top 10 webpages have 23

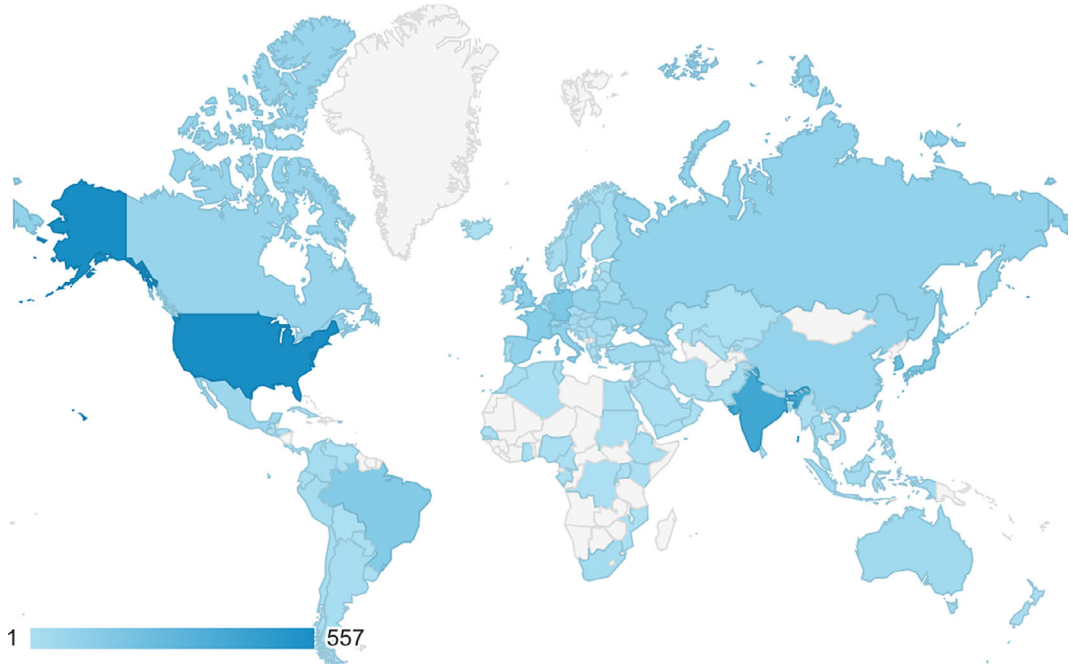


FIGURE 19 Users by country [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 6 Frequency of users by country

	Users	
Country	**#**	**%**
United States	557	14.83
India	379	10.09
South Korea	212	5.65
Germany	164	4.37
Japan	160	4.26
Top 5	1472	39.0
Top 10	2052	54.0
Top 50	3481	92.0

TABLE 7 Engagement of users

	Sessions		Pageviews	
Session Duration	**#**	**%**	**#**	**%**
0–10 s	3446	84	3667	49
11–30 s	196	5	707	9
31–60 s	133	3	585	8
61–180 s	143	3	1011	14
181–600 s	111	3	935	13
601–1800 s	59	1	422	6
1801+ s	5	1	128	2
Total	4093	100	7455	100

TABLE 8 Pages with highest average time

Page	Avg. time on page
http://apisonar.com/java-examples/androidx.fragment.app.FragmentTransaction.commit.html	28 min 28 s
http://apisonar.com/java-examples/pl.allegro.finance.tradukisto.MoneyConverters.asWords.html	28 min 14 s
http://apisonar.com/java-examples/org.infinispan.Cache.getAdvancedCache.html	25 min 5 s
http://apisonar.com/java-examples/java.nio.file.Files.newInputStream.html	24 min 28 s
http://apisonar.com/java-examples/org.powermock.api.mockito.PowerMockito.when.html	23 min 38 s
http://apisonar.com/java-examples/java.net.URL.getPath.html	22 min 54 s
http://apisonar.com/java-examples/android.util.proto.ProtoOutputStream.write.html	20 min 39 s
http://apisonar.com/java-examples/com.google.common.logging.Level.INFO.html	19 min 45 s
http://apisonar.com/java-examples/org.apache.commons.ssl.html	19 min 25 s
http://apisonar.com/java-examples/android.support.v4.content.ContextCompat.getDrawable.html	18 min 11 s
Top 10	23 min 4 s
Top 25	18 min 33 s
Top 50	13 min 40 s
Top 100	9 min 5 s
Top 250	4 min 36 s

min and 4 s, the top 100 have 9 min and 5 s, and the top 250 have close to 4 min and a half. This shows that many specific APIs may grab large attention so that users remain several minutes on the same webpage.

To conclude this evaluation, we present in Figure 20 the behavior flows for three APIs: `StringUtils.leftPad`, `Mockito.doThrow`, and `apache.commons`. Behavior flows show how the users navigate on our website, that is, the flow from one page to another page. Figure 20 (top) presents the flow for the Apache Commons API `StringUtils.leftPad`. It shows that from the five sessions started on this webpage, two sessions went to the related API Apache Commons `StringUtils.rightPad`, while 1 went to the Google Common `Strings.padEnd`.

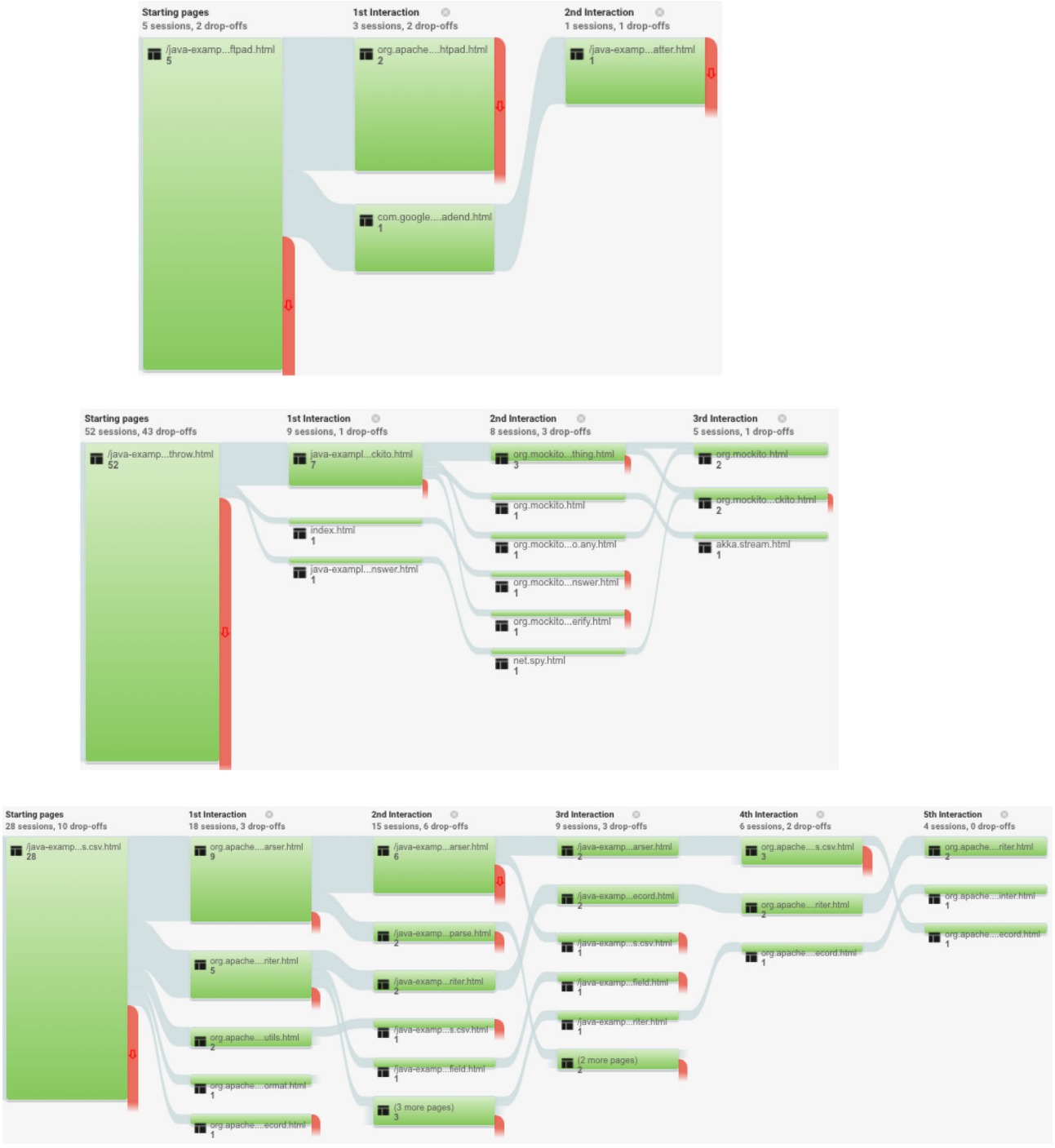


FIGURE 20 Behavior flows of StringUtils.leftPad (top), Mockito.doThrow (middle), and apache.commons (bottom) [Color figure can be viewed at wileyonlinelibrary.com]

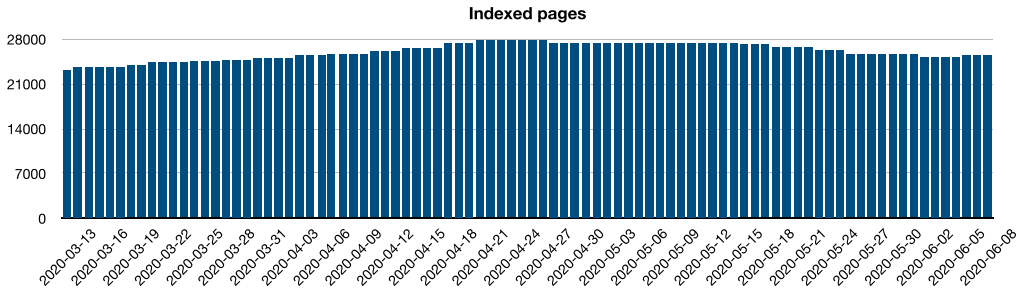


FIGURE 21 Indexing of APISonar pages over time [Color figure can be viewed at wileyonlinelibrary.com]

Figure 20(middle) presents the flow for the Mockito API `Mockito.doThrow`. We see that the navigation flow is diverse: from the 52 sessions, 7 went to the API class `Mockito`, and then the flow expanded to 6 distinct Mockito APIs, such as `Mockito.doNothing`, `Mockito.any`, and `Mockito.doAnswer`, showing that the users are indeed exploring related APIs. Finally, in Figure 20(bottom), we see a more generic flow about all the APIs under `apache.commons`. In this case, we notice that the flow is long: we present only until the fifth interaction level, however, this chart goes until the 12th interaction level, suggesting that those users deeply explored APISonar.

Summary: Despite being a novel website, APISonar has attracted thousands of real users. It received 3.7K users from 119 countries and 1193 cities for 5 months. Users remain on average 40 s viewing a specific page; some visits may long several minutes, For example, the top 10 pages have average 23 min, while the top 250 pages 4 min and a half. The behavior flows show that the users may explore related APIs (this analysis should be expanded for a better comprehension of navigation).

8.2 | Indexing of webpages

8.2.1 | Design

Google suggests that websites should focus on both user experience and performance in search results, as stated by its guidelines: "*You should build a website to benefit your users, and any optimization should be geared toward making the user experience better. One of those users is a search engine, which helps other users discover your content*."[12]That is, in addition to being good for users, webpages should be good for the search engine so that they can appear in Google search results.

This way, we aim to make APISonar examples indexed by the Google search engine, however, this is not a trivial task. The search engine has many quality restrictions:[37] "*The indexing of your content by Google is determined by system algorithms that take into account user demand and quality checks*."[38] To have the webpages indexed, we followed good practices suggested by Google, for instance, we provided useful content, we submitted the pages via a sitemap, we made the webpages mobile responsive, we performed mobile-friendly tests, and we created human intelligible URLs.[38] This information can be monitored with the support of the Google Search Console.[39]

8.2.2 | Results

According to the Google Search Console,[39] APISonar has nowadays[13]25K webpages that are indexed by the Google search engine, hence, they can appear in the search results. Figure 21 presents the indexing status of APISonar over the last 3 months (from March 2020 to June 2020). During this period, the number of indexed webpages varied from around

[12]https://support.google.com/webmasters/answer/7451184?hl=en
[13]June, 2020.

TABLE 9 Top ranked pages

Page	Avg. rank pos.
http://apisonar.com/java-examples/com.streamsets.pipeline.api.impl.Utils.checkNotNull.html	2.6
http://apisonar.com/java-examples/jdk.internal.math.FloatingDecimal.getBinaryToASCIIConverter.html	3.4
http://apisonar.com/java-examples/com.intellij.lang.javascript.html	3.5
http://apisonar.com/java-examples/org.m4m.android.VideoFormatAndroid.setVideoBitRateInKBytes.html	4
http://apisonar.com/java-examples/org.apache.camel.component.mock.MockEndpoint.getReceivedExchanges.html	4.3
http://apisonar.com/java-examples/org.neo4j.graphalgo.html	4.7
http://apisonar.com/java-examples/com.ibm.websphere.simplicity.log.Log.exiting.html	4.8
http://apisonar.com/java-examples/org.simplejavamail.api.email.html	5
http://apisonar.com/java-examples/org.apache.wss4j.policy.html	5
http://apisonar.com/java-examples/java.util.function.Predicate.test.html	5.3
Top 10	4.3
Top 25	5.3
Top 50	6.3
Top 100	7.6
Top 250	9.6

24K to 28K. This shows that APISonar has passed the many quality restrictions of Google[37] (e.g., quality, responsibility, intelligibility, and so forth) and provides relevant and useful content to the search engine.

Table 9 presents the top 10 best ranked pages of APISonar, also according to the Google Search Console.[39] Their ranking positions vary from 2.6 to 5.3; the average is 4.3. The best ranked page refers to the API `Utils.checkNotNull` provided by StreamSets, while the second one refers to `FloatingDecimal.getBinaryToASCIIConverter`, an internal API of the JDK. Other APIs are provided by IntelliJ, Android, Apache Camel, Neo4J, Ibm Websphere, Simple Java Mail, Apache WSS4J, and JDK. Besides, Table 9 also presents the average rank positions for the top 25, 50, 100, and 250 best ranked webpages, which is 5.3, 6.3, 7.6, and 9.6, respectively. That is to say, the top 250 best ranked webpages are likely to appear on the first page of Google search results, which shows 10 results.

Summary: APISonar has 25K webpages indexed by Google, suggesting that it provides relevant and useful content to the search engine. The top 250 best ranked webpages are likely to appear on the first page of Google search results (i.e., their average rank position is smaller than 10).

8.3 | Search queries

8.3.1 | Design

In this evaluation, we assess the search queries performed by the users on Google that led to APISonar webpages. Our goal is to analyze which queries developers perform on the web to find code examples so we can reason whether APISonar is a suited solution. As in the previous evaluation, this data can be tracked and obtained with Google Search Console.[39] Particularly, we analyze 100 queries that developers actually searched for on Google and clicked on the search results linking to APISonar webpages. Finally, after collecting these queries, we manually classify them to better understand their intention.

8.3.2 | Results

Table 10 presents the 100 analyzed search queries, which shows three types of queries: API query, generic query, and error query. We also present, for each query, the number clicks (i.e., the number of times a link to APISonar was clicked in search results), the number of impressions on Google (i.e., the number of times a link to APISonar appears in search results), and the CTR (i.e., clickthrough rate: clicks/impressions). The majority of the queries are about API (87%), followed by generic (7%), and error (6%). This is not surprising since developers are commonly interested in API usage examples, that is, code examples about how to use APIs.[8-10] This way, we provide evidences of such practice in context of a web search engine, which is the primarily tool adopted nowadays to find code examples.[5]

As presented in Table 10, the API queries are typically formed by the full or short API names. For example, the first query "*verifynointeractions*" refers to the API `Mockito.verifyNoInteractions` and led to http://apisonar.com/java-examples/org.mockito.Mockito.verifyNoInteractions.html. The second query, "*mono.error example*," refers to http://apisonar.com/java-examples/reactor.core.publisher.Mono.error.html. Those are examples in which the developers typed short names: the method name in the former, whereas the class and method names in the latter. By contrast, in the query "*org.springframework.data.geo.point*," the full API name is typed, as it refers to http://apisonar.com/java-examples/org.springframework.data.geo.Point.Point.html.

Summary: Google is the *de facto* solution to find code examples on the web nowadays. As APISonar is indexed by the Google search engine, it is likely to appear in search results. We find that the majority (87%) of the search queries actually performed by developers on Google to APISonar webpages are related to APIs. This result is in agreement with previous findings[8-10] in the sense that API queries are common in practice. This way, APISonar can be used (and it is already being used) as a feasible solution to find API examples on the web.

9 | THREATS TO VALIDITY

Code quality. We rely on two code metrics to assess quality: readability and reusability. We recognize that many other code metrics are available, for instance, to measure security, performance, complexity, and so forth. However, readability and reusability are two aspects often present in good code examples, that is, they should be easy to understand and reuse.[11] Moreover, even if other measures could be adopted to assess quality, we are aligned to the related literature, which often rely on readability and reusability to find the best code examples.[12,24,30]

Reusability light and lines per token. When raking the API examples, we adopt the metrics *reusability light* and *lines per token* as proxies of reusability and readability, respectively, because they are moderately and strongly correlated. Moreover, *reusability light* and *lines per token* have the advantage of being language independent, while the original reusability and readability are language dependent. Another factor that ensures *reusability light* and *lines per token* as good proxies is the results presented in our evaluation (Section 7). In both quality comparisons (APISonar vs. programming websites and APISonar vs. Google search), the API examples provided by APISonar have higher values of readability and reusability—notice that our evaluation is performed with the original readability and reusability metrics, that is, not *reusability light* and *lines per token*, which are only adopted in the ranking step of our approach.

Ranking score. Our ranking score aggregates three metrics: API similarity, readability, and reusability. The similarity between documents is often adopted in information retrieval and in the code search literature.[12-14,18] Naturally, developers may expect to see references to the target APIs in the API examples. Thus, the rationale of using API similarity is to better rank API examples with high similarity to their APIs. Nevertheless, we recognize that other metrics can be combined to form the score. We plan to explore that in further future work.

Past project versions. When collecting software projects from GitHub, we get their current as well as past versions in case they have more than 2000 commits (see Section 4). This is performed to assess not only present but also older/legacy APIs and API examples. Our coverage evaluation presented in Section 7 shows that even with much fewer projects than Boa, APISonar covers a large amount of relevant APIs. Notice, however, that the threshold of commits may vary if more or less APIs are needed.

TABLE 10 Search queries

Search Queries	Clicks	Impressions	CTR (%)	Search Queries	Clicks	Impressions	CTR (%)
verifynointeractions	6	67	8.96	getwindow android	1	24	4.17
mono.error example	4	72	5.56	exchange.getproperty	1	23	4.35
failed to init reactor's debug agent	3	40	7.5	resultset.gettimestamp	1	23	4.35
androidx.appcompat.app.alertdialog example	3	16	18.75	keycloaksession	1	22	4.55
exception on "get /sockjs-node" in filter "welcome_screen":	2	105	1.9	context.obtainstyledattributes	1	21	4.76
sendbroadcastasuser	2	93	2.15	textview setmovementmethod	1	20	5
exception on "get /sockjs-node" in filter "welcome_screen"	2	60	3.33	getcachedir android example	1	20	5
androidx.appcompat.app.alertdialog	2	55	3.64	objectutils.isempty	1	19	5.26
camel setproperty	2	43	4.65	boofcv	1	19	5.26
sc_unauthorized	2	42	4.76	mockserverhttprequest example	1	18	5.56
androidx.fragment.app.fragmentmanager	2	41	4.88	resultset.getarray example java	1	16	6.25
getoppackagename	2	37	5.41	exchange.setproperty in camel	1	16	6.25
"can't configure anyrequest after itself"	2	33	6.06	axmlprinter	1	16	6.25
org.elasticsearch.action.admin.cluster.repositories.c	2	29	6.9	wizzardo-http	1	16	6.25
request.getmethod()	2	28	7.14	android getsystemservice example	1	14	7.14
xxmassdeveloper	2	28	7.14	servicemanager.getservice	1	14	7.14
encodebase64string	2	22	9.09	fragmentmanager.begintransaction()	1	14	7.14
mergedannotations$searchstrategy	2	17	11.76	getsystemservice(context.location_service)	1	13	7.69
org.springframework.data.geo.point	2	15	13.33	getqueryparameter	1	13	7.69
apache hudi example	2	13	15.38	android createscaledbitmap	1	13	7.69
ehviewer	1	356	0.28	org.keycloak.models.keycloaksession	1	13	7.69
random nextint	1	144	0.69	sensor_service	1	11	9.09
mockito dothrow	1	124	0.81	androidx.core.content.fileprovider example	1	11	9.09
mockito argthat	1	118	0.85	mockito docallrealmethod example	1	11	9.09
getparcelableextra	1	110	0.91	firechannelread	1	11	9.09
jelly_bean_mr1	1	103	0.97	com.google.android.material.snackbar.snackbar	1	10	10
rand.nextint	1	81	1.23	uri getfragment	1	10	10
hudi cli	1	68	1.47	com.google.android.gms.vision.detector	1	10	10

(Continues)

TABLE 10 (Continued)

Search Queries	Clicks	Impressions	CTR (%)	Search Queries	Clicks	Impressions	CTR (%)
"property 'sonar.jacoco.reportpath' is no longer supported. use jacoco's xml report and sonar-jacoco plugin."	1	68	1.47	docallrealmethod mockito	1	10	10
getexternalstoragestate	1	67	1.49	fileutils.deletequietly	1	10	10
checkcallingorselfpermission	1	67	1.49	request.getservername() example	1	9	11.11
docallrealmethod	1	63	1.59	build.version_codes.kitkat	1	9	11.11
flux.fromiterable	1	61	1.64	androidx.lifecycle.mutablelivedata	1	9	11.11
build.version_codes.p	1	59	1.69	rs.getarray	1	9	11.11
uri.fromparts	1	56	1.79	karate junit5	1	9	11.11
hellokoding	1	55	1.82	stepverifier examples	1	9	11.11
slf4j example	1	46	2.17	getdecorview android	1	9	11.11
android resources getidentifier	1	43	2.33	expectedexception expectcause	1	8	12.5
apisonar	1	41	2.44	openhtmltopdf java example	1	8	12.5
mockito verifynointeractions	1	38	2.63	"getwindow().setflags(windowmanager.layoutparams.flag_fullscreen, windowmanager.layoutparams.flag_fullscreen);"	1	7	14.29
context.bind_auto_create	1	36	2.78	org.springframework.messaging.message example	1	7	14.29
timeunit.seconds.sleep	1	36	2.78	completablefuture.completedfuture(null)	1	7	14.29
mockito anylist	1	32	3.12	getdrawablestate	1	7	14.29
flux.fromiterable example	1	30	3.33	avro logicaltype date example	1	7	14.29
secretkeyfactory.getinstance	1	30	3.33	java security addprovider	1	7	14.29
settranslationx android	1	29	3.45	systemclock.uptimemillis	1	7	14.29
gettheme android	1	27	3.7	jsonvalue.parse java	1	6	16.67
android getexternalcachedir	1	27	3.7	mockwebserver enqueue	1	6	16.67
springapplication.setdefaultproperties	1	26	3.85	removecallbacksandmessages(null)	1	6	16.67
intent.setpackage	1	26	3.85	cookie.setpath	1	6	16.67
Top 50 (avg)	2	58	4.2				
Top 100 (avg)	1	35	6.9				

Note: Blue: API query. Black: Generic query. Red: Error query.

Generalization. In our evaluation, we analyzed hundreds of API usage examples provided by popular software projects. Despite these observations, our findings regarding readability and reusability—as usual in empirical software engineering—may not be directly generalized to other systems, particularly to commercial nor to the ones implemented in other programming languages.

10 | RELATED WORK

10.1 | Code search tools

Several code search tools are available online. Nowadays, it is possible to navigate in code search tools, such as SearchCode[26] and Krugle.[40] Code is also easy to find in GitHub with the support of its own search engine.[14]Over time, other code search tools were discontinued, such as codase,[41] OpenHub Code,[42] and Google Code Search.[43] Some code search tools are dedicated to API usage examples, such as ProgramCreek[22] and Codota.[23]

Several code search engines are proposed by the research community.[10,12-14,18,44-49] Sourcerer[47] indexes millions of lines of Java code and maintains a public database with this data. Exemplar[48,49] finds relevant software systems from large archives. Gu et al.[18] propose an approach that uses deep learning to rank code examples, in which semantically related words can be recognized. Keivanloo et al.[13] provide a low complexity approach to detect code examples that can be adopted by code search engines. Niu et al.[14] present a code search approach based on machine learning techniques.

We compare APISonar with the most popular programming websites that focus on API usage examples (ProgramCreek and Codota) and with the code search engine SearchCode. Our results show that APISonar provides better examples in terms of readability and reusability.

10.2 | API examples and API research

Nasehi et al.[11] present the characteristics of good code examples. They detect that good code examples are less complex and shorter, without implementation details. Other aspects include highlighting key elements of the solution and presenting multiple solutions. Robillard[50] states that, ideally, API documentation must include good code examples so that APIs can be easier to learn. Buse and Weimer[10] provide a technique for mining and synthesizing representative human-readable documentation of APIs. Moreno et al.[12] present Muse, an approach for mining and ranking method usage. Montandon et al.[15] provide a tool that instruments the API documentation with concrete usage examples, called APIMiner. Kim et al.[16] propose to improve documentation by generating rich API documents with code examples. Zhu et al.[17] provide an approach to mine API usage examples from test code. We would like to stress that APISonar is not an API search/recommendation engine. We focus on automatically generate collections of API code examples.

Previous studies also assess how developers search for code, particularly, analyzing usage logs.[8,51,52] For instance, Sadowski et al.[8] present how the developers search for code at Google, while Bajracharya and Lopes[51,52] provide a topic modeling assessment of usage logs of the Koders search engine. They find that API queries are commonly performed by developers.

Many other facets of APIs have been studied, for instance, with respect to their evolution,[53-56] popularity,[57-59] impact on client applications,[60-63] breaking changes,[63-66] deprecation,[67-71] reasons to change,[65,72] to name a few.

Our approach covers aspects of good code examples.[11] For instance, APISonar aims to present API examples that are small, less complex, and without implementation details; it also provides highlighted and multiple examples. Moreover, APISonar relies on metrics successfully adopted in the literature (i.e., similarity, readability, and reusability) to rank code examples.[12-14,18,24,30]

11 | CONCLUSION AND FUTURE WORK

This article presented APISonar, a novel approach to mine API usage examples from code repositories. Our examples are fully available on the web (http://apisonar.com) to help developers find out how to use APIs and discovering related APIs.

[14]https://help.github.com/en/articles/searching-code

Our approach is composed of four major steps (collecting, filtering, ranking, and presenting API examples) to spot and render API examples. To generate our website, we analyzed millions of source files provided by 4486 software projects hosted on GitHub, and extracted 11 million API usage examples about 1.5 million distinct APIs, leading to millions of navigable webpages.

We evaluated APISonar with respect to its quality and usage. Our results show that APISonar is a competitive solution, providing the best API examples in terms of readability and reusability, and covering a large amount of relevant APIs when compared with popular programming websites. We summarize our findings as follows: (1) APISonar examples are more readable and reusable than the ones provided by the popular programming websites; (2) APISonar examples are also more readable than the ones returned in Google search results; and (3) APISonar covers most APIs provided by relevant projects such as Java, Android, Apache, and Spring. In addition, despite being a novel website, APISonar attracted 3.7K users from 119 countries and has 25K webpages indexed by Google, suggesting that it provides relevant and useful content.

As future work, we envision the following directions. First, we plan to extend APISonar to other programming languages, such as Python and JavaScript, and explore other code quality measures in the ranking process. We plan to perform a qualitative analysis to validate the relevance of our API usage examples with developers. We also plan to improve the search capabilities of APISonar to facilitate finding API examples. Finally, another interesting research direction can be to improve the code examples by adding explanations in natural languages and links to API documentation so that developers can have more context to learn the APIs.

ACKNOWLEDGMENT

This research is supported by CAPES and CNPq.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in APISonar at http://apisonar.com.

ORCID

Andre Hora https://orcid.org/0000-0003-4900-1330

REFERENCES

1. Gu X, Zhang H, Zhang D, Kim S. Deep API learning. Paper presented at: Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering. Seattle, USA; 2016:631-642.
2. Stolee KT, Elbaum S, Dobos D. Solving the search for source code. *ACM Trans Softw Eng Methodol*. 2014;23(3):26.
3. Sim SE, Umarji M, Ratanotayanon S, Lopes CV. How well do search engines support code retrieval on the web? *ACM Trans Softw Eng Methodol*. 2011;21(1):4.
4. Sim SE, Agarwala M, Umarji M. *A Controlled Experiment on the Process Used by Developers During Internet-Scale Code Search*. New York, NY: Springer; 2013:53-77.
5. Brandt J, Guo PJ, Lewenstein J, Dontcheva M, Klemmer SR. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. Paper presented at: Proceedings of the Conference on Human Factors in Computing Systems. Boston, USA; 2009:1589-1598.
6. Robillard MP, Deline R. A field study of API learning obstacles. *Empir Softw Eng*. 2011;16(6):703-732.
7. Holmes R, Walker RJ. Systematizing pragmatic software reuse. *ACM Trans Softw Eng Methodol*. 2012;21(4):20.
8. Sadowski C, Stolee KT, Elbaum S. How developers search for code: a case study. Paper presented at: Proceedings of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE). Bergamo, Italy; 2015:191-201.
9. Parnin C, Treude C, Grammel L, Storey MA. *Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow. Technical Report*. Georgia, USA: Georgia Institute of Technology; 2012.
10. Buse RP, Weimer W. Synthesizing API usage examples. Paper presented at: Proceedings of the International Conference on Software Engineering. Zurich, Switzerland; 2012:782-792.
11. Nasehi SM, Sillito J, Maurer F, Burns C. What makes a good code example?a study of programming Q&A in StackOverflow. Paper presented at: Proceedings of the International Conference on Software Maintenance (ICSM). Riva del Garda, Italy; 2012:25-34.
12. Moreno L, Bavota G, Di Penta M, Oliveto R, Marcus A. How can I use this method? Paper presented at: Proceedings of the International Conference on Software Engineering. Florence, Italy; 2015:880-890.
13. Keivanloo I, Rilling J, Zou Y. Spotting working code examples. Paper presented at: Proceedings of the International Conference on Software Engineering. Hyderabad, India; 2014:664-675.
14. Niu H, Keivanloo I, Zou Y. Learning to rank code examples for code search engines. *Empir Softw Eng*. 2017;22(1):259-291.
15. Montandon JE, Borges H, Felix D, Valente MT. Documenting apis with examples: lessons learned with the apiminer platform. Paper presented at: Proceedings of the Working Conference on Reverse Engineering. Koblenz, Germany; 2013:401-408.
16. Kim J, Lee S, Hwang SW, Kim S. Enriching documents with examples: a corpus mining approach. *Trans Inf Syst*. 2013;31(1):1.

17. Zhu Z, Zou Y, Xie B, Jin Y, Lin Z, Zhang L. Mining API usage examples from test code. Paper presented at: Proceedings of the International Conference on Software Maintenance and Evolution. Victoria, Canada; 2014:301-310.
18. Gu X, Zhang H, Kim S. Deep code search. Paper presented at: Proceedings of the International Conference on Software Engineering (ICSE). Gothenburg, Sweden; 2018:933-944.
19. Treude C, Aniche M. Where does Google find API documentation? Paper presented at: Proceedings of the International Workshop on API Usage and Evolution. Gothenburg, Sweden; 2018:19-22.
20. Stack Overflow. https://stackoverflow.com/. Accessed December, 2019.
21. W3Schools. https://www.w3schools.com. Accessed December, 2019.
22. ProgramCreek. https://www.programcreek.com. Accessed December, 2019.
23. Codota. https://www.codota.com. Accessed December, 2019.
24. Buse RP, Weimer WR. Learning a metric for code readability. *IEEE Trans Softw Eng*. 2009;36(4):546-558.
25. Raghothaman M, Wei Y, Hamadi Y. Swim: synthesizing what i mean-code search and idiomatic snippet synthesis. Paper presented at: Proceedings of the International Conference on Software Engineering (ICSE). Austin, USA; 2016:357-367.
26. SearchCode. https://searchcode.com. Accessed December, 2019.
27. Dyer R, Nguyen HA, Rajan H, Nguyen TN. Boa: a language and infrastructure for analyzing ultra-large-scale software repositories. Paper presented at: Proceedings of the International Conference on Software Engineering. San Francisco, USA: IEEE; 2013:422-431.
28. Borges H, Valente MT. What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform. *J Syst Softw*. 2018;146:112–129.
29. Salton G, Wong A, Yang CS. A vector space model for automatic indexing. *Commun ACM*. 1975;18(11):613-620.
30. Bavota G, Russo B. A large-scale empirical study on self-admitted technical debt. Paper presented at: Proceedings of the Working Conference on Mining Software Repositories (MSR). Austin, USA; 2016:315-326.
31. Manning CD, Raghavan P, Schütze H. Scoring, term weighting and the vector space model. *Introduction to Information Retrieval*. Vol 100. Cambridge, MA: Cambridge University Press; 2008:2-4.
32. Swinscow TDV, Campbell MJ. *Statistics at Square One*. London, UK: BMJ Books, 2002.
33. Romano J, Kromrey JD, Coraggio J, Skowronek J. Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen?sd for evaluating group differences on the NSSE and other surveys. Paper presented at: Proceedings of the Annual Meeting of the Florida Association of Institutional Research. Florida, USA; 2006.
34. Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D. The promises and perils of mining GitHub. Paper presented at: Proceedings of theWorking Conference on Mining Software Repositories. Hyderabad, India; 2014:92-101.
35. Google Analytics https://marketingplatform.google.com/about/analytics. Accessed June, 2020.
36. Clifton B. *Advanced Web Metrics with Google Analytics*. Hoboken, NJ: John Wiley & Sons; 2012.
37. How search algorithms work. https://www.google.com/search/howsearchworks/algorithms. Accessed December, 2019.
38. Introduction to indexing. https://developers.google.com/search/docs/guides/intro-indexing. Accessed December, 2019.
39. Google search console. https://search.google.com/search-console. Accessed June, 2020.
40. Krugle http://opensearch.krugle.org; Accessed December, 2019.
41. Codase http://www.codase.com. Accessed December, 2019.
42. OpenHub https://code.openhub.net. Accessed December, 2019.
43. Google Code Search. Google Blog. Shutting down code search. https://googleblog.blogspot.com/2011/10/fall-sweep.html. Accessed December, 2019.
44. Stylos J, Myers BA. Mica: a web-search tool for finding api components and examples. *Visual Languages and Human-Centric Computing*. New York, USA: IEEE; 2006:195-202.
45. Kim J, Lee S, Hwang Sw, Kim S. Towards an intelligent code search engine. Paper presented at: Proceedings of the Conference on Artificial Intelligent. Atlanta, USA; 2010.
46. Brandt J, Dontcheva M, Weskamp M, Klemmer SR. Example-centric programming: integrating web search into the development environment. Paper presented at: Proceedings of the Conference on Human Factors in Computing Systems. Atlanta, USA; 2010:513-522.
47. Bajracharya S, Ngo T, Linstead E, et al. Sourcerer: a search engine for open source code supporting structure-based search. Paper presented at: Proceedings of the Symposium on Object-oriented Programming Systems, Languages, and Applications. Portland, USA; 2006:681-682.
48. Grechanik M, Fu C, Xie Q, McMillan C, Poshyvanyk D, Cumby C. Exemplar: executable examples archive. Paper presented at: Proceedings of the International Conference on Software Engineering. Cape Town, South Africa; 2010:259-262.
49. McMillan C, Grechanik M, Poshyvanyk D, Fu C, Xie Q. Exemplar: a source code search engine for finding highly relevant applications. *Trans Softw Eng*. 2012;38(5):1069-1087.
50. Robillard MP. What makes APIs hard to learn? answers from developers. *IEEE Softw*. 2009;26(6):27-34.
51. Bajracharya S, Lopes C. Mining search topics from a code search engine usage log. Paper presented at: Proceedings of the International Working Conference on Mining Software Repositories. Vancouver, Canada; 2009:111-120.
52. Bajracharya SK, Lopes CV. Analyzing and mining a code search engine usage log. *Empir Softw Eng*. 2012;17(4-5):424-466.
53. Henkel J, Diwan A. CatchUp! capturing and replaying refactorings to support API evolution. Paper presented at: Proceedings of the International Conference on Software Engineering. St. Louis, USA; 2005.
54. Dig D, Johnson R. How do APIs evolve? a story of refactoring: research articles. *J Softw Maint Evol Res Pract*. 2006;18(2):83–107.
55. McDonnell T, Ray B, Kim M. An empirical study of API stability and adoption in the android ecosystem. Paper presented at: Proceedings of the International Conference on Software Maintenance. Eindhoven, The Netherlands; 2013.

56. Hora A, Robbes R, Valente MT, Anquetil N, Etien A, Ducasse S. How do developers react to API evolution? a large-scale empirical study. *Softw Qual J*. 2018;26(1):161-191.
57. Mileva YM, Dallmeier V, Zeller A. Mining API popularity. Paper presented at: Proceedings of the International Academic and Industrial Conference on Testing - Practice and Research Techniques. Windsor, UK; 2010.
58. Hora A, Valente MT. apiwave: keeping track of api popularity and migration. Paper presented at: Proceedings of the International Conference on Software Maintenance and Evolution. Bremen, Germany; 2015. http://apiwave.com.
59. Lima C, Hora A. What are the characteristics of popular APIs? a large scale study on java, android and 165 libraries. *Softw Qual J*. 2020;28:425-458.
60. Businge J, Serebrenik A, Van Den Brand, M. G. Eclipse API usage: the good and the bad. *Softw Qual J* 2013;23:107–141.
61. Linares-Vásquez M, Bavota G, Bernal-Cárdenas C, Di Penta M, Oliveto R, Poshyvanyk D. API change and fault proneness: a threat to the success of android apps. Paper presented at: Proceedings of the International Symposium on the Foundations of Software Engineering. Saint Petersburg, Russia; 2013.
62. Bavota G, Linares-Vasquez M, Bernal-Cardenas CE, Di Penta M, Oliveto R, Poshyvanyk D. The impact of API change-and fault-proneness on the user ratings of android apps. *IEEE Trans Softw Eng*. 2015;41(4):384–407.
63. Xavier L, Brito A, Hora A, Valente MT. Historical and impact analysis of API breaking changes: a large scale study. Paper presented at: Proceedings of the International Conference on Software Analysis, Evolution and Reengineering. Klagenfurt, Austria; 2017.
64. Bogart C, Kästner C, Herbsleb J, Thung F. How to break an API: cost negotiation and community values in three software ecosystems. Paper presented at: Proceedings of the International Symposium on the Foundations of Software Engineering. Seattle, USA; 2016.
65. Brito A, Xavier L, Hora A, Valente MT. Why and how java developers break APIs. Paper presented at: Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). Campobasso, Italy; 2018:255-265.
66. Brito A, Xavier L, Hora A, Valente MT. APIDiff: detecting API breaking changes. Paper presented at: Proceedings of the International Conference on Software Analysis, Evolution and Reengineering. Campobasso, Italy; 2018.
67. Robbes R, Lungu M, Röthlisberger D. How do developers react to API deprecation? the case of a smalltalk ecosystem. Paper presented at: Proceedings of the International Symposium on the Foundations of Software Engineering; 2012.
68. Sawant AA, Robbes R, Bacchelli A. On the reaction to deprecation of 25,357 clients of 4+1 popular java APIs. Paper presented at: Proceedings of the 32nd International Conference on Software Maintenance and Evolution (ICSME). Raleigh, USA; 2016:400-410.
69. Sawant AA, Robbes R, Bacchelli A. On the reaction to deprecation of clients of 4+1 popular java APIs and the JDK. *Empir Softw Eng*. 2017;23:2158-2197.
70. Brito G, Hora A, Valente MT, Robbes R. On the use of replacement messages in API deprecation: an empirical study. *J Syst Softw*. 2018;137:306-321.
71. Sawant AA, Robbes R, Bacchelli A. To react, or not to react: Patterns of reaction to API deprecation. *Empir Softw Eng*. 2019;24:3824-3870.
72. Brito A, Valente MT, Xavier L, Hora A. You broke my code: understanding the motivations for breaking changes in APIs. *Empir Softw Eng*. 2020;25:1458-1492.

How to cite this article: Hora A. APISonar: Mining API usage examples. *Softw Pract Exper*. 2020;1–34. https://doi.org/10.1002/spe.2906