Impact of Evaluation Methodologies on Code Summarization

Pengyu Nie, Jiyang Zhang, Junyi Jessy Li, Raymond J. Mooney, Milos Gligoric

The University of Texas at Austin

ACL 2022
ML Models for Code Summarization

- Advances in ML/NLP are helping developers to produce and maintain software-related artifacts, for example code summarization

```java
public static String selectText(XPathExpression expr, Node context) {
    try {
        return (String)expr.evaluate(context, XPathConstants.STRING);
    } catch (XPathExpressionException e) {
        throw new XmlException(e);
    }
}
```

```java
int ?(String str, char ch) {
    int num = 0;
    int index = -1;
    do {
        index = str.indexOf(ch, index + 1);
        if (index >= 0) num++;
    } while (index >= 0);
    return num;
}
```

/** Evaluates the xpath expression as text. */

- Comment generation

- Count occurrences

- Method naming
Evaluation Methodologies of Code Summarization ML Models

• extract a dataset of (code, comment) samples
• split the dataset into training, validation, test sets
• train on training + validation sets
• report automatic metrics on test set
Temporal Relations Not Explicitly Modeled in Prior Work

1. written in 2018

```java
/** Returns the total number of connections in the pool. */
public synchronized int connectionCount() {
    return connections.size();
}
```

2. written in 2019

```java
/** Returns the number of idle connections in the pool. */
public synchronized int idleConnectionCount() {
    int total = 0;
    for (RealConnection connection : connections) {
        if (connection.allocations.isEmpty()) total++;
    }
    return total;
}
```

- **Training:**
  - `1` (a)
  - `2` (b)
  - `...`

- **Validation:**
  - `1` (a)
  - `2` (b)
  - `...`

- **Test:**
  - `1` (a)
  - `2` (b)
  - `...`

**Evaluation Methodologies in Prior Work**
- Mixed-project using future to predict past
- Cross-project too strong assumption

**Use Case of ML Model**
- Use model on `2` (b)
- Train model on `1` (a)

**Misunderstanding** if a model might be useful / not useful once adopted
Our Contributions

• Study the **evaluation methodologies** of 18 recent papers on code summarization
  • found **two** commonly used evaluation methodologies: mixed-project and cross-project
  • define **two use cases** that could be evaluated by these methodologies

• Define a more **practical use case**: continuous-mode

• Propose an appropriate evaluation methodology for this use case: time-segmented

• Experiment several existing ML models using the three methodologies

<table>
<thead>
<tr>
<th>evaluation methodology</th>
<th>use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>mixed-project (used by 15/18)</td>
<td>in-project batch-mode</td>
</tr>
<tr>
<td>cross-project (used by 4/18)</td>
<td>cross-project batch-mode</td>
</tr>
<tr>
<td>time-segmented (proposed)</td>
<td>continuous-mode</td>
</tr>
</tbody>
</table>
Outline

• **Evaluation methodologies and use cases**
  - mixed-project
  - cross-project
  - time-segmented
  - in-project batch-mode
  - cross-project batch-mode
  - continuous-mode

• **Experiments to study the impact of evaluation methodologies**
  - experiments setup
  - dataset
  - results and findings
Mixed-Project Evaluation Methodology & In-Project Batch-Mode Use Case

- Used in prior work
- Randomly shuffle the samples and split them into training, validation, and test sets

![Diagram showing project samples and ML model process]

- Alice's project
- Other projects
- ML model
- Training
- Validation
- Test

1. Alice's project
2. Other projects
3. ML model
4. Train
5. Apply
6. Output
Cross-Project Evaluation Methodology & Cross-Project Batch-Mode Use Case

- Used in prior work
- Randomly shuffle the projects and split them into training, validation, and test sets

Alice’s project

Alice

ML model

train

other projects

- Used in prior work
- Randomly shuffle the projects and split them into training, validation, and test sets

Alice’s project

Alice

ML model

train

other projects

- Used in prior work
- Randomly shuffle the projects and split them into training, validation, and test sets
Limitation of Batch-Mode Use Cases

Usually happen only once in the lifecycle of a project

use ML model at $\tau^{-1}$ in-project batch-mode

use ML model at $\tau$ temporal relations among samples exists
Continuous-Mode Use Case

Alice

write comments for each method around the same time as the method itself

download the latest model

apply it on each newly written method

\( \tau^{-1} \)

\( \tau \)

time

::

ML model

other projects

Alice’s project

1

2

3

4

5

6

7

8

9

ML model

other projects

Alice’s project

1

2

3

4

5

6

7

8

9

Download the latest model

Apply it on each newly written method
Time-Segmented Evaluation Methodology

• Not used in prior work on developing new ML models for code summarization

• Split samples in a time-aware method
  • assign samples before $\tau^{-2}$ to training set
  • assign samples after $\tau^{-2}$ and before $\tau^{-1}$ to validation set
  • assign samples after $\tau^{-1}$ and before $\tau$ to test set
Outline

• Evaluation methodologies and use cases
  - mixed-project
  - in-project batch-mode
  - cross-project
  - cross-project batch-mode
  - time-segmented
  - continuous-mode

• Experiments to study the impact of evaluation methodologies
  • experiments setup
  • dataset
  • results and findings
## Experiments Setup

<table>
<thead>
<tr>
<th>Task</th>
<th>comment generation</th>
<th>method naming</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Models</strong></td>
<td>DeepComHybrid</td>
<td>Code2Vec</td>
</tr>
<tr>
<td></td>
<td>Transformer</td>
<td>Alon et al. POPL’19</td>
</tr>
<tr>
<td></td>
<td>Seq2Seq</td>
<td>Code2Seq</td>
</tr>
<tr>
<td></td>
<td>Hu et al. ESE’20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ahmad et al. ACL’20</td>
<td></td>
</tr>
<tr>
<td><strong>Metrics</strong></td>
<td>BLEU</td>
<td>Precision</td>
</tr>
<tr>
<td></td>
<td>METEOR</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td>ROUGE-L</td>
<td>F1</td>
</tr>
<tr>
<td></td>
<td>EM (exact match)</td>
<td>EM (exact match)</td>
</tr>
</tbody>
</table>
Dataset

- 77,745 (code, comment) with timestamps from 160 popular Java projects on GitHub
- Given a dataset of (code, comment) with timestamps, split it to get
  - training (Train), validation (Val), and standard test (TestS) sets for each methodology
  - common test (TestC) set to compare each pair of methodologies

<table>
<thead>
<tr>
<th>Task</th>
<th>Train</th>
<th>Val</th>
<th>TestS</th>
<th>TestC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>MP 50,879</td>
<td>7,569</td>
<td>14,956</td>
<td>MP ∩ CP 3,362</td>
</tr>
<tr>
<td>Generation</td>
<td>CP 50,879</td>
<td>8,938</td>
<td>15,661</td>
<td>MP ∩ T 2,013</td>
</tr>
<tr>
<td>Method Naming</td>
<td>T 50,879</td>
<td>11,312</td>
<td>9,870</td>
<td>CP ∩ T 2,220</td>
</tr>
</tbody>
</table>

- MP = mixed-project
- CP = cross-project
- T = time-segmented

Split

2019.1.1
2020.1.1
2021.1.1
70%
10%
20%
Results and Findings (1/4)

• Different methodologies may lead to conflicting evaluation results
  • Code2Vec is better than Code2Seq under the mixed-project and time-segmented methodologies, but is worse under the cross-project methodology
• Different methodologies may lead to conflicting evaluation results
  • Transformer is statistically significantly better than Seq2Seq under the time-segmented methodology, but not under the cross-project methodology

*no significant difference between the models
Results and Findings (3/4)

• Evaluation results from prior work do not represent the ML models’ performance in the continuous-mode use case
  • Results under the mixed-project methodology are inflated
  • Results under the cross-project methodology may be an under-estimation

---

task: method naming
metric: F1

![Bar chart comparing Code2Vec and Code2Seq for different methodologies.](image-url)
Results and Findings (4/4)

• Evaluation results from prior work do not represent the ML models’ performance in the continuous-mode use case
  • Results under the mixed-project methodology are inflated
  • Results under the cross-project methodology may be an under-estimation

<table>
<thead>
<tr>
<th>Task: Comment Generation</th>
<th>Metric: BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>56.3, 56.1</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>65.6, 53.3</td>
</tr>
<tr>
<td>DeepComHybrid</td>
<td>61.7, 53.2</td>
</tr>
</tbody>
</table>

| Transformer              | 52.6, 45.6   |
| Seq2Seq                  | 53.2, 14.3   |
| DeepComHybrid            | 61.7, 13.4   |

- time-segmented
- mixed-project
- cross-project
Conclusions

• We need to more diligently choose evaluation methodology and report results of ML models according to the intended use cases.
• Time-segmented evaluation methodology should be adopted in the evaluation of ML models for code summarization.

Data and code: https://github.com/EngineeringSoftware/time-segmented-evaluation

Pengyu Nie <pynie@utexas.edu>
backup slides
Evaluation Methodologies of ML Models

• Evaluation paradigm based on **automatic metrics**:
  • split a dataset of (code, comment) samples into training, validation, and test sets
  • train ML models on training + validation sets
  • report automatic metrics (e.g., BLEU, F1) on test set

• What is the **intended use cases** of the ML model?
• **How to split the dataset**, such that the evaluation results represent the ML model’s performance in the intended use cases?
• In the context of code summarization: should we consider the timestamps of code and comments?
Developers iteratively add/edit code and comments
  • the style of newer code and comments written can be affected by older code and comments

Temporal relations among samples are not explicitly modeled in the evaluation of prior work
  • can lead to inflated values for automatic metrics
  • can lead to misunderstanding if a model might be useful once adopted
Experiments Setup

• Dataset
  • (code, comment) with timestamps from popular Java projects on GitHub using English for summaries
  • collected samples before $\tau = 2021.1.1$
    time-segmented on $\tau^{-2} = 2019.1.1$ and $\tau^{-1} = 2020.1.1$
  • splitting ratios for in-project and cross-project: 70%, 10%, 20%

• Models
  • comment generation: DeepComHybrid, Transformer, Seq2Seq
  • method naming: Code2Vec, Code2Seq

• Automatic metrics
  • comment generation: BLEU, METEOR, ROUGE-L, EM (exact match)
  • method naming: precision, recall, F1, EM (exact match)

• Run each model 3 times & perform statistical significance tests
• Depending on the methodology, one model can perform better or worse than another
• Depending on the methodology, the differences between models may or may not be observable.
Results and Findings (3/3)

- Results under the **mixed-project** methodology are **inflated**
- Results under the **cross-project** methodology may be an **under-estimation** of the more realistic continuous-mode use case
Conclusions

• We need to more *diligently choose evaluation methodology* and report results of ML models according to the *intended use cases*

• **Time-segmented** evaluation methodology should be adopted in the evaluation of ML models for code summarization

• Misuse of evaluation methodologies can lead to *inflated values* for automatic metrics and *misunderstanding* if a model might be *useful* once adopted

---

data and code: [https://github.com/EngineeringSoftware/time-segmented-evaluation](https://github.com/EngineeringSoftware/time-segmented-evaluation)

Pengyu Nie <pynie@utexas.edu>
Given a dataset of (code, comment) with timestamps, split it to get:

- training (Train), validation (Val), and standard test (TestS) sets for each methodology
- common test (TestC) set to compare each pair of methodologies
Application of Methodologies: Step 1/6

• Given a dataset of (code, comment) with timestamps, split it to get
  • training (Train), validation (Val), and standard test (TestS) sets for each methodology
  • common test (TestC) set to compare each pair of methodologies

1. time-segment samples in each project
• Given a dataset of (code, comment) with timestamps, split it to get
  • training (Train), validation (Val), and standard test (TestS) sets for each methodology
  • common test (TestC) set to compare each pair of methodologies

1. time-segment samples in each project
2. perform in-project split

splitting ratios

$$r_x = 70\%$$
$$r_y = 10\%$$
$$r_z = 20\%$$
Application of Methodologies: Step 3/6

- Given a dataset of (code, comment) with timestamps, split it to get
  - training (Train), validation (Val), and standard test (TestS) sets for each methodology
  - common test (TestC) set to compare each pair of methodologies

1. time-segment samples in each project
2. perform in-project split
3. perform cross-project split

- splitting ratios
  - \( r_x = 70\% 
  - r_y = 10\%
  - r_z = 20\% 

Application of Methodologies: Step 4/6

- Given a dataset of (code, comment) with timestamps, split it to get
  - training (Train), validation (Val), and standard test (TestS) sets for each methodology
  - common test (TestC) set to compare each pair of methodologies

1. time-segment samples in each project
2. perform in-project split
3. perform cross-project split
4. group into Train, Val, and TestS sets

MP = mixed-project    CP = cross-project    T = time-segmented
Application of Methodologies: Step 5/6

- Given a dataset of (code, comment) with timestamps, split it to get:
  - training (Train), validation (Val), and standard test (TestS) sets for each methodology
  - common test (TestC) set to compare each pair of methodologies

1. time-segment samples in each project
2. perform in-project split
3. perform cross-project split
4. group into Train, Val, and TestS sets
5. intersect TestS sets to get TestC sets

MP = mixed-project  CP = cross-project  T = time-segmented
• Given a dataset of (code, comment) with timestamps, split it to get
  • training (Train), validation (Val), and standard test (TestS) sets for each methodology
  • common test (TestC) set to compare each pair of methodologies

1. time-segment samples in each project
2. perform in-project split
3. perform cross-project split
4. group into Train, Val, and TestS sets
5. intersect TestS sets to get TestC sets
6. perform post-processing

MP = mixed-project    CP = cross-project    T = time-segmented
• downsample Train sets to the same size
• remove duplicates from Val, TestS, and TestC sets
Mixed-Project Evaluation Methodology

Randomly shuffle the **samples** and split them into training, validation, and test sets.
In-Project Batch-Mode Use Case

- **Alice**
  - write comments for only a part of methods
  - time to add documentations, with ML model

\[ \tau \]

- **ML model**
  - train
  - apply
  - other projects
  - project 1, project 2, project 3, ..., project n-1, project n

- **Alice’s project**
  - 1, 2, 3, 4, 5, 6
Randomly shuffle the projects and split them into training, validation, and test sets.
Cross-Project Batch-Mode Use Case

- Alice’s project
  - do not write comments for any method
  - time to add documentations, with ML model

- Other projects

\[ \tau \]

time
Not Considering Temporal Relations During Evaluation

- **Temporal relations** among samples are not explicitly modeled
- Can lead to **misunderstanding** if a model might be useful / not useful once adopted
<table>
<thead>
<tr>
<th>Task</th>
<th>comment generation</th>
<th>method naming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models</td>
<td>DeepComHybrid, Transformer, Seq2Seq, Code2Vec, Code2Seq</td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td>BLEU, METEOR, ROUGE-L, EM (exact match)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Precision, Recall, F1, EM (exact match)</td>
<td></td>
</tr>
<tr>
<td>References</td>
<td>Hu et al. ESE'20, Ahmad et al. ACL'20, Alon et al. POPL'19, Alon et al. ICLR'19</td>
<td></td>
</tr>
</tbody>
</table>