

Summarizing Software API Usage Examples Using Clustering Techniques

N. Katirtzis^{1,2} T. Diamantopoulos³ and C. Sutton²

¹Hotels.com

²School of Informatics
University of Edinburgh, Edinburgh, UK

³Electrical and Computer Engineering Department
Aristotle University of Thessaloniki, Thessaloniki, Greece

21st International Conference on Fundamental Approaches to
Software Engineering (*FASE*), 2018



CLustering for Api Mining of Snippets

CLAMS

An approach for mining API usage examples from client code that lies between snippet and sequence mining methods, which ensures lower complexity and thus could apply more readily to other languages.

Sample Snippet

```
AccessToken accessToken;  
String oauthToken;  
String oAuthVerifier;  
Twitter twitter;  
try {  
    accessToken = twitter.getOAuthAccessToken(oauthToken,oAuthVerifier);  
    // Do something with accessToken  
} catch (TwitterException e) {  
    e.printStackTrace();  
}
```



Outline

① Background

Introduction

Problem Statement

Related Work

② Methodology

The Concept

System Overview

Deploying to New Languages

③ Evaluation

Evaluation Framework

Evaluation Results



Outline

① Background

Introduction

Problem Statement

Related Work

② Methodology

The Concept

System Overview

Deploying to New Languages

③ Evaluation


Evaluation Framework

Evaluation Results



Introduction

- Third-party libraries are used heavily during SDLC.
- Lack of proper documentation for the APIs of the libraries.
- Creating API usage examples is time-consuming.



Time = \$

Outline

① Background

Introduction

Problem Statement

Related Work

② Methodology

The Concept

System Overview

Deploying to New Languages

③ Evaluation

Evaluation Framework

Evaluation Results



Problem Statement



The Problem

Automatically identify a set of patterns that characterize how an API is typically used from a corpus of client code (*API usage mining*).

Outline

① Background

Introduction

Problem Statement

Related Work

② Methodology

The Concept

System Overview

Deploying to New Languages

③ Evaluation

Evaluation Framework

Evaluation Results



Related Work

- Systems that Output API Call Sequences

```
Twitter.setOAuthConsumer  
Twitter.setOAuthAccessToken
```

- Systems that Output Source Code Snippets

```
String mConsumerKey;  
Twitter mTwitter;  
AccessToken mAccessToken;  
String mSecretKey;  
if (mAccessToken != null) {  
    mTwitter.setOAuthConsumer(mConsumerKey, mSecretKey);  
    mTwitter.setOAuthAccessToken(mAccessToken);  
}
```



Related Work

Systems that Output API Call Sequences

- MAPO (frequent sequence mining, clustering)
- UP-Miner (clustering)
- PAM (probabilistic modeling)

Disadvantages

- API call sequences do not always describe important information like method arguments and control flow.
- The output cannot be directly included in one's code.



Related Work

Systems that Output Source Code Snippets

- eXoaDocs (clustering, program slicing)
- APIMiner (program slicing, association rules)
- Buse and Weimer (path-sensitive data flow analysis, clustering, pattern abstraction)

Disadvantages

- Rely on detailed semantic analysis and semantic features which can make them more difficult to deploy to new languages.
- Limited source code summarization capabilities.



Outline

1 Background

Introduction

Problem Statement

Related Work

2 Methodology

The Concept

System Overview

Deploying to New Languages

3 Evaluation

Evaluation Framework

Evaluation Results



The Concept



- 1 Cluster a large set of usage examples based on their API calls.
- 2 Generate summarized versions for the top snippets of each cluster.
- 3 Select the most representative snippet from each cluster, using a tree edit distance metric on the ASTs.
- 4 Rank the snippets in descending order of support.

Outline

1 Background

Introduction

Problem Statement

Related Work

2 Methodology

The Concept

System Overview

Deploying to New Languages

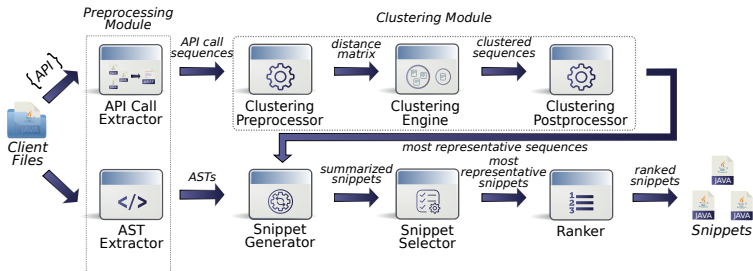
3 Evaluation

Evaluation Framework

Evaluation Results



System Overview



Input \Rightarrow $\begin{cases} \text{A set of Client Files} \\ \text{API of the library} \end{cases}$

Output \Rightarrow A set of source code snippets

Preprocessing Module

API Call Extractor

```
String mConsumerKey;  
Twitter mTwitter;  
AccessToken mAccessToken;  
String mSecretKey;  
if (mAccessToken != null) {  
    mTwitter.setOAuthConsumer(mConsumerKey,  
        mSecretKey);  
    mTwitter.setOAuthAccessToken(mAccessToken);  
}
```



```
Twitter.setOAuthConsumer  
Twitter.setOAuthAccessToken
```

AST Extractor

```
String mConsumerKey;  
Twitter mTwitter;  
AccessToken mAccessToken;  
String mSecretKey;  
if (mAccessToken != null) {  
    mTwitter.setOAuthConsumer(mConsumerKey,  
        mSecretKey);  
    mTwitter.setOAuthAccessToken(mAccessToken);  
}
```



```
<unit ... language="java" filename="test.java">  
  <decl_stmt><decl><type><name>String</name></  
    type><name>mConsumerKey</name></decl>;  
  </decl_stmt> ...  
  <if>if<condition>...</condition>  
  <then>  
    <block>{  
      ...  
    }</block>  
  </then>  
  </if>  
</unit>
```



Clustering Module

Clustering Preprocessor

- We cluster at sequence level:

```
editor.putString("", tkn.getToken());  
editor.putString("", tkn.getTokenSecret());
```

(a)

```
if (token != null) {  
    editor.putString("", token.getToken());  
    editor.putString("", token.getTokenSecret());  
}
```

(b)

- Using the distance matrix which is based on the *Longest Common Subsequence (LCS)* between any two API call sequences:

$$LCS_dist(S_1, S_2) = 1 - 2 \cdot \frac{|LCS(S_1, S_2)|}{|S_1| + |S_2|} \quad (1)$$

where $|S_1|$ and $|S_2|$ are the lengths of S_1 and S_2 , and $|LCS(S_1, S_2)|$ is the length of their LCS.



Clustering Module

Clustering Engine/Postprocessor

- Explores two different clustering algorithms:
 - ① **k-medoids** by Bauckhage.
 - ② **HDBSCAN** by McInnes et al.
- We then select multiple snippets for each cluster, this way retaining source code structure information, which shall be useful for selecting a single snippet.



Snippet Generator - Summarizer

Summarizer Input

```
if (t.getCreatedAt().getTime() + 1000 < mTime) {  
    breakPaging = 'y';  
    //TODO  
} else {  
    userName = t.getFromUser().toLowerCase();  
    JUser user = userMap.get(userName);  
    if (user == null) {  
        user = new JUser(userName).init(t);  
        userMap.put(userName, user);  
    }  
}
```

Step 2: Identify API statements

```
if (t.getCreatedAt().getTime() + number < mTime) {  
    breakPaging = char;  
} else {  
    userName = t.getFromUser().toLowerCase();  
    JUser user = userMap.get(userName);  
    if (user == null) {  
        user = new JUser(userName).init(t);  
        userMap.put(userName, user);  
    }  
}
```

Step 4: Remove non-API statements

```
if (t.getCreatedAt().getTime() + number < mTime) {  
} else {  
    userName = t.getFromUser().toLowerCase();  
}
```

Step 5: Filtering variables

```
if (t.getCreatedAt().getTime() + number < mTime) {  
} else {  
    userName = t.getFromUser().toLowerCase();  
}
```

Step 1: Preprocess comments and literals

```
if (t.getCreatedAt().getTime() + number < mTime) {  
    breakPaging = char;  
} else {  
    userName = t.getFromUser().toLowerCase();  
    JUser user = userMap.get(userName);  
    if (user == null) {  
        user = new JUser(userName).init(t);  
        userMap.put(userName, user);  
    }  
}
```

Step 3: Retrieve local scope variables

```
if (t.getCreatedAt().getTime() + number < mTime) {  
    breakPaging = char;  
} else {  
    userName = t.getFromUser().toLowerCase();  
    JUser user = userMap.get(userName);  
    if (user == null) {  
        user = new JUser(userName).init(t);  
        userMap.put(userName, user);  
    }  
}
```

Step 6: Add declaration statements and comments

```
long mTime;  
char char;  
JUser t;  
String userName;  
  
if (t.getCreatedAt().getTime() + number < mTime) {  
    // Do something  
} else {  
    userName = t.getFromUser().toLowerCase();  
    // Do something with userName  
}
```



Snippet Selector

Goal

Select the most representative snippet of each cluster.

Concept

- Create a matrix for each cluster, which contains the distance between any two top snippets of the cluster.
- Use the APTED algorithm to compute the tree edit distance between any two snippets.
- Select the snippet with the minimum sum of distances in each cluster's matrix.



Goal

Rank the snippets in descending order of support.

Concept

A client file supports a snippet if the API call sequence of the snippet is a subsequence of the sequence of that file.

Example

The snippet with API call sequence:

[`twitter4j.Status.getUser`, `twitter4j.Status.getText`]

is supported by a client file with sequence:

[`twitter4j.Paging.<init>`, `twitter4j.Status.getUser`,
`twitter4j.Status.getId`, `twitter4j.Status.getText`, `twitter4j.Status.getUser`].



Outline

1 Background

Introduction

Problem Statement

Related Work

2 Methodology

The Concept

System Overview

Deploying to New Languages

3 Evaluation

Evaluation Framework

Evaluation Results



Deploying to New Languages

Our methodology can be easily deployed to additional programming languages.

Table: Concept on which the CLAMS modules are based on.

Module	Main Concept
Preprocessing Module	AST
Clustering Module	API Call Sequence
Snippet Generator	Statement/Control Flow
Snippet Selector	AST
Ranker	API Call Sequence



Outline

1 Background

Introduction

Problem Statement

Related Work

2 Methodology

The Concept

System Overview

Deploying to New Languages

3 Evaluation

Evaluation Framework

Evaluation Results



Evaluation Framework

Dataset

Table: Summary of the evaluation dataset.

Project	Package Name	Client LOC	Example LOC
Apache Camel	org.apache.camel	141,454	15,256
Drools	org.drools	187,809	15,390
Restlet Framework	org.restlet	208,395	41,078
Twitter4j	twitter4j	96,020	6,560
Project Wonder	com.webobjects	375,064	37,181
Apache Wicket	org.apache.wicket	564,418	33,025



Evaluation Framework

Research Questions

-
- RQ1:** How much more concise, readable, and precise with respect to handwritten examples are the snippets after summarization?
 - RQ2:** Do more powerful clustering techniques, that cluster similar rather than identical sequences, lead to snippets that more closely match handwritten examples?
 - RQ3:** Does our tool mine more diverse patterns than other existing approaches?
 - RQ4:** Do snippets match handwritten examples more than API call sequences?
-

Figure: Research Questions (RQs) to be evaluated.



Outline

1 Background

Introduction

Problem Statement

Related Work

2 Methodology

The Concept

System Overview

Deploying to New Languages

3 Evaluation

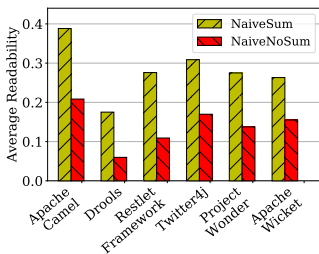
Evaluation Framework

Evaluation Results

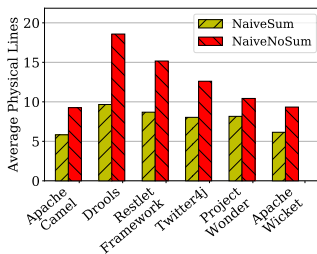


Evaluation Results - RQ1

RQ1: How much more **concise, **readable**, and precise with respect to handwritten examples are the snippets after summarization?**



(a)



(b)

Figure: (a) average readability, and (b) average PLOCs of the snippets, for each library, with (*NaiveSum*) and without (*NaiveNoSum*) summarization.



Evaluation Results - RQ1

RQ1: How much more concise, readable, and **precise with respect to handwritten examples are the snippets after summarization?**

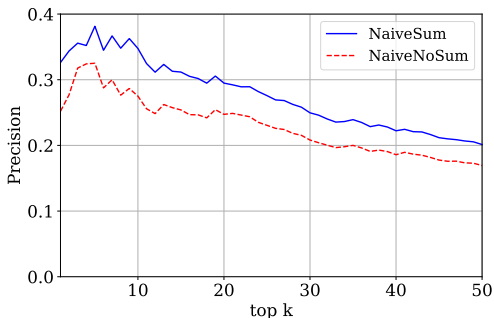


Figure: Precision at top k , with (*NaiveSum*) or without (*NaiveNoSum*) summarization using the top 50 mined snippets.



Evaluation Results - RQ2

RQ2: Do more powerful clustering techniques, that cluster similar rather than identical sequences, lead to snippets that more closely match handwritten examples?

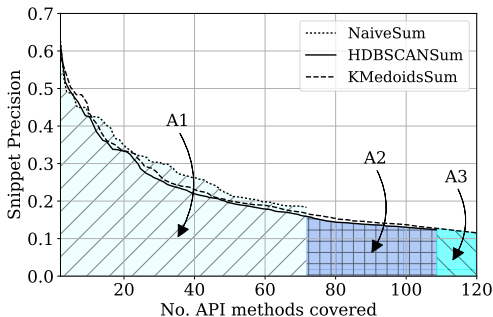


Figure: Average interpolated snippet precision versus API coverage for three clustering algorithms, using the top 100 mined snippets.

Evaluation Results - RQ3

RQ3: Does our tool mine more diverse patterns than other existing approaches?

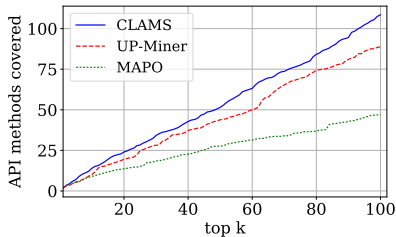


Figure: Coverage in API methods achieved by CLAMS, MAPO, and UP-Miner on average, at top k , using the top 100 examples.

Evaluation Results - RQ4

RQ4: Do source code snippets match handwritten examples more than API call sequences?

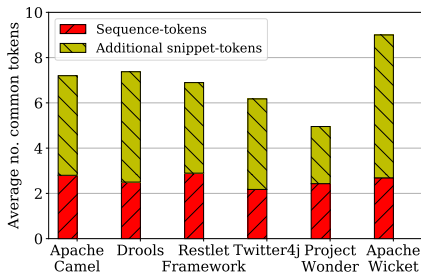


Figure: Additional information revealed when mining snippets instead of sequences.

Evaluation Results - RQ4

RQ4: Do source code snippets match handwritten examples more than API call sequences?

```
AccessToken accessToken;  
String oauthToken;  
String oauthVerifier;  
Twitter twitter;  
try {  
    accessToken = twitter.getOAuthAccessToken(oauthToken,oauthVerifier);  
    // Do something with accessToken  
} catch (TwitterException e) {  
    e.printStackTrace();  
}
```

Figure: Example snippet matched to handwritten example.
Sequence-tokens are encircled and additional snippet-tokens are highlighted in bold.



Applying CLAMS to the industry



- We conducted a pilot user survey at a team of Java developers at Hotels.com which received encouraging feedback:

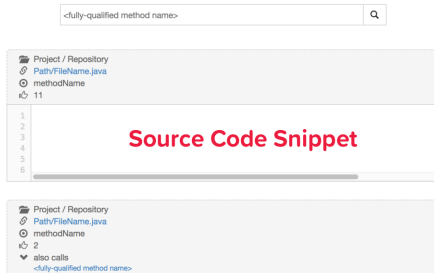
Developer 1: The system generates clear and concise snippets which would be easy to follow and useful when using an API you are unfamiliar with.

Developer 2: The system is great and I think it would be very useful particularly in discovering clients of our APIs!



Applying CLAMS to the industry

- We applied CLAMS at Hotels.com and developed an internal method-based search engine which can be used for API documentation purposes.



Resources



CLAMS Website

▶ <https://mast-group.github.io/clams/>



CLAMS User Survey at Hotels.com

▶ <https://mast-group.github.io/clams/user-survey/>



CLAMS Source Code

▶ <https://github.com/mast-group/clams>

