CODECATCH

Extracting Source Code Snippets from Online Sources

May 10, 2018

Themistoklis Diamantopoulos
Georgios Karagiannopoulos
Andreas Symeonidis

Electrical & Computer Engineering Dept.
Aristotle University of Thessaloniki

CONTENTS

INTRODUCTION

RECOMMENDATION SYSTEMS

The routine process of writing new code involves using search engines to find **snippets** from websites like StackOverflow, blogs, documentation etc.

This approach is time-consuming, distracting, and implies a lot of personal effort by the developer.

CodeCatch: A system that accelerates the process of searching and separating the solutions for a specific programming task.

RECOMMENDATION SYSTEMS

The routine process of writing new code involves using search engines to find **snippets** from websites like StackOverflow, blogs, documentation etc.

This approach is time-consuming, distracting, and implies a lot of personal effort by the developer.

CodeCatch: A system that accelerates the process of searching and separating the solutions for a specific programming task.

RELATED WORK

Many similar systems have been proposed in the past:

›Prospector, PARSEWeb, MAPO, UP-Miner, PAM, APIMiner, eXoaDocs, DECKARD, Blueprint, …‹

Deficits of the aforementioned systems:

- Too much knowledge is required beforehand for the API to be used.
- Most of them do not return ready-to-use snippets.
- Results are presented in form of lists, setting the different implementations difficult to separate.
- They preserve local indexes with limited size and diversity and sometimes outdated.
- Quality and reusability of results usally not evaluated.
- They involve some specialized query language requiring additional effort.

RELATED WORK

Many similar systems have been proposed in the past:

›Prospector, PARSEWeb, MAPO, UP-Miner, PAM, APIMiner, eXoaDocs, DECKARD, Blueprint, …‹

Deficits of the aforementioned systems:

→ Too much knowledge is required beforehand for the API to be used.
→ Most of them do not return ready-to-use snippets.
→ Results are presented in form of lists, setting the different implementations difficult to separate.
→ They preserve local indexes with limited size and diversity and sometimes outdated.
→ Quality and reusability of results usally not evaluated.
→ They involve some specialized query language requiring additional effort.

CODECATCH

CODECATCH SYSTEM OVERVIEW

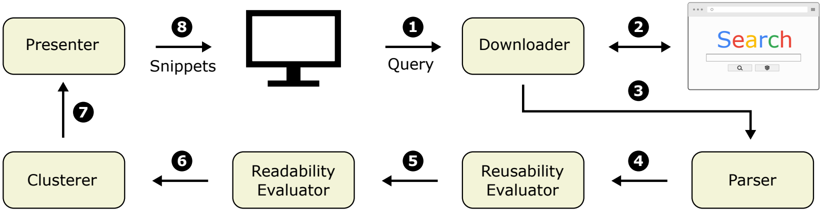
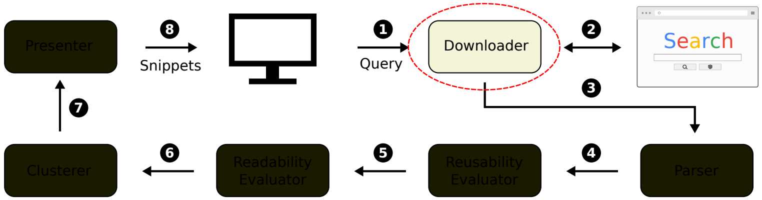


Figure: The components of CodeCatch and how they are connected.

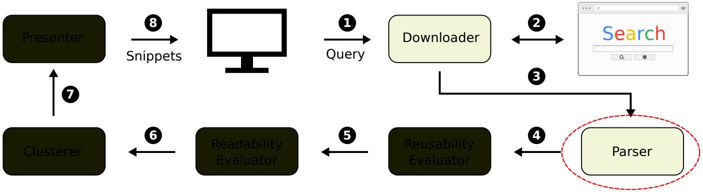
DOWNLOADER



Functionality of Downloader

1. Receives as input the query of the developer in natural language (e.g. "How to read a CSV file").
2. Augments the query with Java-related keywords in order to retrieve more relevant results.
3. Issues the query in the search engine and scrapes the returned web pages.

PARSER

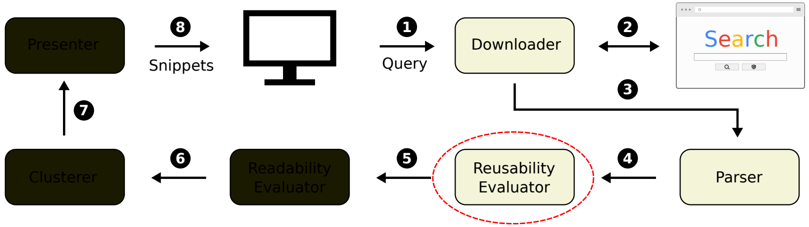


Functionality of Parser

1. Extracts the Abstract Syntax Tree (AST) of each snippet.
2. One pass over AST to extract type declarations and one pass to extract API calls.
3. Drops any snippets not referring to Java source code or not producing API calls.

Note: The parser is robust even when the snippets are not compilable.

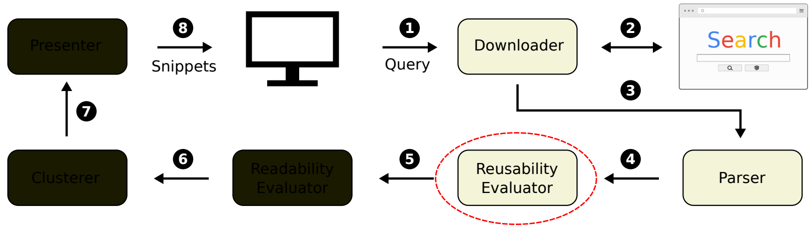
REUSABILITY EVALUATOR



Notion behind evaluating reusability...

Assumption: Snippets with API calls commonly used by other developers are more probable to be of reuse.
Reason: Frequently used APIs usually indicate common practice, thus are prone to be reused in different projects.

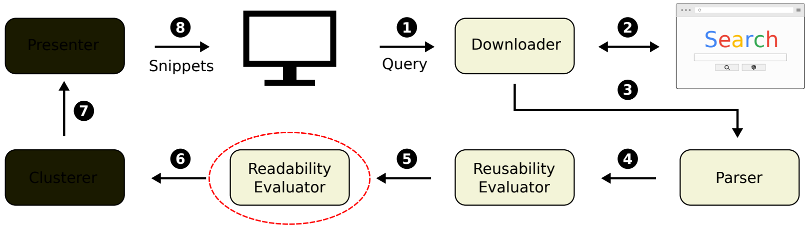
REUSABILITY EVALUATOR



Functionality of Reusability Evaluator

1. We download locally a set of high-quality projects.
2. We construct a local index where we store their API calls, extracted using the Parser.
3. We calculate the appearence frequency of each API call.
4. We evaluate the reusability of each snippet by averaging the scores of its API calls.

READABILITY EVALUATOR

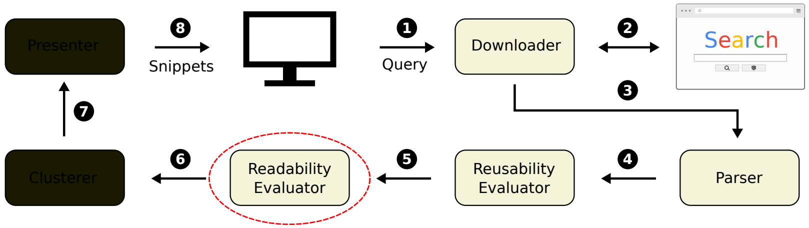


Definition

We define **readability** as the human judgement of how easy a text is to understand.

»Our target is to build a **classifier** using supervised-learning and a dataset of annotated snippets regarding their readability, which will be capable of judging the readability of new unseen snippets.«

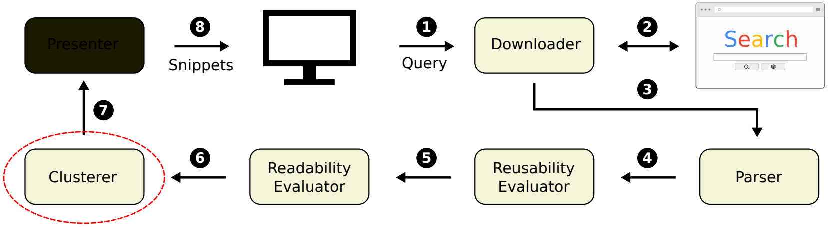
READABILITY EVALUATOR



Functionality of Readability Evaluator

1. We extract a set of 25 features that are related to readability (e.g. avg identifier length, avg number of comments, etc.
2. We build a binary classifier using AdaBoost and decision trees. The training was done on a publicly available dataset.
3. We evaluate the readability of new snippets using our trained model with 85% F-Measure score.

CLUSTERER

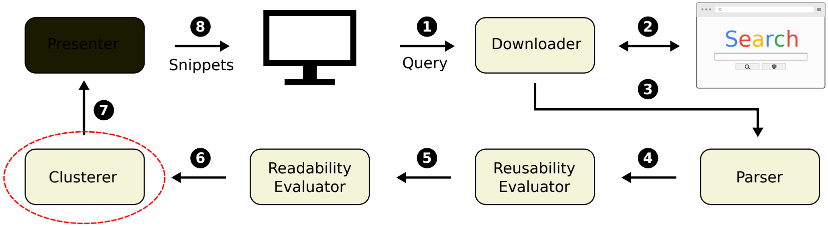


»Our target is to cluster the retrieved snippets based on their different implementations to the problem.«

Assumption...

...snippets which use different API calls represent different implementations of the problem.

CLUSTERER



»Our target is to cluster the retrieved snippets based on their different implementations to the problem.«

Assumption...

...snippets which use different API calls represent different implementations of the problem.

CLUSTERER

```
String line = "";
BufferedReader br = null;
try {
    br = new BufferedReader(new FileReader("test.csv"));
    while((line = br.readLine()) != null) {
        String[] data = line.split(",");
    }
    br.close();
} catch (Exception e) {
    System.err.println("CSV file cannot be read: " + e);
}
```

```
Scanner scanner = null;
try{
    scanner = new Scanner(new File("test.csv"));
    scanner.useDelimiter(",");
    while(scanner.hasNext()) {
        System.out.print(scanner.next() + " ");
    }
    scanner.close();
} catch (Exception e) {
    System.err.println("CSV file cannot be read: " + e);
}
```

CLUSTERING

```
String line = "";
BufferedReader br = null;
try {
    br = new BufferedReader (new FileReader ("test.csv"));
    while((line = br. readLine ()) != null) {
        String[] data = line.split(",");
    }
    br. close ();
} catch (Exception e) {
    System.err.println("CSV file cannot be read: " + e);
}
```

```
Scanner scanner = null;
try{
    scanner = new Scanner (new File ("test.csv"));
    scanner.useDelimiter(",");
    while(scanner. hasNext ()) {
        System.out.print(scanner. next () + " ");
    }
    scanner. close ();
} catch (Exception e) {
    System.err.println("CSV file cannot be read: " + e);
}
```

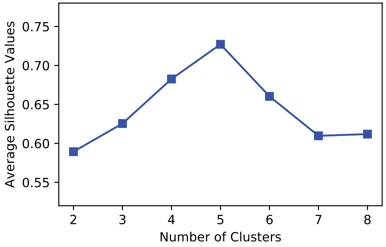
Clustering snippets by examining them as plain text documents is **not efficient**!

CLUSTERER

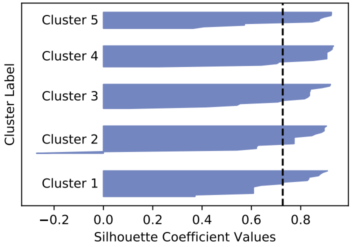
Preprocessing & Clustering

1. We represent each snippet as a vector in a **Vector Space Model** (VSM) with respect to its API calls.
2. We use a **tf-idf** vectorizer to extract the vector representation for each document.
3. We calculate the distance between snippets measuring the **cosine similarity**.
4. We perform **silhouette analysis** in order to determine the optimal number of clusters.
5. We employ **K-Means** algorithm for the final clustering, as it is known to be effective in text clustering problems like ours.

EXAMPLE SILHOUETTE ANALYSIS



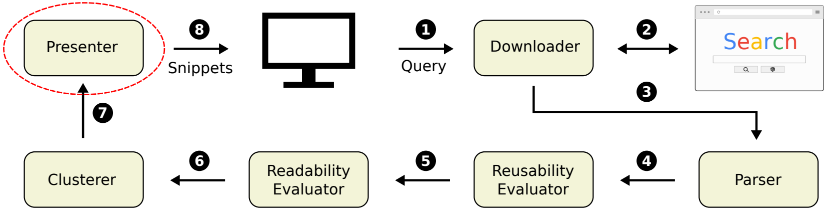
(a)



(b)

Figure: Example silhouette analysis for clustering the snippets of query "How to read a CSV file", including (a) the silhouette score for different number of clusters and (b) the silhouette of each of the 5 clusters.

PRESENTER



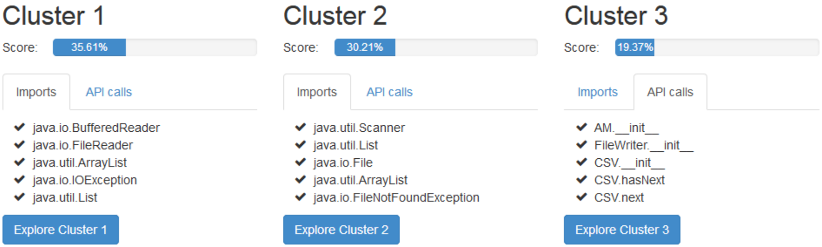
»The presenter handles the ranking and the presentation of the results.«

Ranking

The snippets within a cluster are ranked according to their API reusability score, and in cases of equal scores according to their distance from the cluster centroid.
The overall cluster score emerges from averaging the scores of its containing snippets.

PRESENTER

»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«



Cluster 1

Score: 35.61%

Imports | API calls

- ✓ java.io.BufferedReader
- ✓ java.io.FileReader
- ✓ java.util.ArrayList
- ✓ java.io.IOException
- ✓ java.util.List

Cluster 2

Score: 30.21%

Imports | API calls

- ✓ java.util.Scanner
- ✓ java.util.List
- ✓ java.io.File
- ✓ java.util.ArrayList
- ✓ java.io.FileNotFoundException

Cluster 3

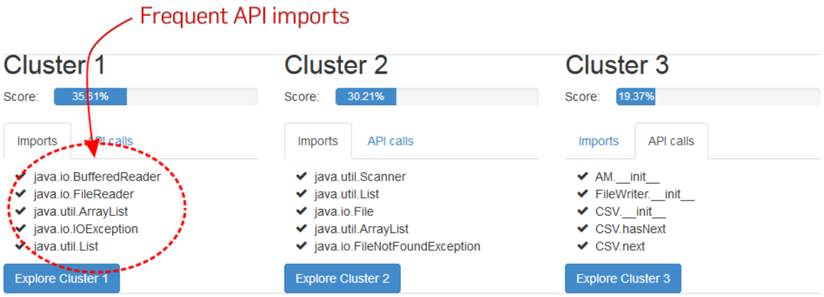
Score: 19.37%

Imports | API calls

- ✓ AM.__init__
- ✓ FileWriter.__init__
- ✓ CSV.__init__
- ✓ CSV.hasNext
- ✓ CSV.next

PRESENTER

»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«



Cluster 1
Score: 35.61%
Imports | API calls
- ✓ java.io.BufferedReader
- ✓ java.io.FileReader
- ✓ java.util.ArrayList
- ✓ java.io.IOException
- ✓ java.util.List

Explore Cluster 1

Cluster 2
Score: 30.21%
Imports | API calls
- ✓ java.util.Scanner
- ✓ java.util.List
- ✓ java.io.File
- ✓ java.util.ArrayList
- ✓ java.io.FileNotFoundException

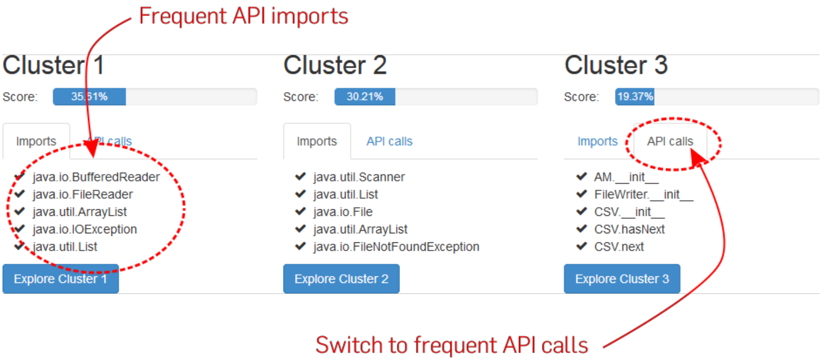
Explore Cluster 2

Cluster 3
Score: 19.37%
Imports | API calls
- ✓ AM.__init__
- ✓ FileWriter.__init__
- ✓ CSV.__init__
- ✓ CSV.hasNext
- ✓ CSV.next

Explore Cluster 3

PRESENTER

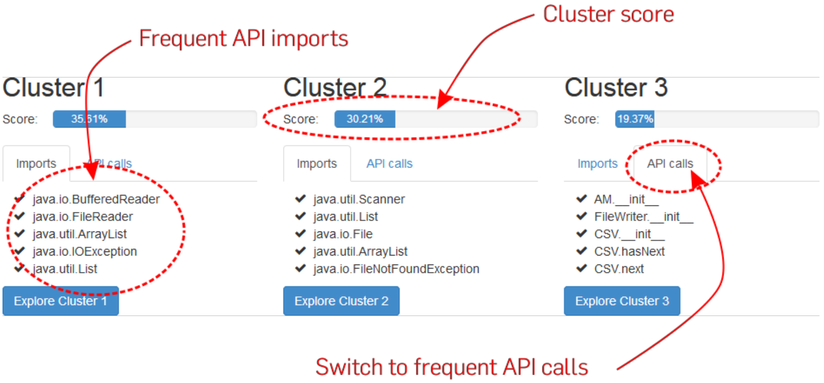
»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«



Cluster 1	Cluster 2	Cluster 3
Score: 35.61%	Score: 30.21%	Score: 19.37%
Imports / API calls	Imports / API calls	Imports / API calls
✓ java.io.BufferedReader	✓ java.util.Scanner	✓ AM.__init__
✓ java.io.FileReader	✓ java.util.List	✓ FileWriter.__init__
✓ java.util.ArrayList	✓ java.io.File	✓ CSV.__init__
✓ java.io.IOException	✓ java.util.ArrayList	✓ CSV.hasNext
✓ java.util.List	✓ java.io.FileNotFoundException	✓ CSV.next
Explore Cluster 1	Explore Cluster 2	Explore Cluster 3

PRESENTER

»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«



Cluster 1

Score: 35.61%

- ✓ java.io.BufferedReader
- ✓ java.io.FileReader
- ✓ java.util.ArrayList
- ✓ java.io.IOException
- ✓ java.util.List

Cluster 2

Score: 30.21%

- ✓ java.util.Scanner
- ✓ java.util.List
- ✓ java.io.File
- ✓ java.util.ArrayList
- ✓ java.io.FileNotFoundException

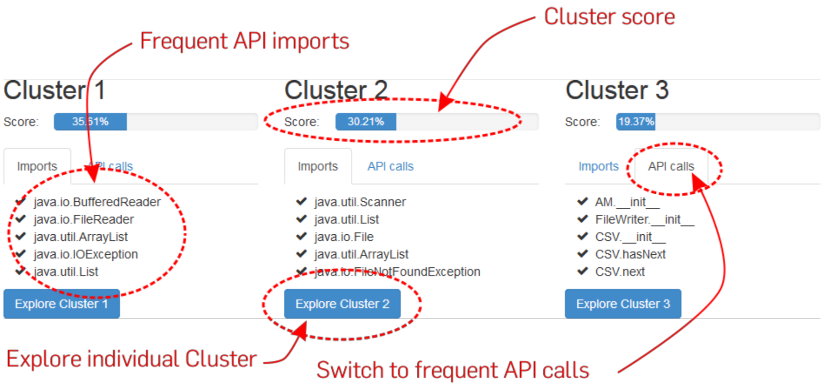
Cluster 3

Score: 19.37%

- ✓ AM.__init__
- ✓ FileWriter.__init__
- ✓ CSV.__init__
- ✓ CSV.hasNext
- ✓ CSV.next

PRESENTER

»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«



Cluster 1

Score: 35.61%

Imports | API calls

- ✓ java.io.BufferedReader
- ✓ java.io.FileReader
- ✓ java.util.ArrayList
- ✓ java.io.IOException
- ✓ java.util.List

Explore Cluster 1

Cluster 2

Score: 30.21%

Imports | API calls

- ✓ java.util.Scanner
- ✓ java.util.List
- ✓ java.io.File
- ✓ java.util.ArrayList
- ✓ java.io.FileNotFoundException

Explore Cluster 2

Cluster 3

Score: 19.37%

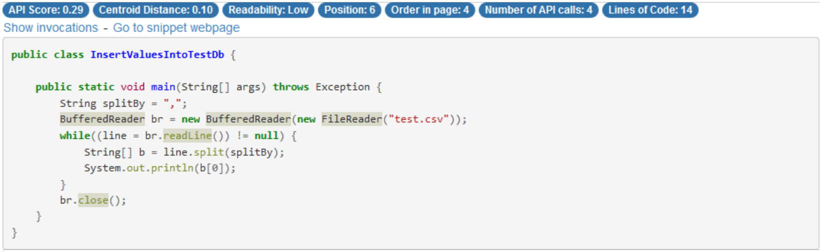
Imports | API calls

- ✓ AM.__init__
- ✓ FileWriter.__init__
- ✓ CSV.__init__
- ✓ CSV.hasNext
- ✓ CSV.next

Explore Cluster 3

PRESENTER

»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«


API Score: 0.29 | Centroid Distance: 0.10 | Readability: Low | Position: 6 | Order in page: 4 | Number of API calls: 4 | Lines of Code: 14

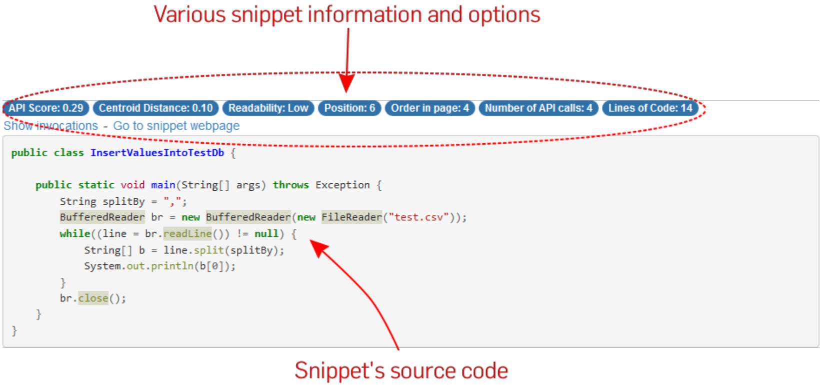
Show invocations - Go to snippet webpage

```
public class InsertValuesIntoTestDb {

    public static void main(String[] args) throws Exception {
        String splitBy = ",";
        BufferedReader br = new BufferedReader(new FileReader("test.csv"));
        while((line = br.readLine()) != null) {
            String[] b = line.split(splitBy);
            System.out.println(b[0]);
        }
        br.close();
    }
}
```

PRESENTER

»User inserts in CodeCatch the query **"How to read CSV file"** and initially gets presented with the clusters formed.«



```
public class InsertValuesIntoTestDb {

    public static void main(String[] args) throws Exception {
        String splitBy = ",";
        BufferedReader br = new BufferedReader(new FileReader("test.csv"));
        while((line = br.readLine()) != null) {
            String[] b = line.split(splitBy);
            System.out.println(b[0]);
        }
        br.close();
    }
}
```

SYSTEM EVALUATION

EVALUATION FRAMEWORK

The purpose of our evaluation is:

- → To assess whether the snippets of our snippets are relevant.
- → To determine whether the developer can indeed more easily find snippets for all different APIs relevant to a query.

We perform reusability-related evaluation against Google search engine on a dataset of common queries.

Index	Query	Clusters	Snippets	Snippets/Cluster
1	How to read CSV file	3	44	14.7
2	How to generate MD5 hash code	4	43	10.8
3	How to upload file to FTP	3	13	4.3
4	How to split string	5	53	10.6
5	How to draw text graphics	3	40	13.3
6	How to play audio file	5	79	15.8
7	How to substitute string	3	31	10.3
8	How to convert collection to an array	4	49	12.3
9	How to send email	3	46	15.3
10	How to connect to a JDBC database	5	44	8.8
11	How to execute select statement JDBC database	3	59	19.7
12	How to initialize thread	3	33	11.0
13	How to write binary data	3	35	11.7
14	How to read ZIP archive	2	32	16.0
15	How to send packet via UDP	2	31	15.5

EVALUATION FRAMEWORK

The purpose of our evaluation is:

- → To assess whether the snippets of our snippets are relevant.
- → To determine whether the developer can indeed more easily find snippets for all different APIs relevant to a query.

We perform reusability-related evaluation against Google search engine on a dataset of common queries.

Index	Query	Clusters	Snippets	Snippets/Cluster
1	How to read CSV file	3	44	14.7
2	How to generate MD5 hash code	4	43	10.8
3	How to upload file to FTP	3	13	4.3
4	How to split string	5	53	10.6
5	How to draw text graphics	3	40	13.3
6	How to play audio file	5	79	15.8
7	How to substitute string	3	31	10.3
8	How to convert collection to an array	4	49	12.3
9	How to send email	3	46	15.3
10	How to connect to a JDBC database	5	44	8.8
11	How to execute select statement JDBC database	3	59	19.7
12	How to initialize thread	3	33	11.0
13	How to write binary data	3	35	11.7
14	How to read ZIP archive	2	32	16.0
15	How to send packet via UDP	2	31	15.5

EVALUATION FRAMEWORK

- → The annotation procedure was performed without any knowledge on the ranking of snippets in order to maintain an objective outlook.
- → Snippets were marked as **relevant** iff their code covers the functionality described by the query.
- → We consider that the user examines the results subsequently (i.e. as a list of results) for both systems.
- → For further the assessment of each cluster, we annotate the results to consider them relevant for each implementation.
- → We use **reciprocal rank** as a metric for comparison.

EVALUATION FRAMEWORK

→ The annotation procedure was performed without any knowledge on the ranking of snippets in order to maintain an objective outlook.

→ Snippets were marked as **relevant** iff their code covers the functionality described by the query.

→ We consider that the user examines the results subsequently (i.e. as a list of results) for both systems.

→ For further the assessment of each cluster, we annotate the results to consider them relevant for each implementation.

→ We use **reciprocal rank** as a metric for comparison.

EVALUATION FRAMEWORK

→ The annotation procedure was performed without any knowledge on the ranking of snippets in order to maintain an objective outlook.

→ Snippets were marked as **relevant** iff their code covers the functionality described by the query.

→ We consider that the user examines the results subsequently (i.e. as a list of results) for both systems.

→ For further the assessment of each cluster, we annotate the results to consider them relevant for each implementation.

→ We use **reciprocal rank** as a metric for comparison.

EVALUATION FRAMEWORK

- → The annotation procedure was performed without any knowledge on the ranking of snippets in order to maintain an objective outlook.
- → Snippets were marked as **relevant** iff their code covers the functionality described by the query.
- → We consider that the user examines the results subsequently (i.e. as a list of results) for both systems.
- → For further the assessment of each cluster, we annotate the results to consider them relevant for each implementation.
- → We use **reciprocal rank** as a metric for comparison.

EVALUATION FRAMEWORK

→ The annotation procedure was performed without any knowledge on the ranking of snippets in order to maintain an objective outlook.

→ Snippets were marked as **relevant** iff their code covers the functionality described by the query.

→ We consider that the user examines the results subsequently (i.e. as a list of results) for both systems.

→ For further the assessment of each cluster, we annotate the results to consider them relevant for each implementation.

→ We use **reciprocal rank** as a metric for comparison.

EVALUATION RESULTS

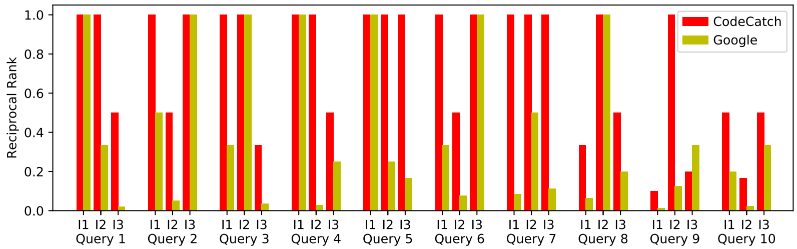
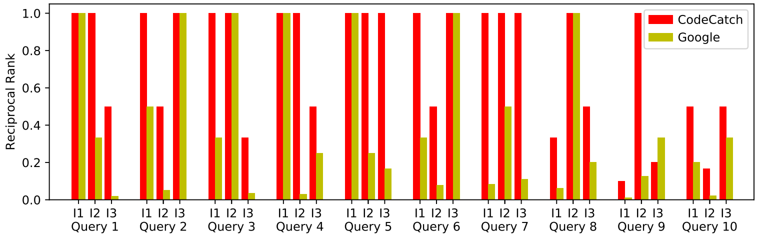


Figure: Reciprocal Rank of CodeCatch and Google for the three most popular implementations (I1, I2, I3) of each query.

EVALUATION CONCLUSIONS

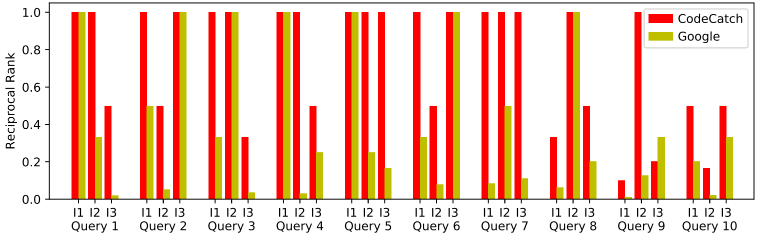


→ In terms of the **relevance** of the results both systems are very effective.

→ In some cases (e.g. Q1 - "How to read CSV file") developers can find relevant snippets for different implementations using CodeCatch **quicker**.

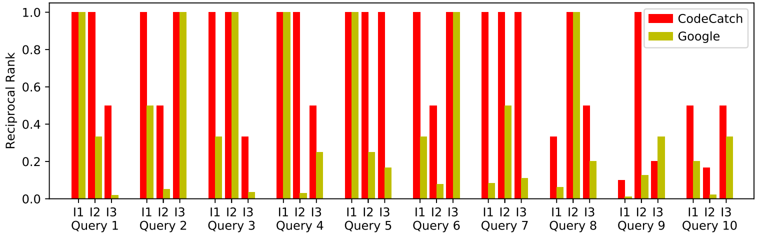
→ CodeCatch places the **most popular** implementation at the top of the list more often than Google (e.g. queries 2, 3, 6, 7, 10).

EVALUATION CONCLUSIONS



- → In terms of the **relevance** of the results both systems are very effective.
- → In some cases (e.g. Q1 - "How to read CSV file") developers can find relevant snippets for different implementations using CodeCatch **quicker**.
- → CodeCatch places the **most popular** implementation at the top of the list more often than Google (e.g. queries 2, 3, 6, 7, 10).

EVALUATION CONCLUSIONS



→ In terms of the **relevance** of the results both systems are very effective.

→ In some cases (e.g. Q1 - "How to read CSV file") developers can find relevant snippets for different implementations using CodeCatch **quicker**.

→ CodeCatch places the **most popular** implementation at the top of the list more often than Google (e.g. queries 2, 3, 6, 7, 10).

CONCLUSIONS - FUTURE WORK

Conclusion from our work:

- → Software engineers can save valuable time using recommendation systems.
- → Assessing the readability and reusability can improves the quality of results.
- → Grouping results into clusters provides a comprehensive view of the different implementations.

Future work could be directed into:

- → Improving the ranking scheme to include further information (e.g. developer's preference)
- → Performing snippet summarization using information from the clustering.
- → Conducting a developer study in order to further assess CodeCatch for its effectiveness.

CONCLUSIONS - FUTURE WORK

Conclusion from our work:

→ Software engineers can save valuable time using recommendation systems.
→ Assessing the readability and reusability can improves the quality of results.
→ Grouping results into clusters provides a comprehensive view of the different implementations.

Future work could be directed into:

→ Improving the ranking scheme to include further information (e.g. developer's preference)
→ Performing snippet summarization using information from the clustering.
→ Conducting a developer study in order to further assess CodeCatch for its effectiveness.

CodeCatch web application is available at:
http://codecatch.ee.auth.gr

We thank you for your attention!
Questions?