# Summarizing Source Code with Transferred API Knowledge

***Xing Hu*[1]** , Ge Li[1], Xin Xia[2], David Lo[3], Shuai Lu, Zhi Jin[1]

[1] Key Laboratory of High Confidence Software Technologies (PKU), China

[2] Faculty of Information Technology, Monash University, Australia

[3] School of Information Systems, Singapore Management University, Singapore

**IJCAI 2018**

# Why Code Summarization?

- **Code Comprehension**
  – Comments are often missed, mismatch, outdated, …
- **Summarization**
  – Aims to obtain a reductive transformation from a source text to a summary text through different techniques.

In software development and maintenance, developers spend around **59%** of their time on program comprehension activities[1]

*[1]Xia X, Bao L, Lo D, et al. Measuring program comprehension: A large-scale field study with professionals[J]. IEEE Transactions on Software Engineering, 2017.*

# Existing Approaches

- **Information Retrieval Approaches**
  - Extract natural descriptions from software artifacts, e.g., bug report, Stack Overflow…
  - Extract keywords from source code

**Limitations:**

➢ Heavily rely on whether similar code snippets can be retrieved and how similar the snippets are.

➢ Fail to extract accurate keywords when identifiers and methods are poorly named.

# Existing Approaches

- **Deep Learning based Approaches**
  - CODE-NN[2]
  - DeepCom[3]

**Limitations:**

➢ Simply treat the code summarization task as machine translation task

➢ Ignore some latent knowledge in the source code

*[2] Iyer S, Konstas I, Cheung A, et al. Summarizing Source Code using a Neural Attention Model. ACL. 2016.*

*[3] Xing Hu, Ge Li, Xin Xia, et al. Deep code comment generation. ICPC.2018*

# API Knowledge in Source Code

- **Developers often invoke a specific API sequence to implement a function.**

**“Parse XML files”**

```
DocumentBuilderFactory.newInstance
           ↓
DocumentBuilderFactory.newDocumentBuilder
           ↓
DocumentBuilder.parse
```

**“open a url”**

```
URL.new
   ↓
URL.openConnection
```

# API Knowledge in Source Code

- **Developers often invoke a specific API sequence to implement a function.**

**“Parse XML files”**

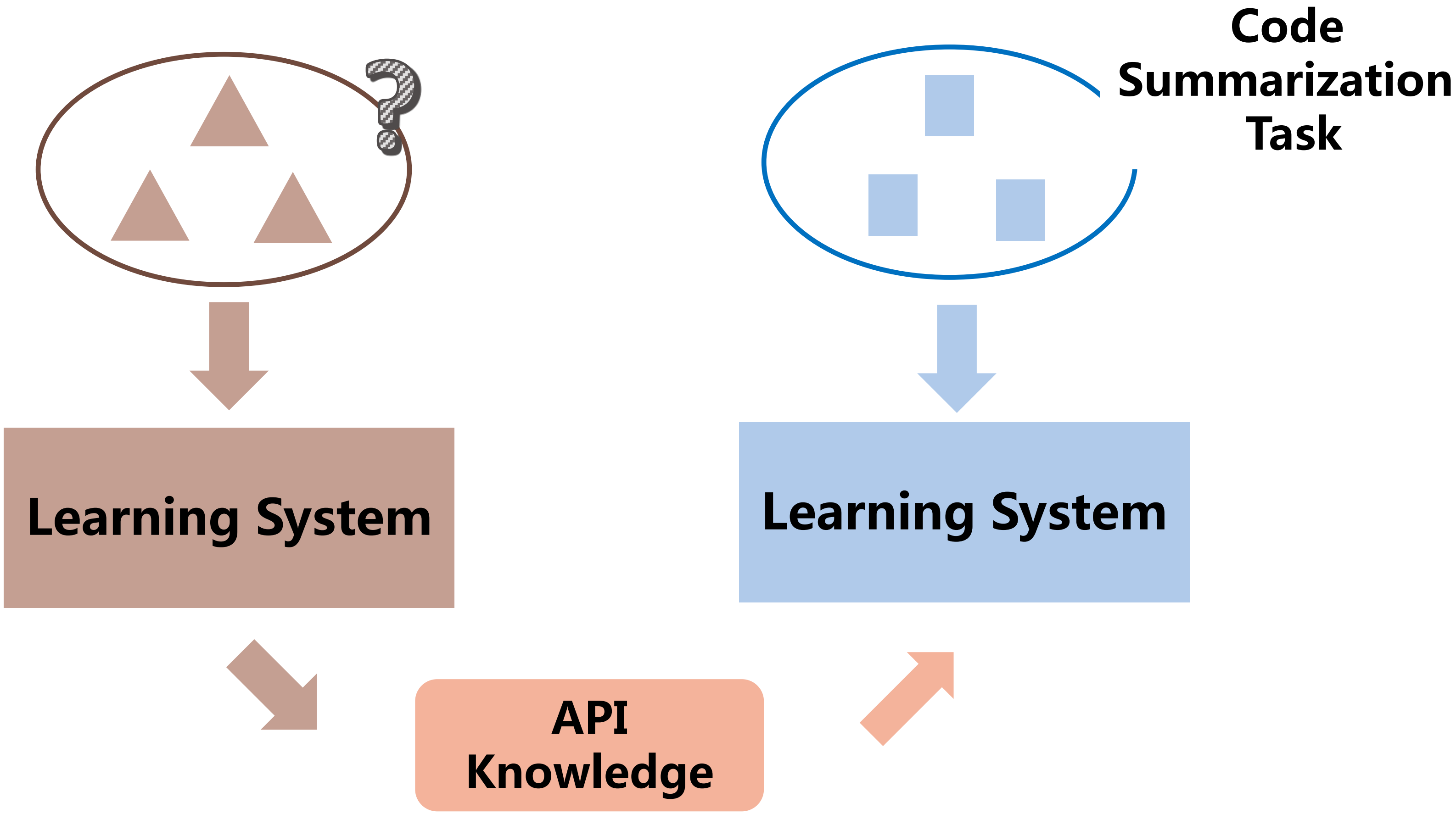
```
DocumentBuilderFactory.newInstance
           ↓
DocumentBuilderFactory.newDocumentBuilder
           ↓
DocumentBuilder.parse
```

**“open a url”**

```
URL.new
   ↓
URL.openConnection
```

**The latent knowledge in API sequence can assist the generation of code summaries.**

# To Better Use of API Knowledge



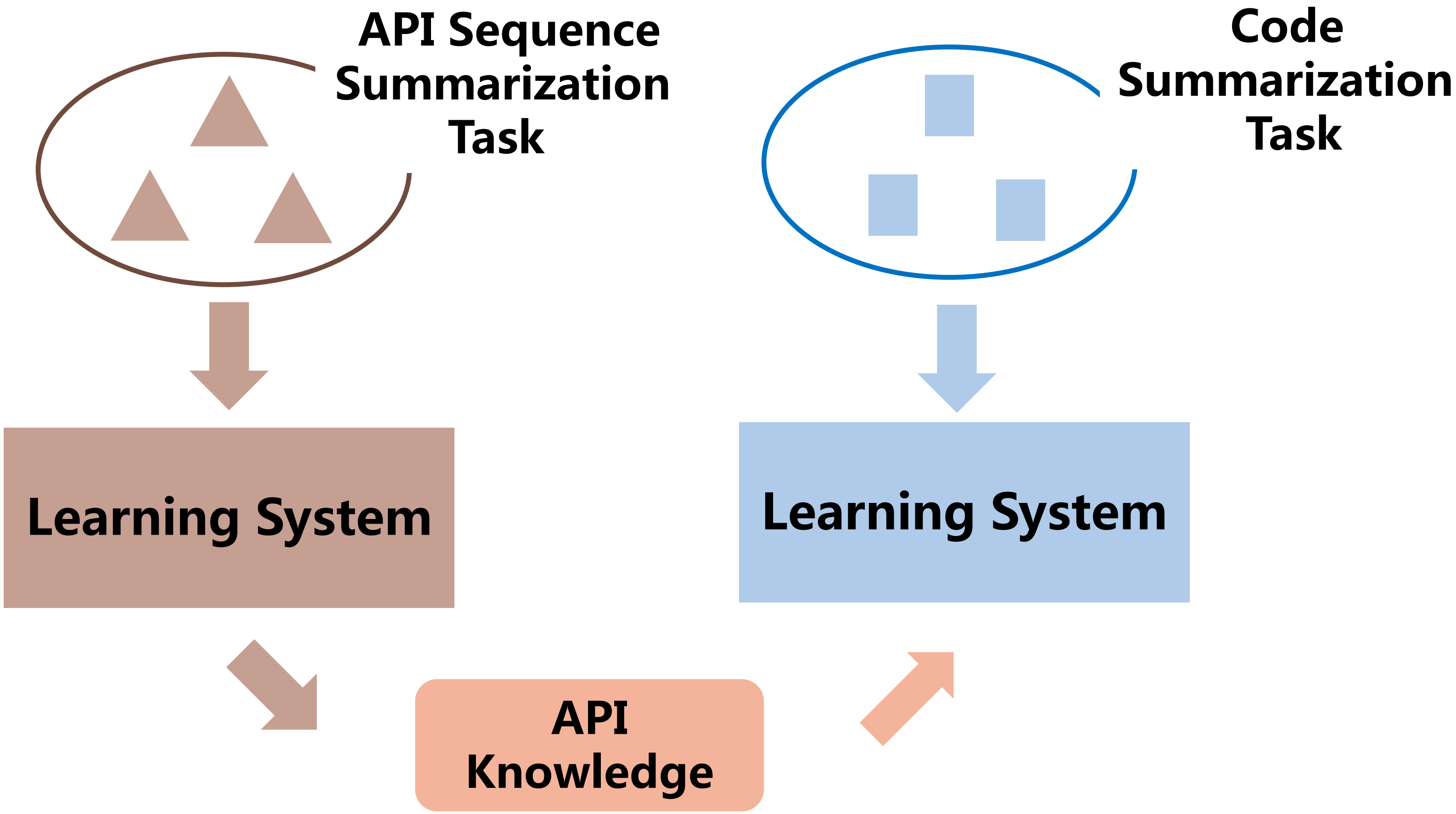
Code Summarization Task

Learning System

Learning System

API Knowledge

# To Better Use of API Knowledge



API Sequence Summarization Task

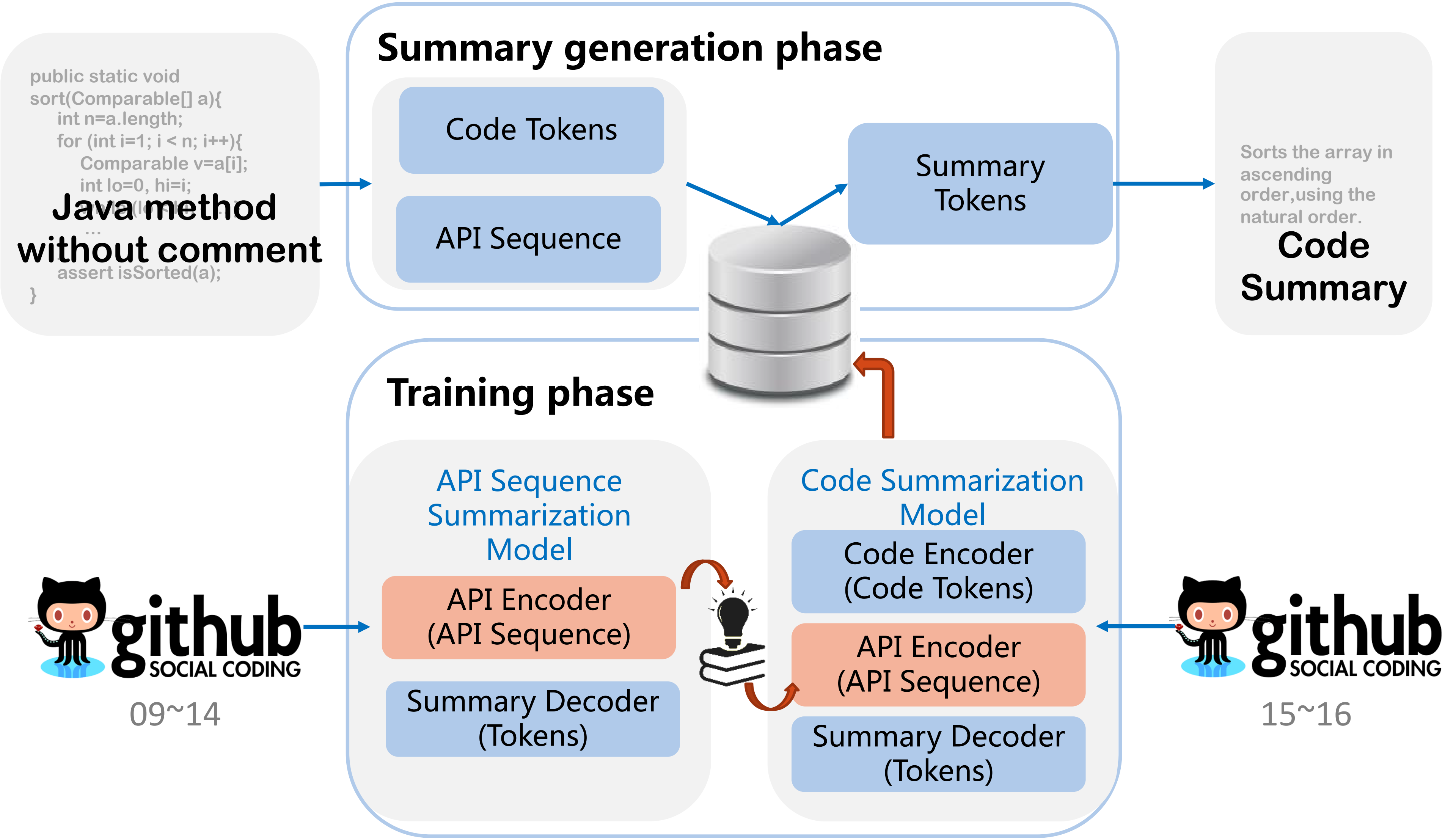
Code Summarization Task

Learning System

Learning System

API Knowledge

# TL-CodeSum: Workflow



```
public static void
sort(Comparable[] a){
    int n=a.length;
    for (int i=1; i < n; i++){
        Comparable v=a[i];
        int lo=0, hi=i;
        while (lo < hi) ...
         ...
    assert isSorted(a);
}
```

Sorts the array in ascending order,using the natural order.

# API Summarization Task

- **API sequence summarization aims to build the mappings between API knowledge and natural language descriptions.**

```
DataOutputStream.writeByte ->
DataOutputStream.writeShort->
DataOutputStream.writeShort
```

⇨

*"Write the constant to the output stream"*

# API Summarization Task

- **API sequence summarization aims to build the mappings between API knowledge and natural language descriptions.**

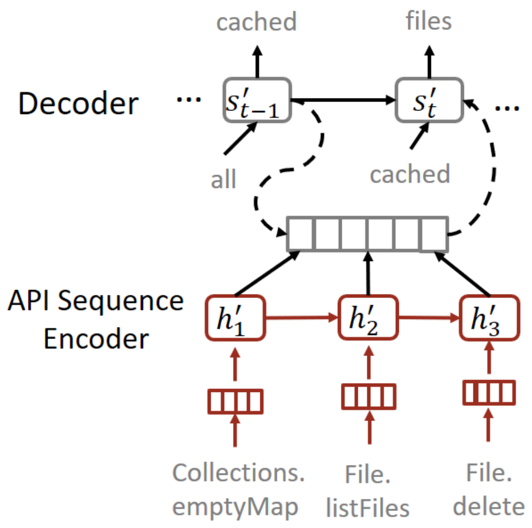
```
DataOutputStream.writeByte ->
DataOutputStream.writeShort->
DataOutputStream.writeShort
```

⇒

*“Write the constant to the output stream”*

**Attention based Seq2Seq**

- Encoder: embeds API sequence
- Decoder: generates NL descriptions with API vectors



# API Summarization Task

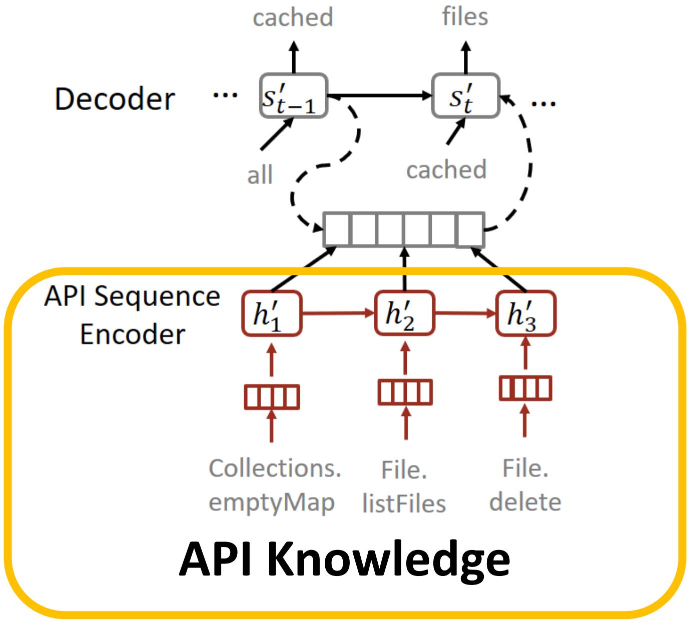
- **API sequence summarization aims to build the mappings between API knowledge and natural language descriptions.**

```
DataOutputStream.writeByte ->
DataOutputStream.writeShort->
DataOutputStream.writeShort
```

*“Write the constant to the output stream”*

**Attention based Seq2Seq**

- Encoder: embeds API sequence
- Decoder: generates NL descriptions with API vectors



# Code Summarization Task

```
void write(Environment env,
           DataOutputStream out,
           ConstantPool tab) throw IOException{
     out.writeByte(CONSTANT_NAMEANDTYPE);
     out.writeShort(tab.index(name));
     out.writeShort(tab.index(type));
}
```

⇨

*“Write the constant to the output stream”*

# Code Summarization Task

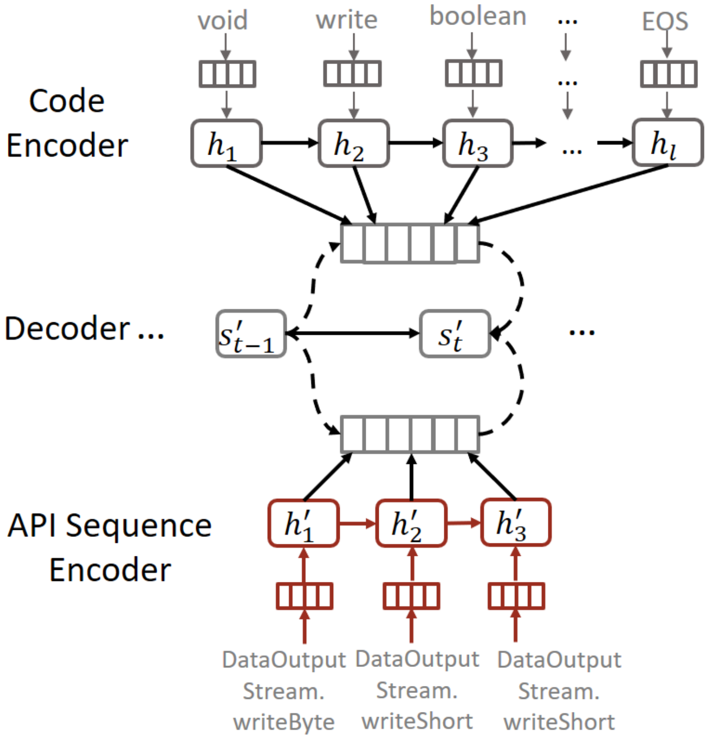
```
void write(Environment env,
           DataOutputStream out,
           ConstantPool tab) throw IOException{
     out.writeByte(CONSTANT_NAMEANDTYPE);
     out.writeShort(tab.index(name));
     out.writeShort(tab.index(type));
}
```

*“Write the constant to the output stream”*

# Code Summarization Task

```
void write(Environment env,
           DataOutputStream out,
           ConstantPool tab) throw IOException{
     out.writeByte(CONSTANT_NAMEANDTYPE);
     out.writeShort(tab.index(name));
     out.writeShort(tab.index(type));
}
```

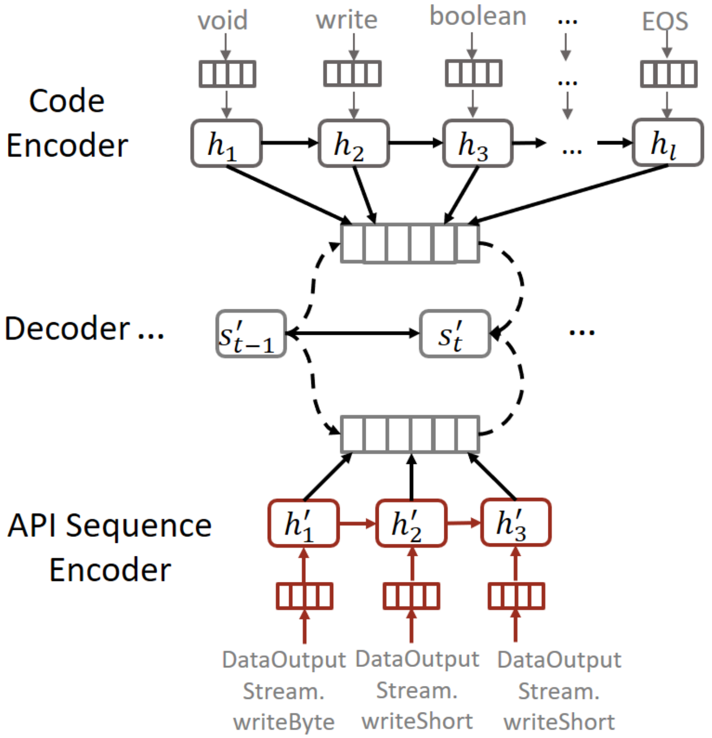
*“Write the constant to the output stream”*



# Code Summarization Task

```
void write(Environment env,
           DataOutputStream out,
           ConstantPool tab) throw IOException{
    out.writeByte(CONSTANT_NAMEANDTYPE);
    out.writeShort(tab.index(name));
    out.writeShort(tab.index(type));
}
```

*“Write the constant to the output stream”*

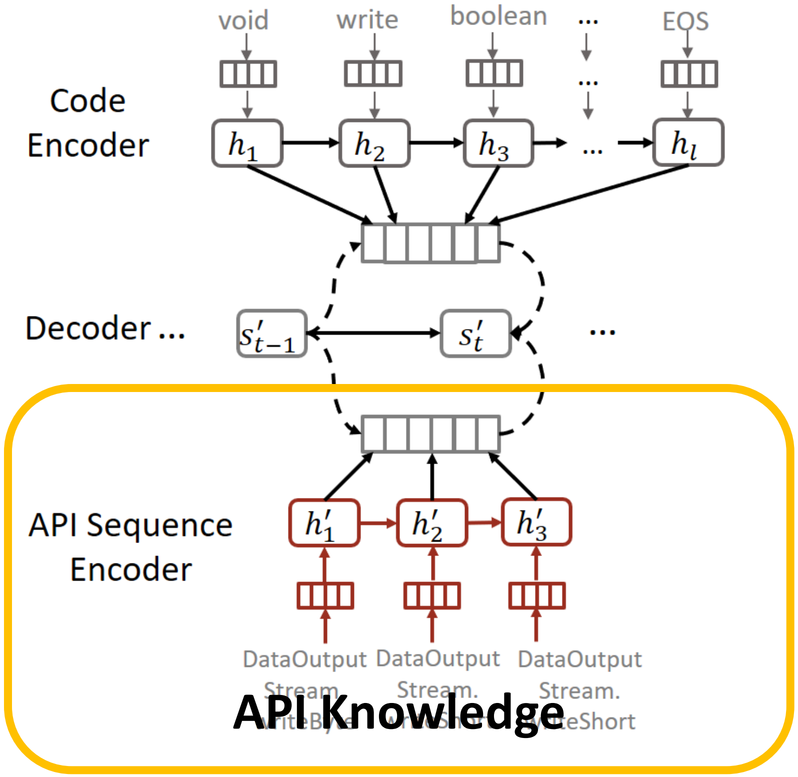


*DataOutputStream.writeByte –>*
*DataOutputStream.writeShort–>*
*DataOutputStream.writeShort*

# Code Summarization Task

```
void write(Environment env,
           DataOutputStream out,
           ConstantPool tab) throw IOException{
    out.writeByte(CONSTANT_NAMEANDTYPE);
    out.writeShort(tab.index(name));
    out.writeShort(tab.index(type));
}
```

*“Write the constant to the output stream”*

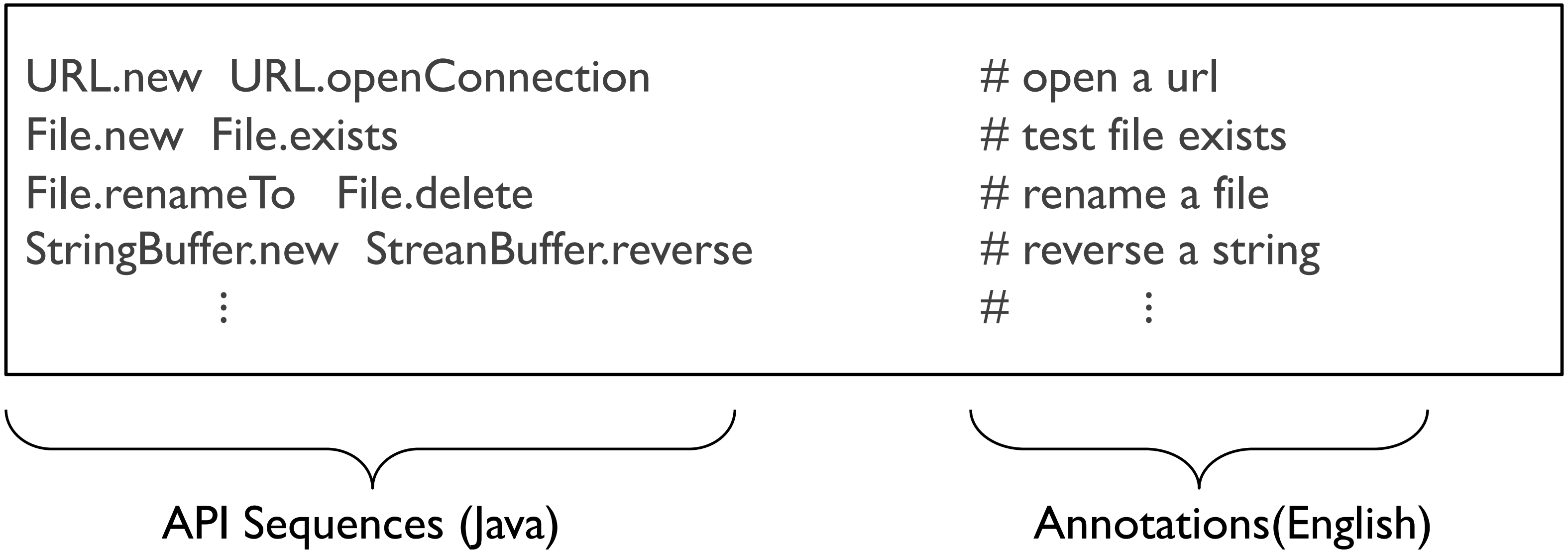


*DataOutputStream.writeByte –>*
*DataOutputStream.writeShort–>*
*DataOutputStream.writeShort*

# Collecting two Corpora

- **API Summarization Task**

<API Sequence, Annotation> pairs



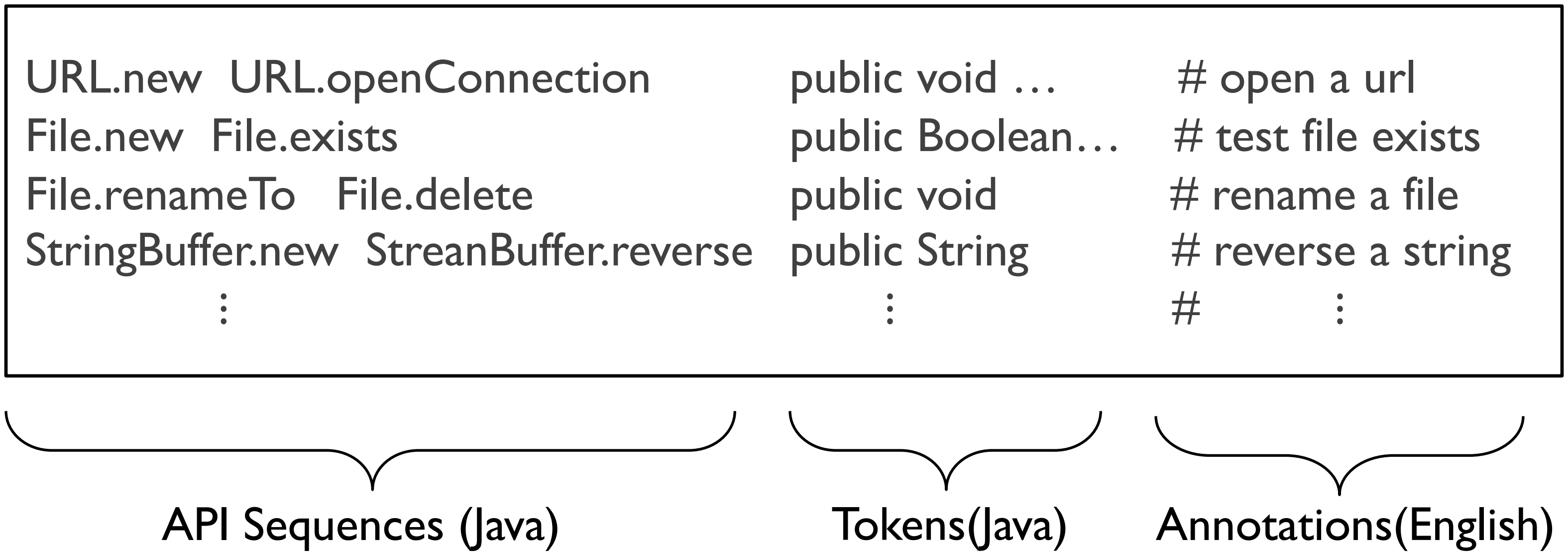
| API Sequences (Java) | Annotations(English) |
|---|---|
| URL.new  URL.openConnection | # open a url |
| File.new  File.exists | # test file exists |
| File.renameTo  File.delete | # rename a file |
| StringBuffer.new  StreanBuffer.reverse | # reverse a string |
| ⋮ | #  ⋮ |

- Collect 13,154 Java projects from GitHub (2008-2014)
- Extract an API sequence and an annotation for each method body (when Javadoc comment exists)
- 340,922 pairs

# Collecting two Corpora

- **Code Summarization Task**

<API Sequence, Code Tokens, Annotation> instances



| | | |
|---|---|---|
| URL.new URL.openConnection | public void … | # open a url |
| File.new File.exists | public Boolean… | # test file exists |
| File.renameTo File.delete | public void | # rename a file |
| StringBuffer.new StreanBuffer.reverse | public String | # reverse a string |
| ⋮ | ⋮ | # ⋮ |
| API Sequences (Java) | Tokens(Java) | Annotations(English) |

- Collect 9,732 Java projects from GitHub (2015-2016)
- Extract an API sequence, the method tokens and an annotation for each method body (when Javadoc comment exists)
- 69,708 instances

# Experiment

- **Experiment Settings**
  - GRU, 128 hidden states
  - 128 for API, code tokens, and summary embeddings
  - Batch size: 32
  - SGD algorithm
  - Vocabulary size: 50,000, 33,082, and 26,971 for code, API, and summaries respectively.
  - Beam size: 5

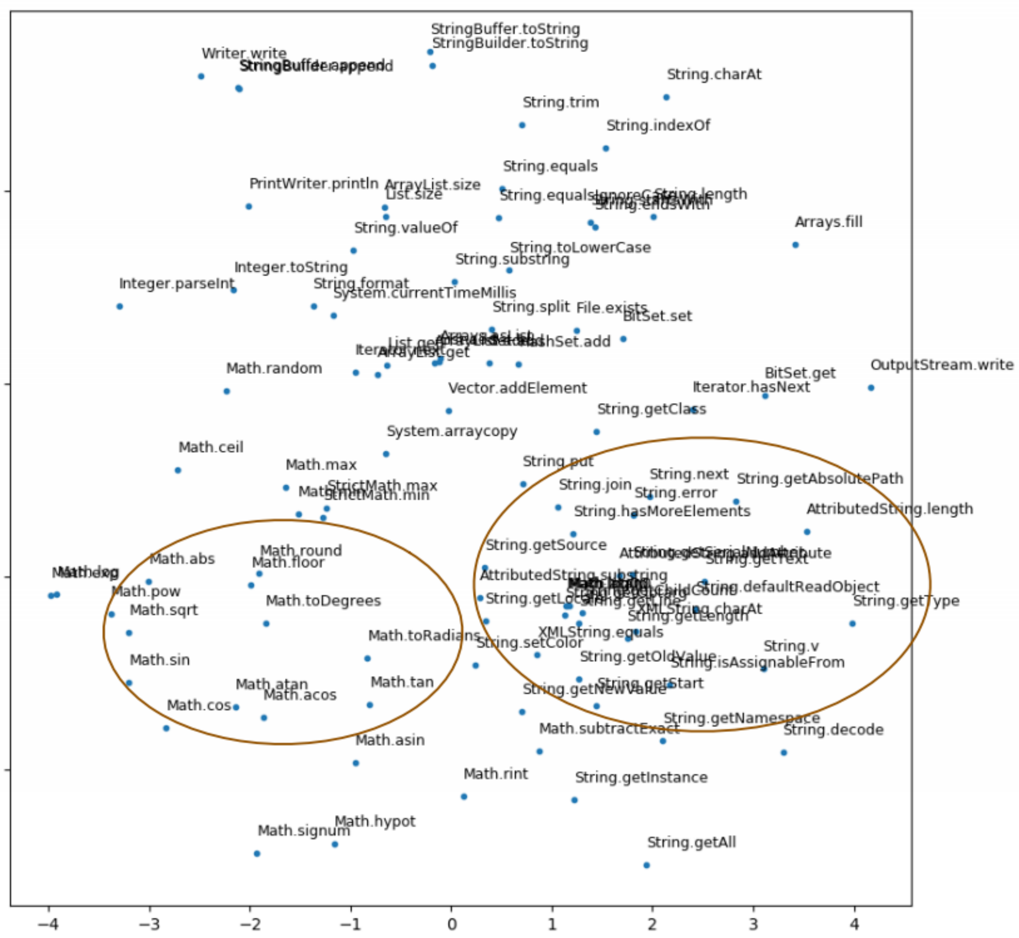
# Results-Accuracy

- **Baselines**
  - CODE-NN [Iyer, et al. ACL. 2016]
  - API-Only
  - Code-Only } NMT model
  - API+Code Two encoders and a decoder

- **Metric**
  - **IR**: Precision, Recall, F-Score
  - **NMT**: BLEU, METEOR

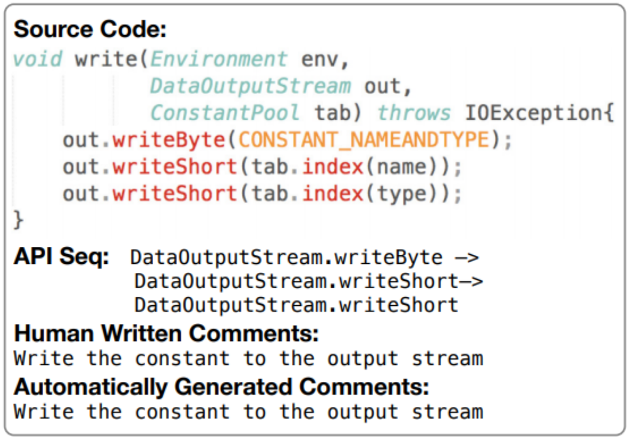
| Approaches | Precision | Recall | F-Score | BLEU score | METEOR |
|---|---|---|---|---|---|
| CODE-NN | 26.21 | 14.71 | 18.4 | 25.3 | 6.92 |
| API-Only | 30.72 | 21.14 | 25.05 | 26.45 | 10.71 |
| Code-Only | 38.89 | 28.81 | 33.10 | 35.50 | 14.78 |
| API + Code | 41.06 | 30.34 | 34.90 | 37.28 | 15.88 |
| TL-CodeSum(fix) | **42.20** | 34.38 | 37.89 | 36.42 | 18.07 |
| TL-CodeSum(fine-tuned) | 40.78 | **35.41** | **37.91** | **41.98** | **18.81** |

# Results-API Embedding



# Results-Complementarity of API and Code

- **Attention weights for the API sequence and code tokens while generating summaries**



**Source Code:**

```
void write(Environment env,
           DataOutputStream out,
           ConstantPool tab) throws IOException{
    out.writeByte(CONSTANT_NAMEANDTYPE);
    out.writeShort(tab.index(name));
    out.writeShort(tab.index(type));
}
```

**API Seq:**

```
DataOutputStream.writeByte ->
DataOutputStream.writeShort->
DataOutputStream.writeShort
```

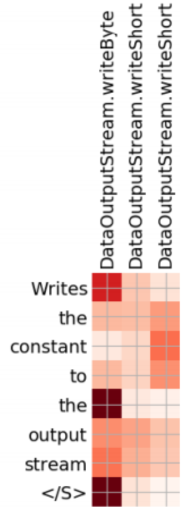
**Human Written Comments:**

```
Write the constant to the output stream
```

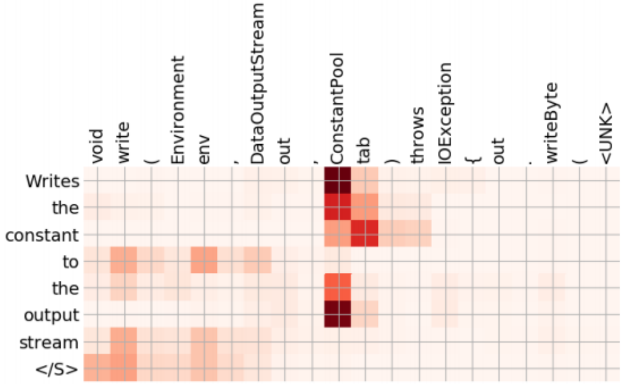
**Automatically Generated Comments:**

```
Write the constant to the output stream
```

(a) An example of code snippet



(b) Attention weights for API sequences



(c) Attention weights for source code tokens

TL-CodeSum aligns different words with specific API or code tokens.

# Results-Examples

```java
protected void sprint(double doubleField){
    sprint(String.valueOf(doubleField));
}
```

**API Seq:** String.valueOf
**Human-Written:** Pretty printing accumulator function for *doubles*
**TL-CodeSum:** pretty printing accumulator function for *longs*

### Word Replacement

Some words are replaced by their synonyms, antonyms, or words in the same domain.

```java
public void removeMouseListener(GlobalMouseListener
listener){
     listeners.remove(listener);
}
```

**API Seq:** List.remove
**Human-Written:** Removes a *global mouse* listener
**TL-CodeSum:** removes an *existing message* listener

### More general

The generated summaries may present more general meaning and give the abstract semantics of given Java methods.

# Results-Examples

```
private static boolean instanceOfAny(Object o,
Collection<Class> classes){
    for(Class c: classes){
        if (c.isInstance(o))
            return true;
    }
    return false;
}
```

**API Seq:** Collection.isEmpty->Collection.add->Class.isInstance

**Human-Written:** returns true if the Object '*o*' is an instance of any class in the *Collection*

**TL-CodeSum:** returns true if the object is registered in classes, or false otherwise.

### Missed Identifiers

Learning the identifiers is challenging. TL-CodeSum misses some identifiers or replaces them with "UNK" sometimes

# Conclusion

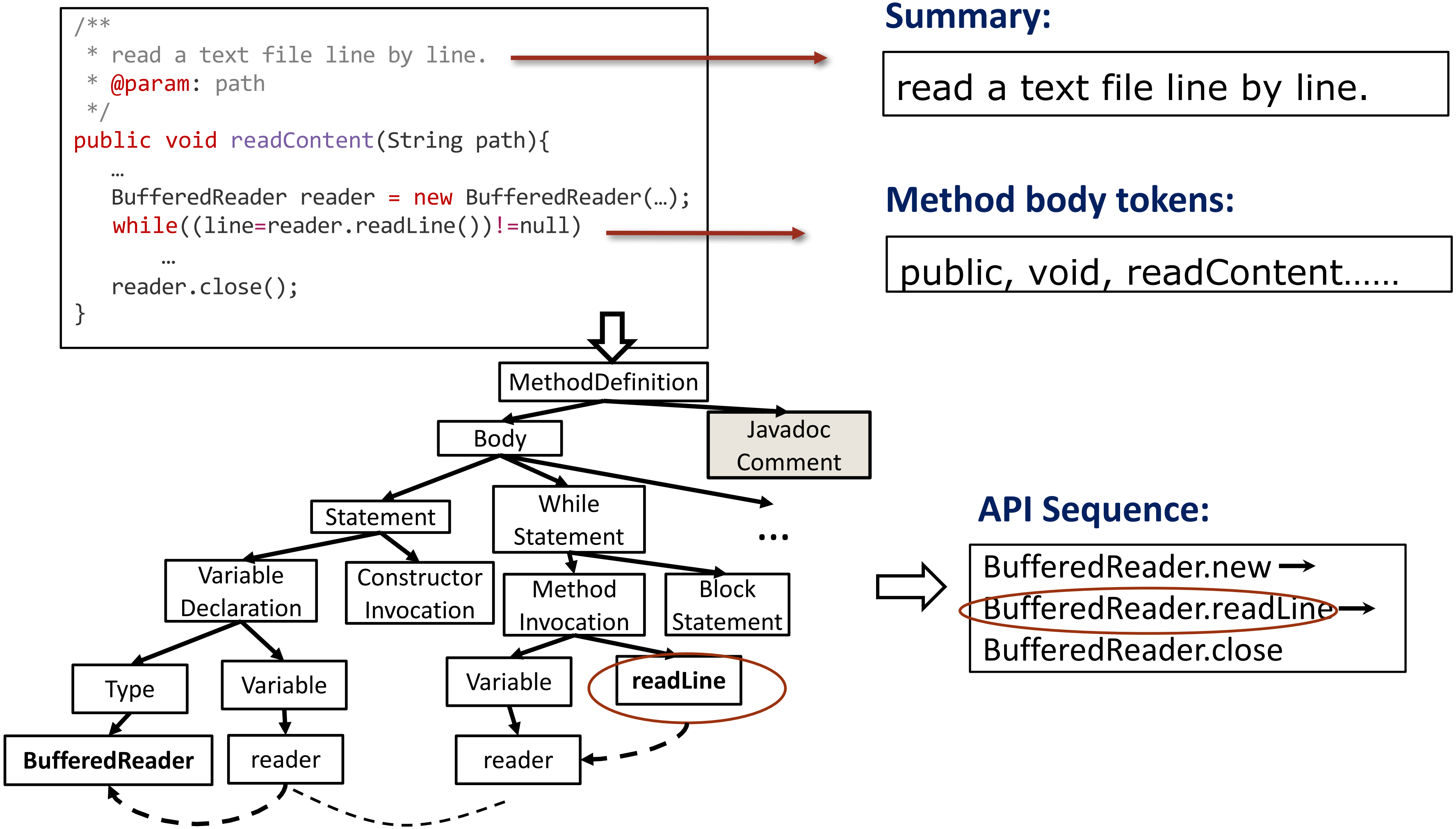
- **Apply the API knowledge to assist the code summarization task**
  - Learn the mappings between the API knowledge and natural language descriptions
  - Transfer the knowledge into a different but related task

- **Future Work**
  - Apply the API knowledge into other tasks
  - Mining more latent knowledge in the source code

# Q&A

Thanks

# Preprocessing Dataset



```
/**
 * read a text file line by line.
 * @param: path
 */
public void readContent(String path){
   …
   BufferedReader reader = new BufferedReader(…);
   while((line=reader.readLine())!=null)
       …
   reader.close();
}
```

**Summary:**

read a text file line by line.

**Method body tokens:**

public, void, readContent……

**API Sequence:**

BufferedReader.new →
BufferedReader.readLine →
BufferedReader.close

Gu X, Zhang H, Zhang D, et al. Deep API learning[C]//Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016: 631-642.