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

partially supported by Google NSF TEXAS The University of Texas at Austin

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

comment generation

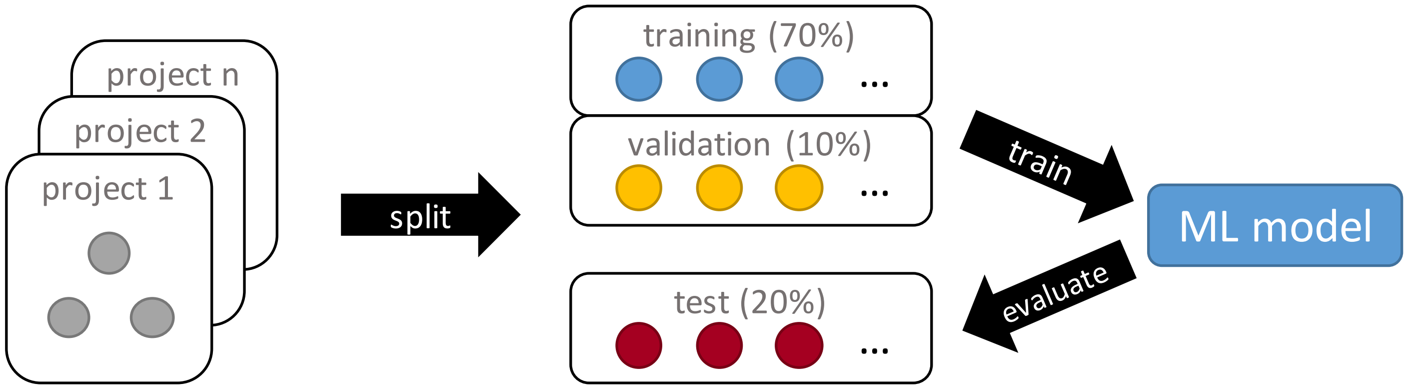
```java
int ?(String str, char ch) {
  int num = 0;
  int index = -1;
  do {
    index = str.indexOf(ch, index + 1);
    if (index >= 0) num++;
  } while (index >= 0);
  return num;
}
```

```
countOccurrences
```

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



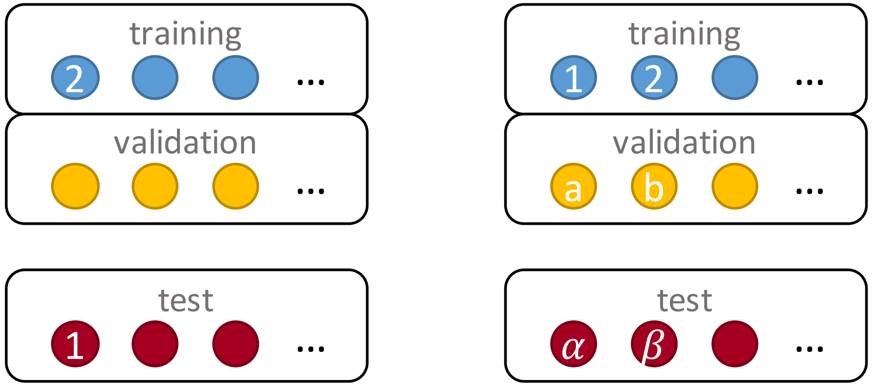
Temporal Relations Not Explicitly Modeled in Prior Work

1 written in 2018

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

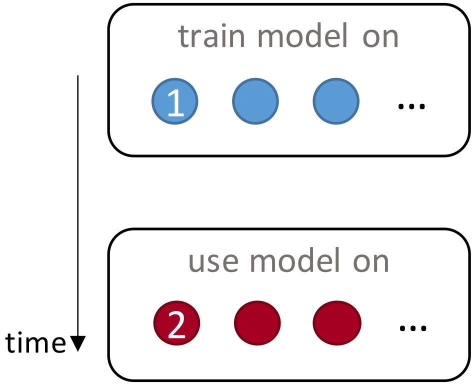
2 written in 2019

```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;
}
```



mixed-project
using future to predict past

cross-project
too strong assumption



evaluation methodologies in prior work ≠ use case of ML model

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

evaluation methodology	**use case**
mixed-project (used by 15/18)	in-project batch-mode
cross-project (used by 4/18)	cross-project batch-mode
time-segmented (proposed)	continuous-mode

Outline

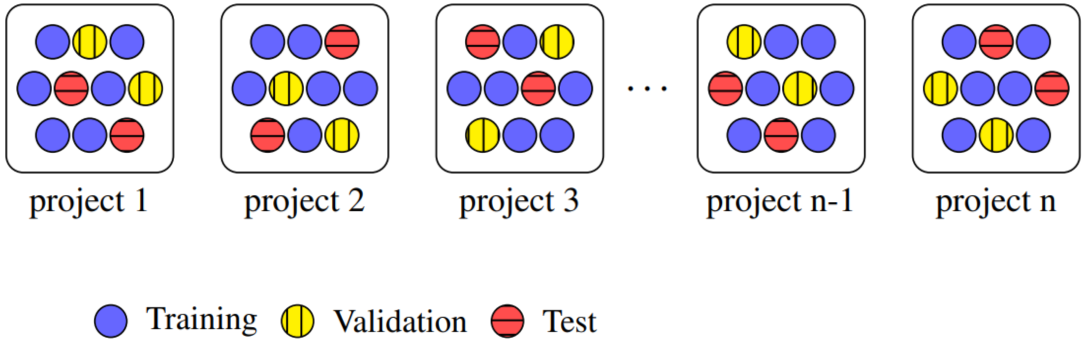
- Evaluation methodologies and use cases

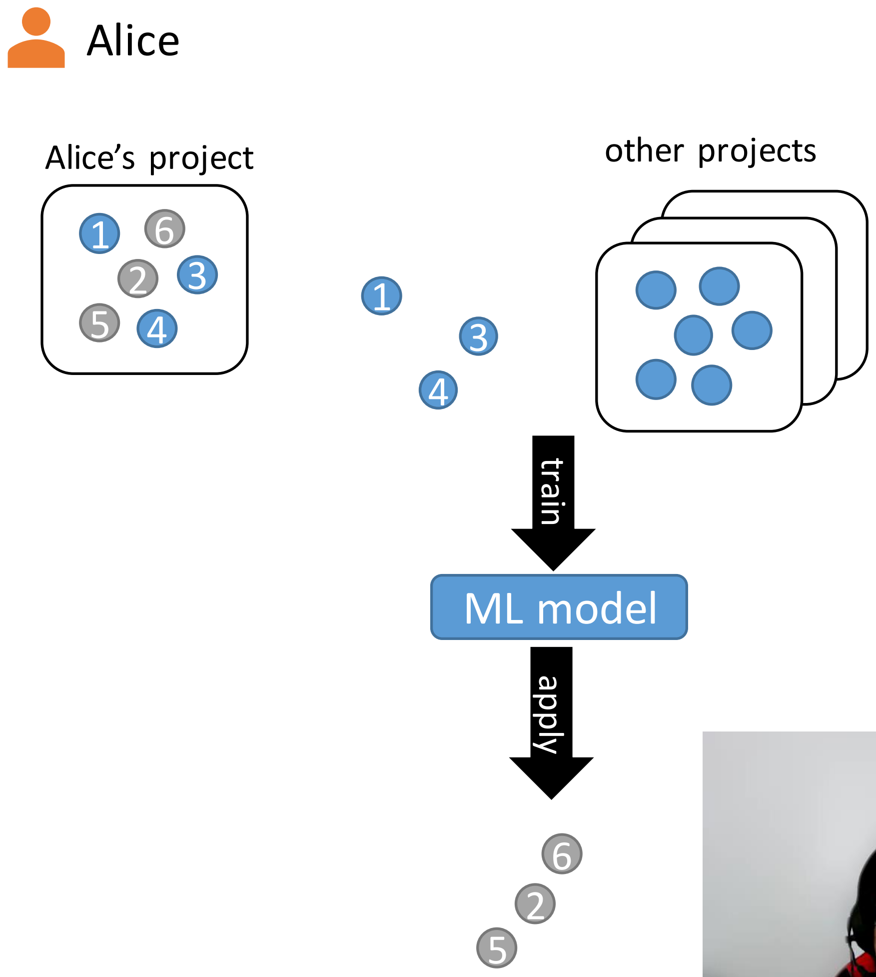
mixed-project	in-project batch-mode
cross-project	cross-project batch-mode
time-segmented	continuous-mode

- Experiments to study the impact of evaluation methodologies
  - experiments setup
  - dataset
  - results and findings

Mixed-Project Evaluation Methodology & In-Project Batch-Mode Use Case

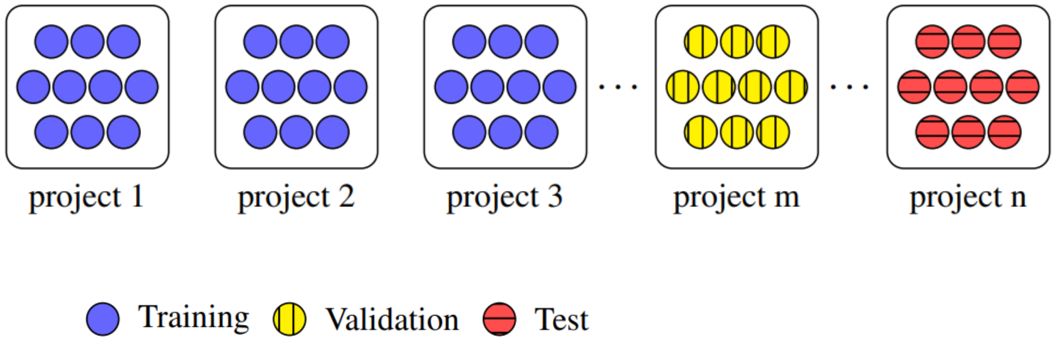
- Used in prior work
- Randomly shuffle the samples and split them into training, validation, and test sets

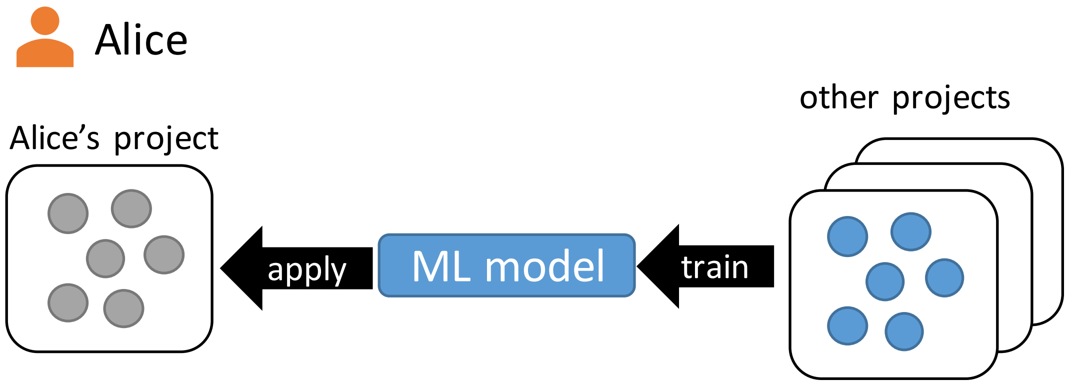


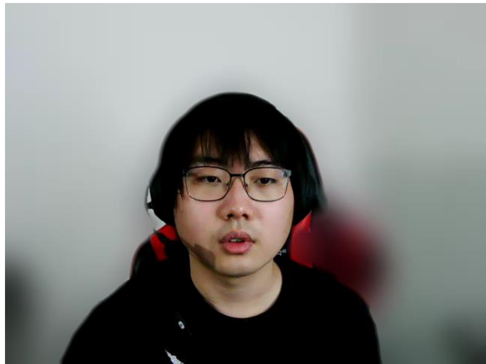


Cross-Project Evaluation Methodology & Cross-Project Batch-Mode Use Case

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

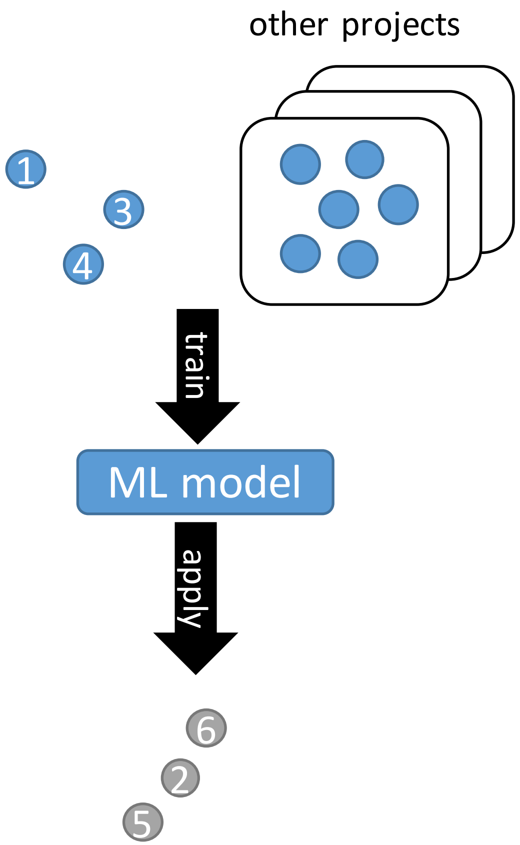




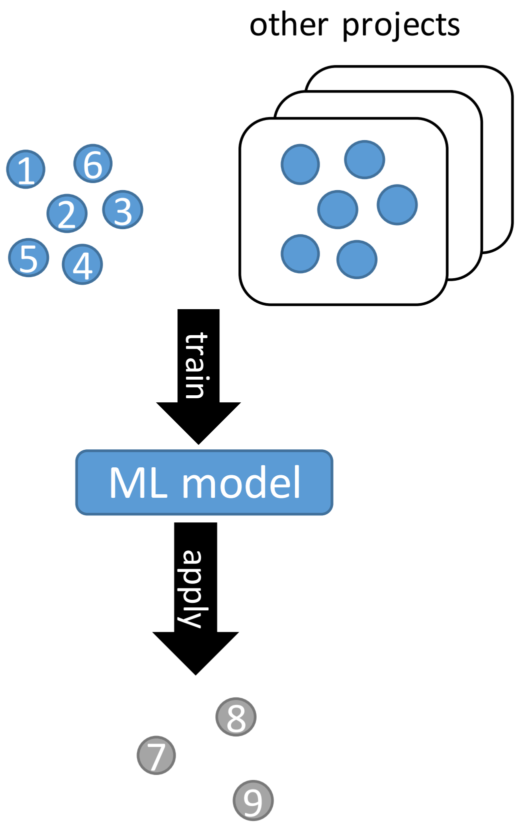


Limitation of Batch-Mode Use Cases

Usually happen only once in the lifecycle of a project



use ML model at τ^{-1}
in-project batch-mode



use ML model at τ
temporal relations among samples exists

Continuous-Mode Use Case

Alice

τ^{-1}

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

download the latest model

apply it on each newly written method

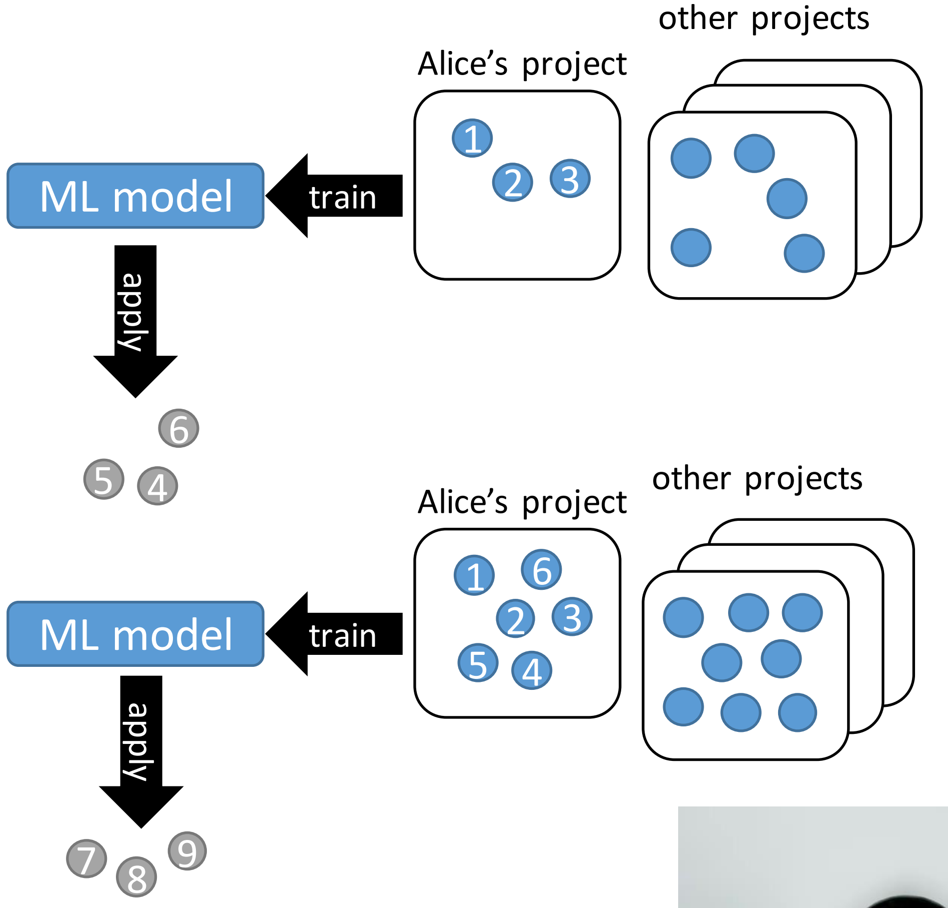
τ

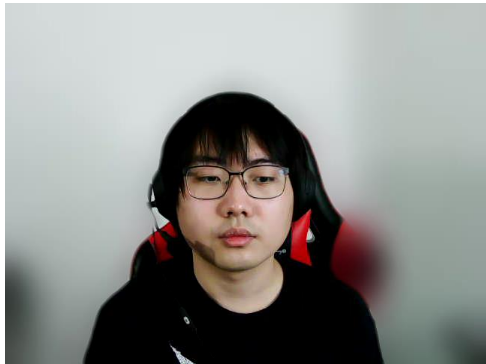
download the latest model

apply it on each newly written method

⋮

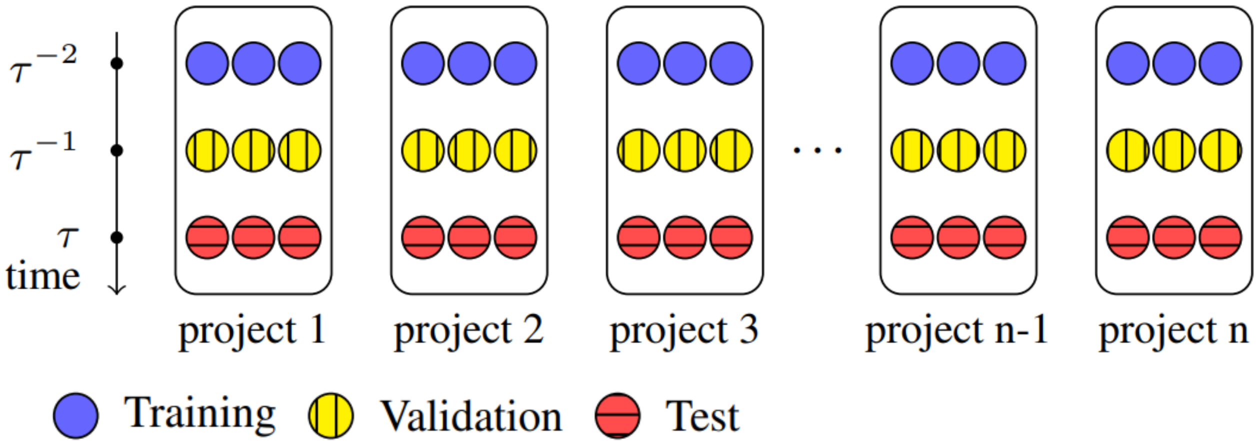
time

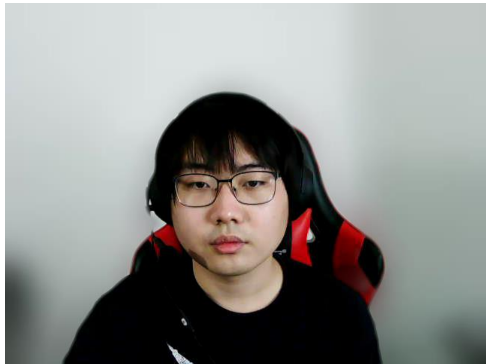




Time-Segmented Evaluation Methodology

- Not used in prior work on developing new ML models for code summarization
- Split samples in a time-aware method
  - assign samples before τ^{-2} to training set
  - assign samples after τ^{-2} and before τ^{-1} to validation set
  - assign samples after τ^{-1} and before τ 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

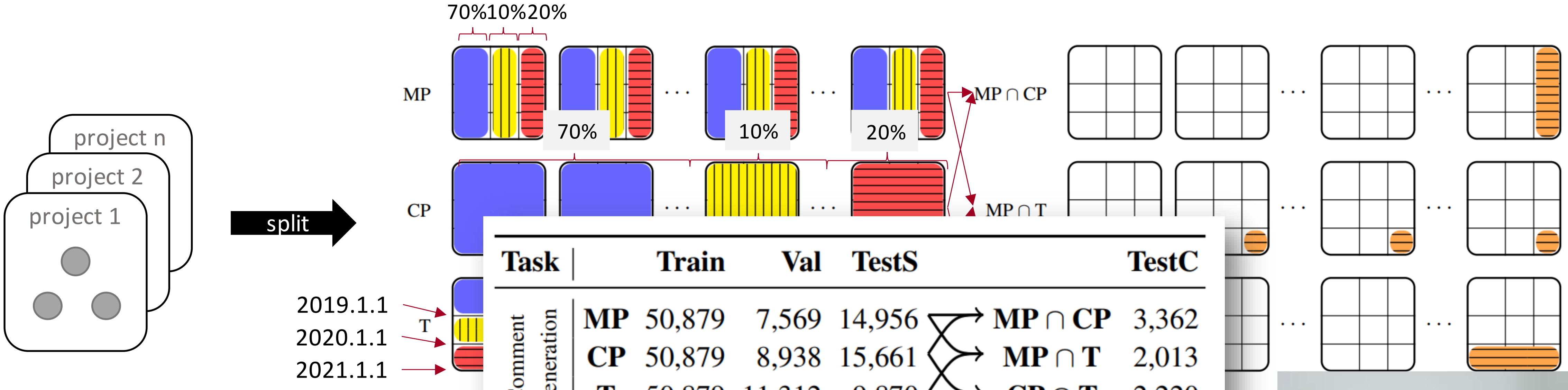
- Experiments to study the impact of evaluation methodologies
  - experiments setup
  - dataset
  - results and findings

Experiments Setup

Task	comment generation		method naming	
Models	DeepComHybrid	Hu et al. ESE'20	Code2Vec	Alon et al. POPL'19
	Transformer Seq2Seq	Ahmad et al. ACL'20	Code2Seq	Alon et al. ICLR'19
Metrics	BLEU METEOR ROUGE-L EM (exact match)		Precision Recall F1 EM (exact match)	

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



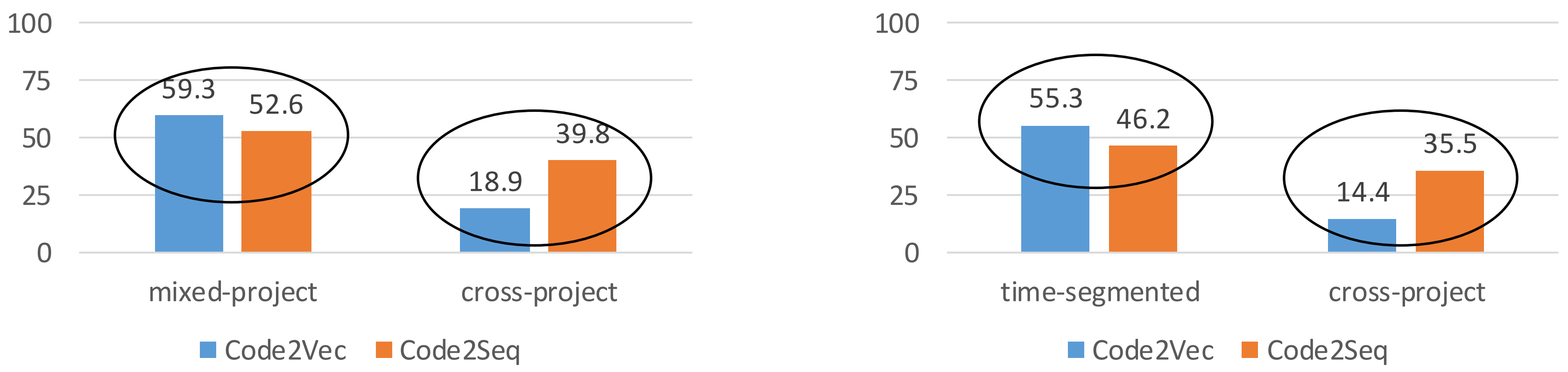
Task		Train	Val	TestS		TestC
Comment Generation	MP	50,879	7,569	14,956	MP ∩ CP	3,362
	CP	50,879	8,938	15,661	MP ∩ T	2,013
	T	50,879	11,312	9,870	CP ∩ T	2,220
Method Naming	MP	50,879	7,523	14,796	MP ∩ CP	3,344
	CP	50,879	8,811	15,332	MP ∩ T	2,011
	T	50,879	11,223	9,807	CP ∩ T	2,211

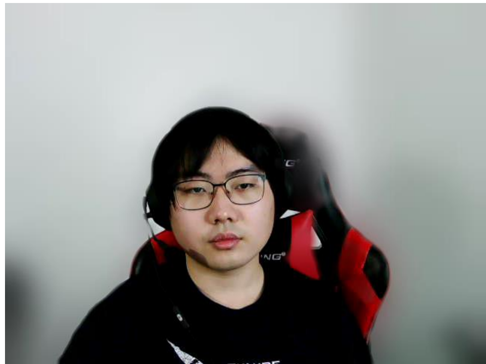
MP = mix

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

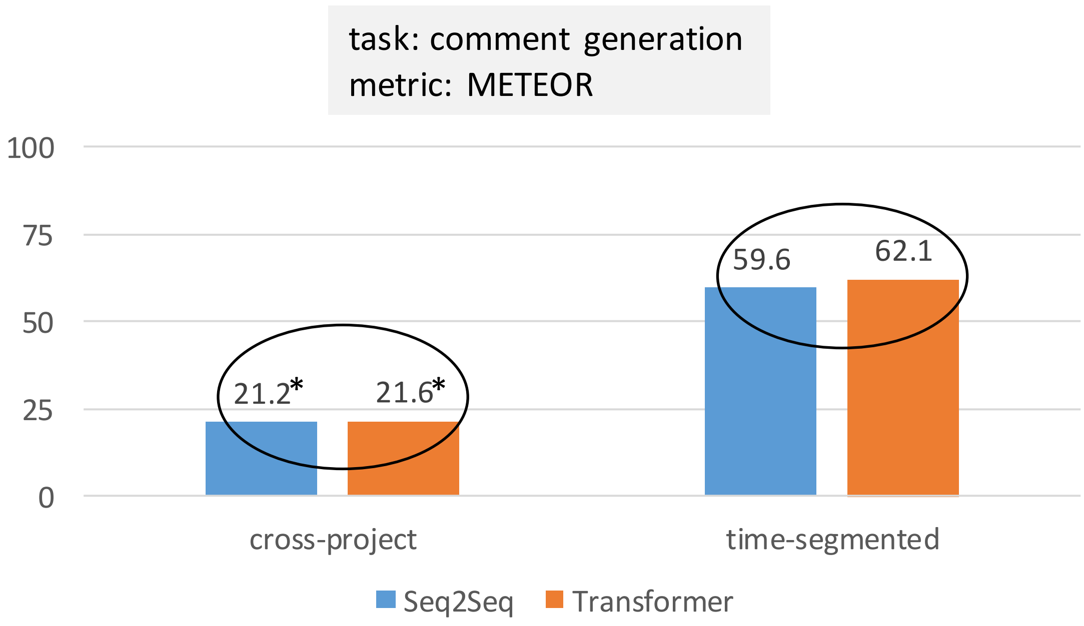
task: method naming
metric: F1



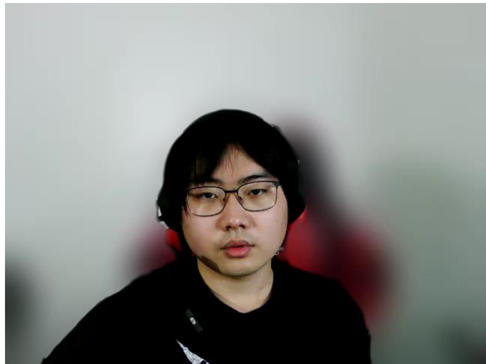


Results and Findings (2/4)

- 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



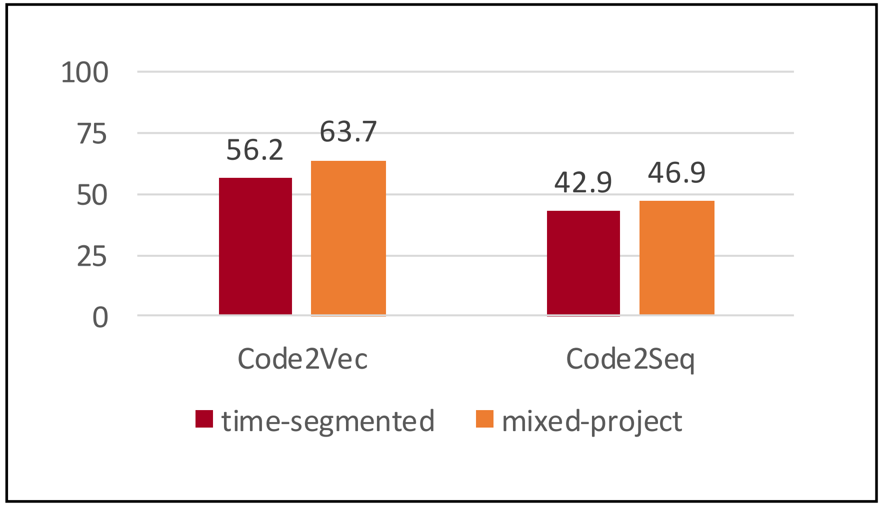
*no significant difference between the models

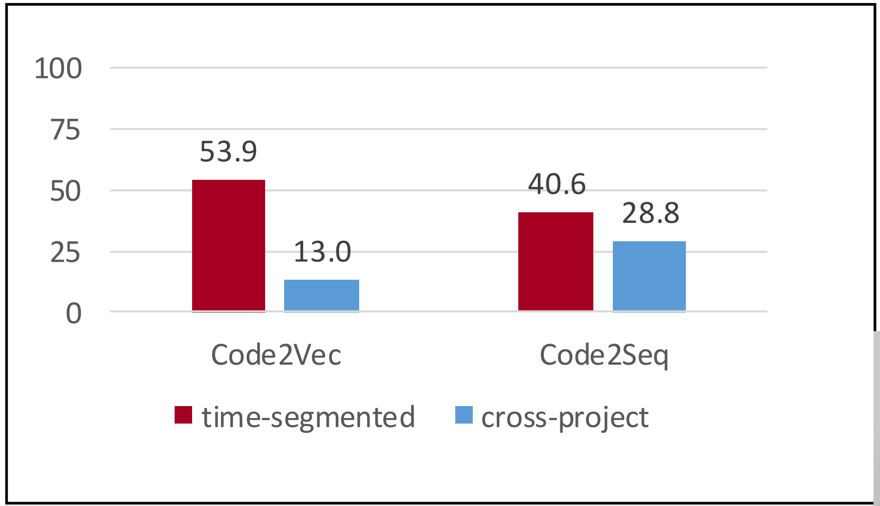


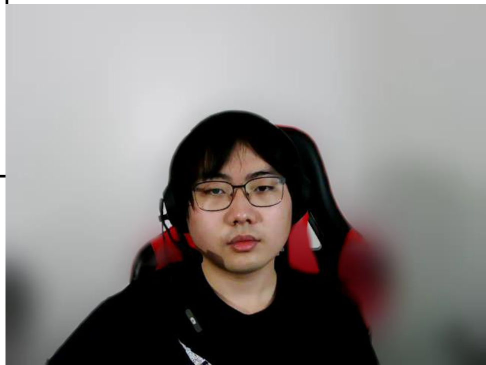
Results and Findings (3/4)

- 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



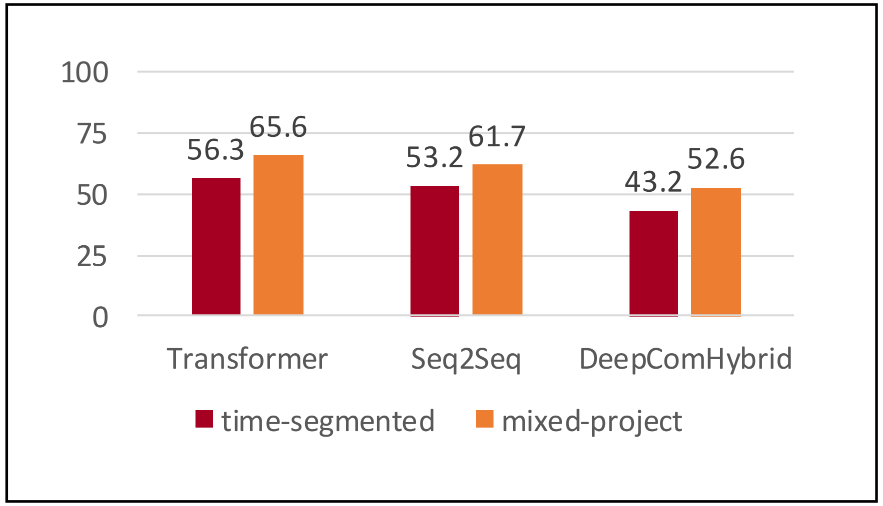


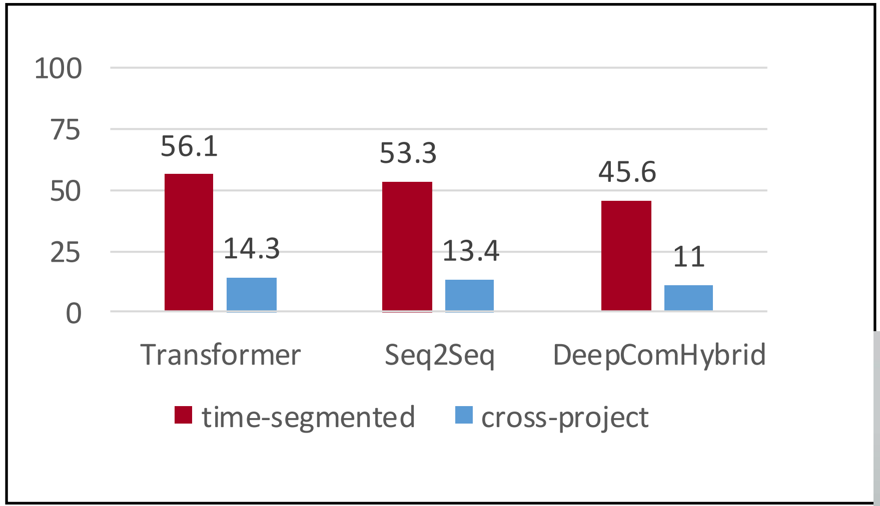


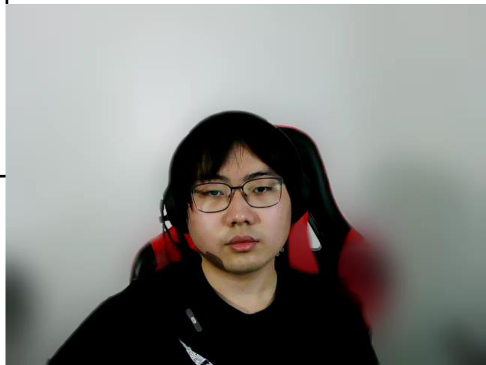
Results and Findings (4/4)

- 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

task: comment generation
metric: BLEU







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
preprint: https://arxiv.org/abs/2108.09619

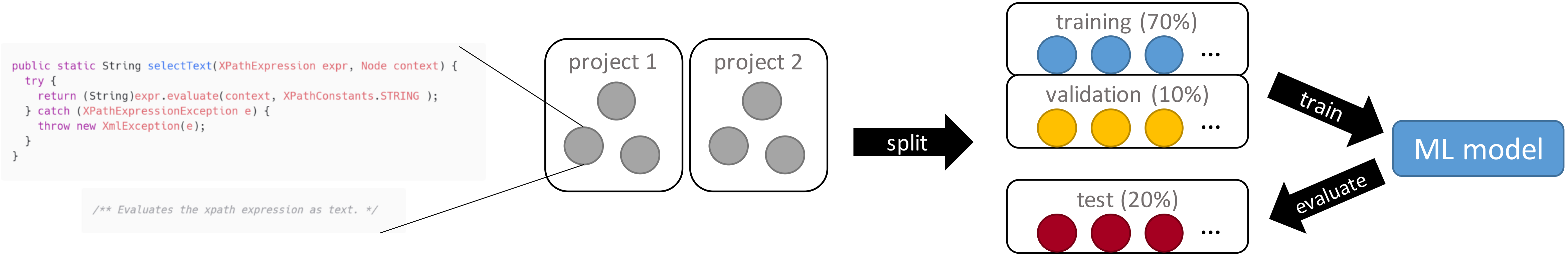
Pengyu Nie <pynie@utexas.edu>

TEXAS
The University of Texas at Austin

backup slides

Evaluation Methodologies of ML Models

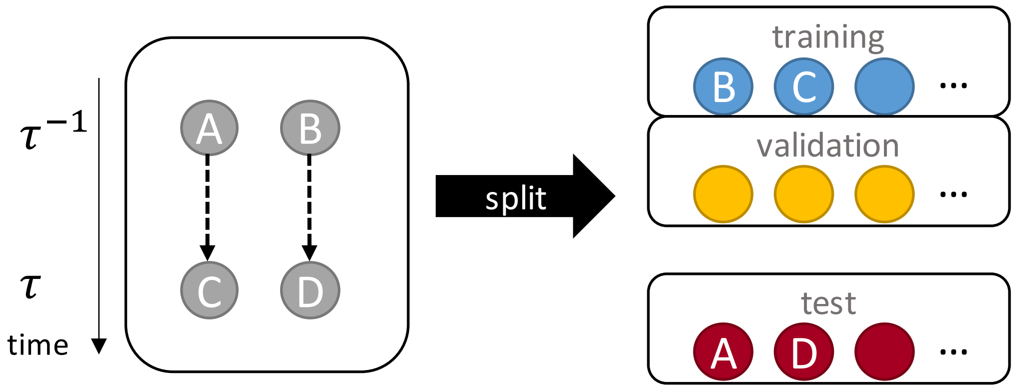
- Evaluation paradigm based on automatic metrics:
  - split a dataset of (code, comment) samples into training, validation, and test sets
  - train ML models on training + validation sets
  - report automatic metrics (e.g., BLEU, F1) on test set



- 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?

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

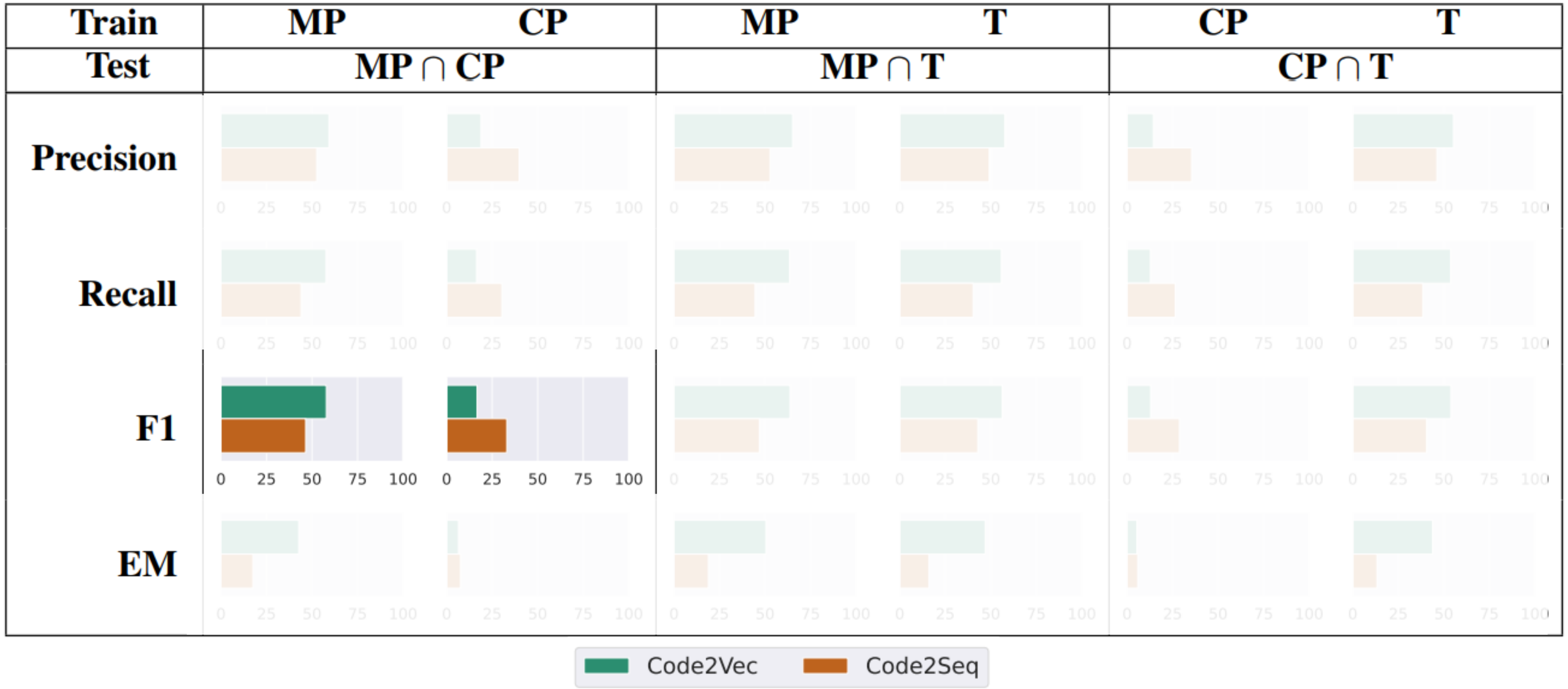


Experiments Setup

- 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

Task		Train	Val	TestS	TestC	
Comment Generation	**MP**	50,879	7,569	14,956	**MP** $\cap$ **CP**	3,362
	CP	50,879	8,938	15,661	**MP** $\cap$ **T**	2,013
	T	50,879	11,312	9,870	**CP** $\cap$ **T**	2,220
Method Naming	**MP**	50,879	7,523	14,796	**MP** $\cap$ **CP**	3,344
	CP	50,879	8,811	15,332	**MP** $\cap$ **T**	2,011
	T	50,879	11,223	9,807	**CP** $\cap$ **T**	2,211

Results and Findings (1/3)



- Depending on the methodology, one model can perform better or worse than another

Results and Findings (2/3)

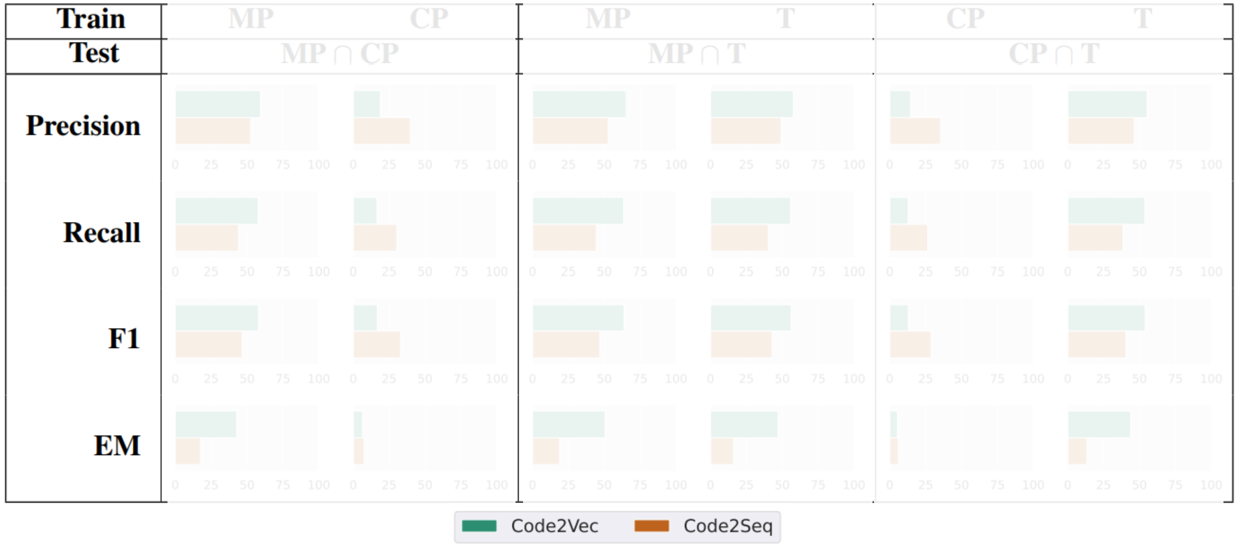
Train / Test	MP / MP ∩ CP	CP / MP ∩ CP	MP / MP ∩ T	T / MP ∩ T	CP / CP ∩ T	T / CP ∩ T
▼ BLEU [%]						
DeepComHybrid	45.0	11.4	$^{\beta}$52.6	43.2	11.0	45.6
Seq2Seq	53.8	$^{\alpha}$13.7	61.7	$^{\beta}$53.2	13.4	53.3
Transformer	**58.1**	$^{\alpha}$**14.1**	**65.6**	**56.3**	**14.3**	**56.1**
▼ METEOR [%]						
DeepComHybrid	52.9	18.3	$^{\delta}$59.3	50.0	18.2	52.0
Seq2Seq	62.0	$^{\gamma}$21.2	68.0	$^{\delta}$59.8	$^{\epsilon}$21.2	59.6
Transformer	**66.0**	$^{\gamma}$**21.4**	**71.5**	**62.7**	$^{\epsilon}$**21.6**	**62.1**
▼ ROUGE-L [%]						
DeepComHybrid	57.0	23.9	62.8	53.9	22.8	55.9
Seq2Seq	66.7	29.4	72.0	$^{\zeta}$64.1	28.7	64.3
Transformer	**70.1**	**30.4**	**74.9**	**66.6**	**30.0**	**66.4**
▼ EM [%]						
DeepComHybrid	30.2	$^{\eta\theta}$1.4	$^{\kappa}$39.5	31.0	$^{\lambda\mu}$1.3	35.4
Seq2Seq	37.3	$^{\eta\iota}$1.2	48.7	$^{\kappa}$41.0	$^{\lambda}$1.1	42.7
Transformer	**41.1**	$^{\theta\iota}$**1.7**	**52.3**	**44.2**	$^{\mu}$**1.7**	**45.8**

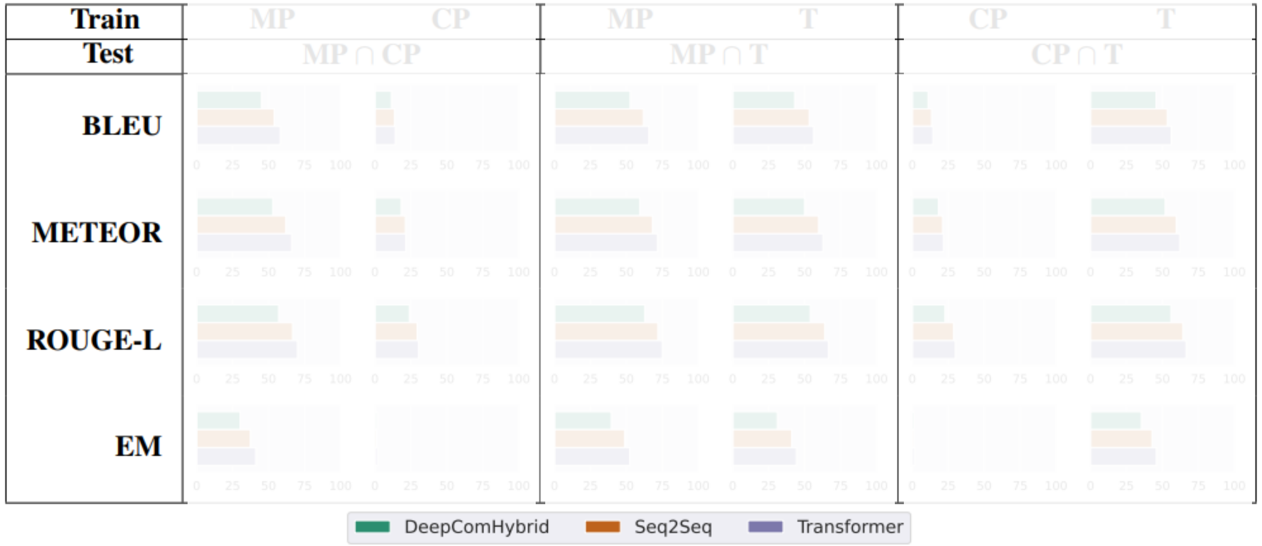
not significant

not significant

- 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

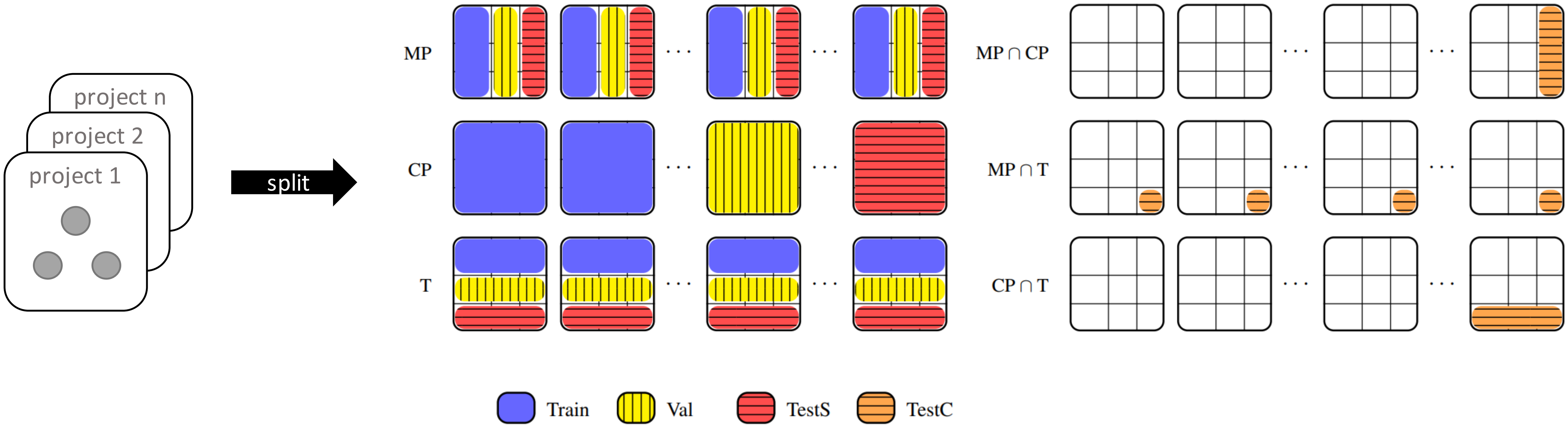
data and code: https://github.com/EngineeringSoftware/time-segmented-evaluation
preprint: https://arxiv.org/abs/2108.09619

Pengyu Nie <pynie@utexas.edu>

TEXAS
The University of Texas at Austin

Application of Methodologies: Goals

- 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

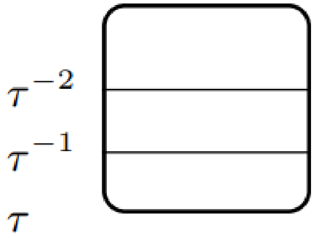


MP = mixed-project CP = cross-project T = time-segmented

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

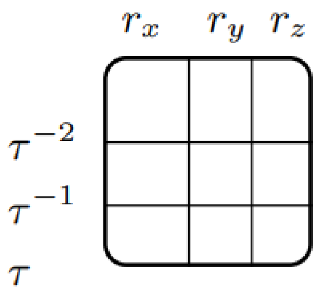


Application of Methodologies: Step 2/6

- 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



splitting ratios

$r_x = 70\%$

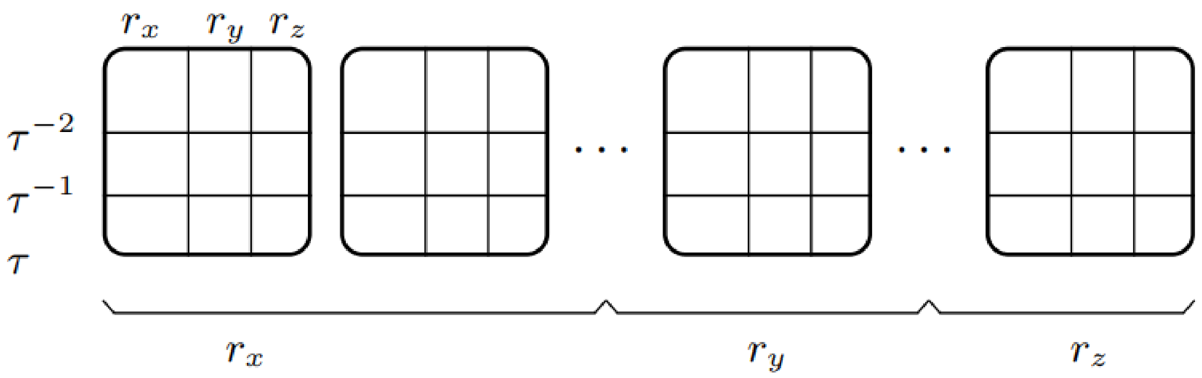
$r_y = 10\%$

$r_z = 20\%$

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

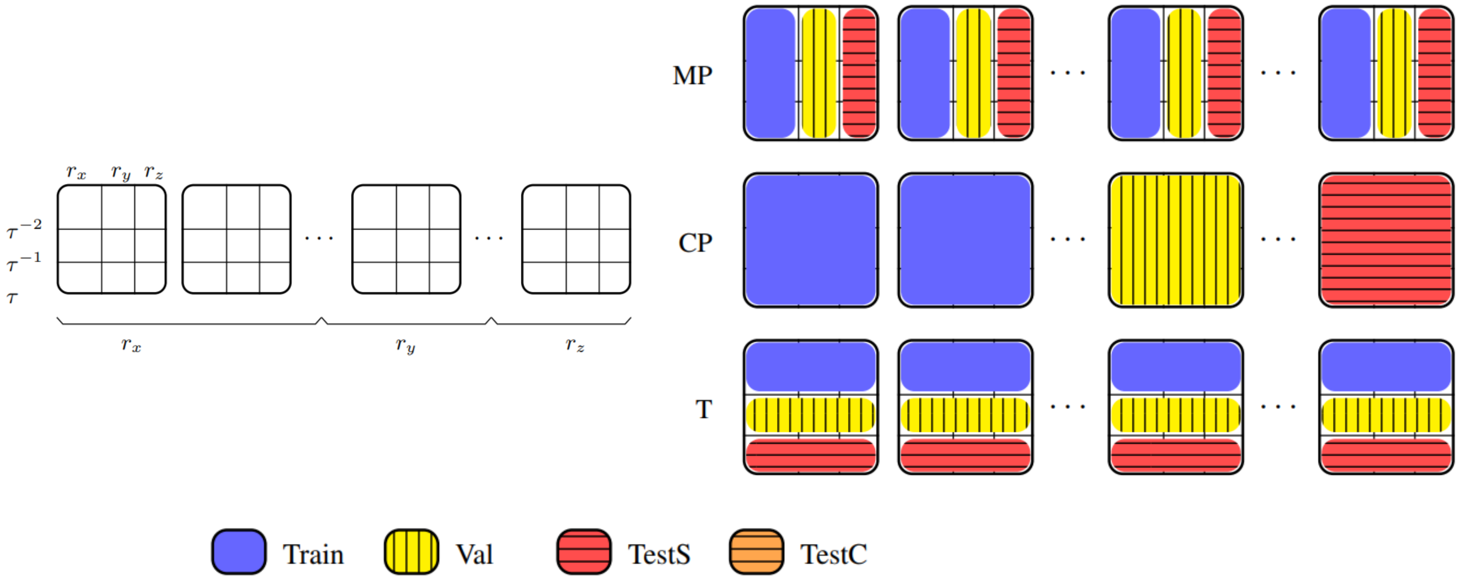


splitting ratios
$r_x = 70\%$
$r_y = 10\%$
$r_z = 20\%$

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

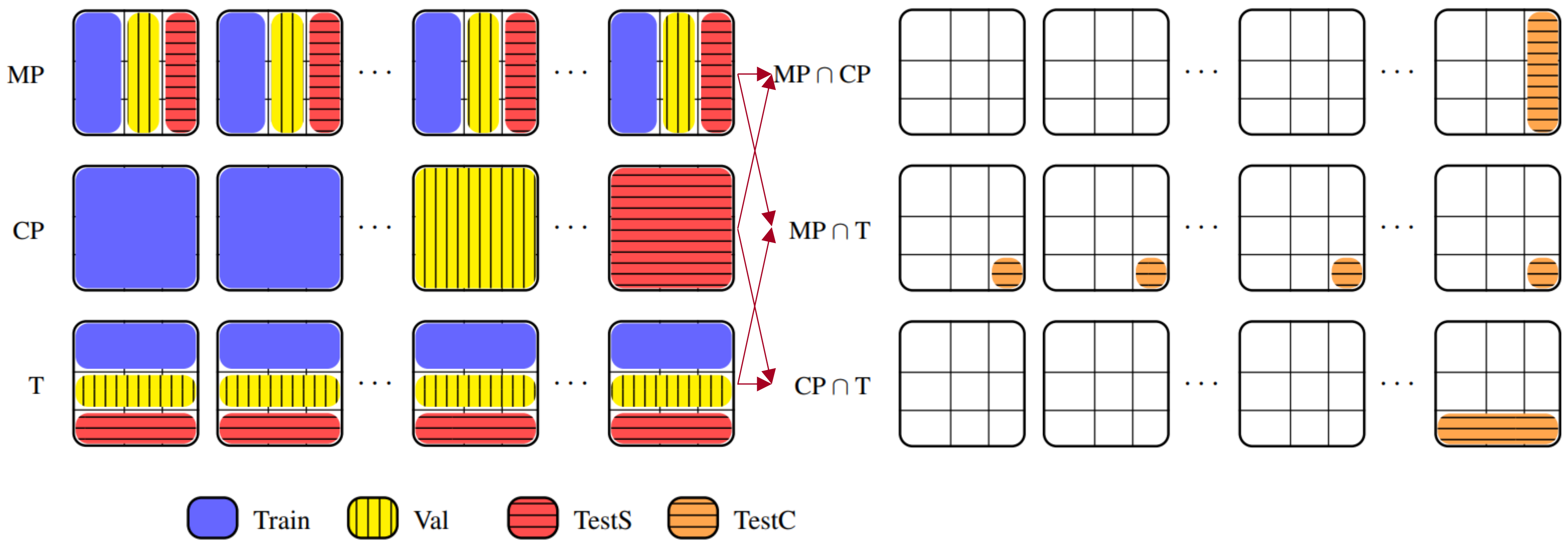


MP = mixed-project CP = cross-project T = time-segmented

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

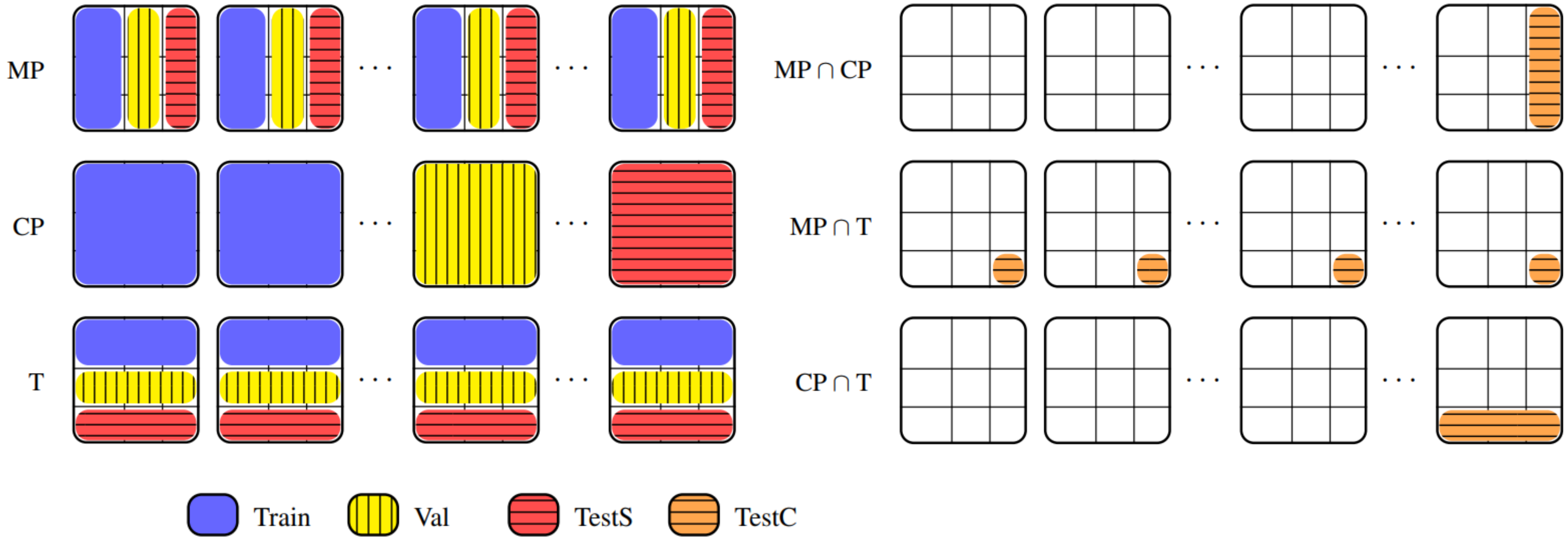


MP = mixed-project CP = cross-project T = time-segmented

Application of Methodologies: Step 6/6

- 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
6. perform post-processing

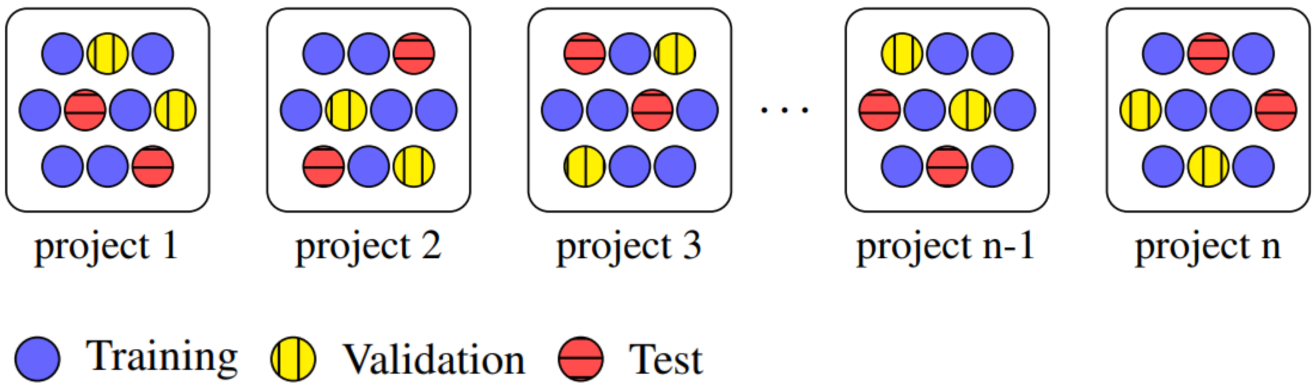


MP = mixed-project CP = cross-project T = time-segmented

- 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

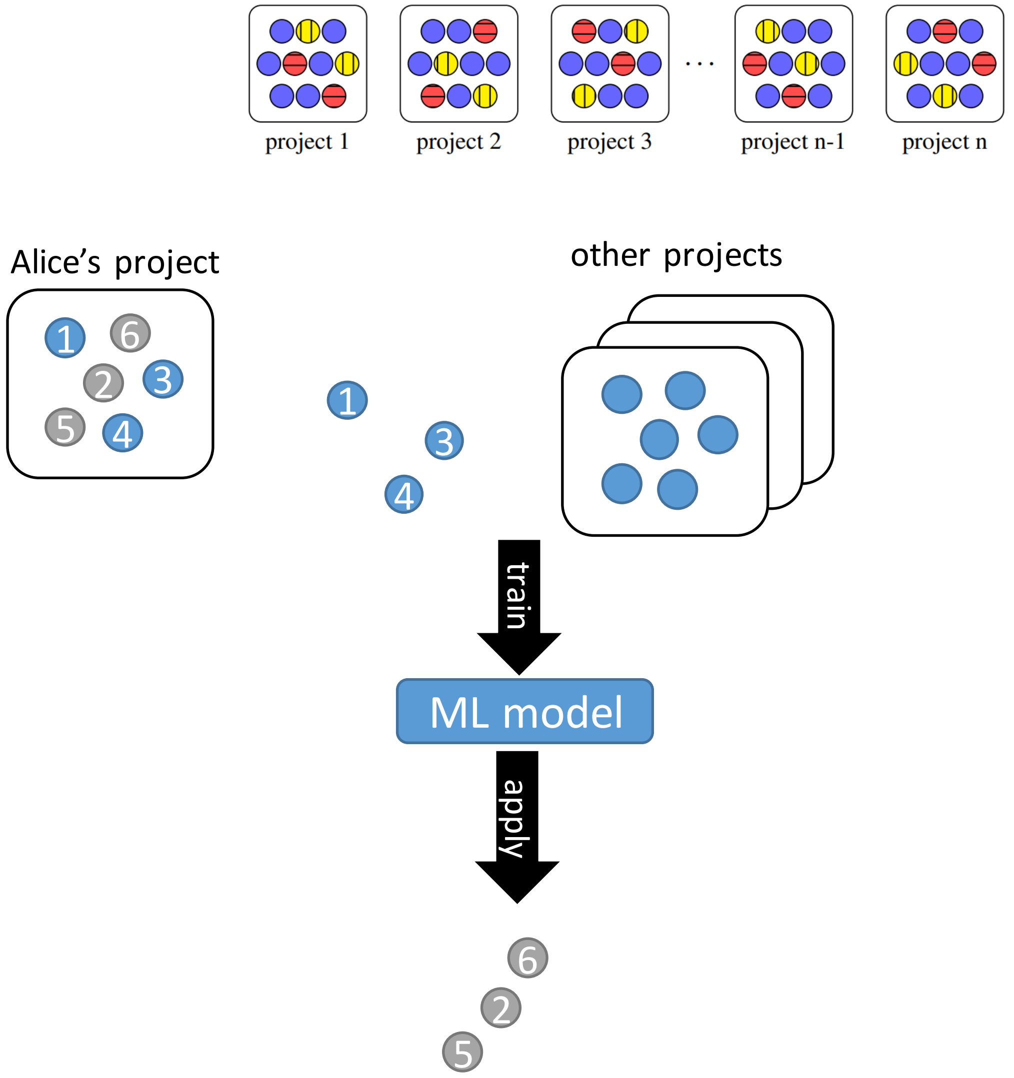
Alice

write comments for only a part of methods

τ

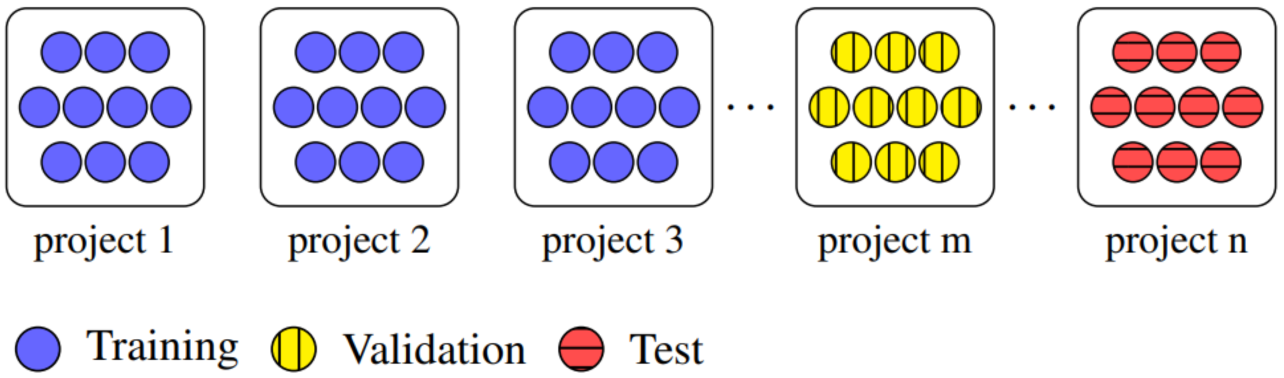
time to add documentations, with ML model

time

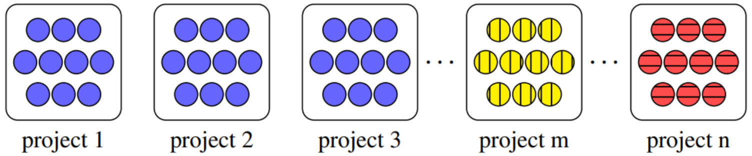


Cross-Project Evaluation Methodology

Randomly shuffle the projects and split them into training, validation, and test sets



Cross-Project Batch-Mode Use Case

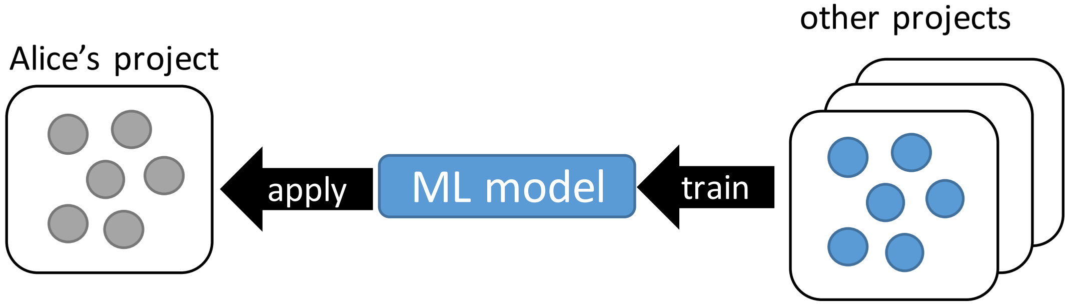


Alice

do not write comments for any method

τ

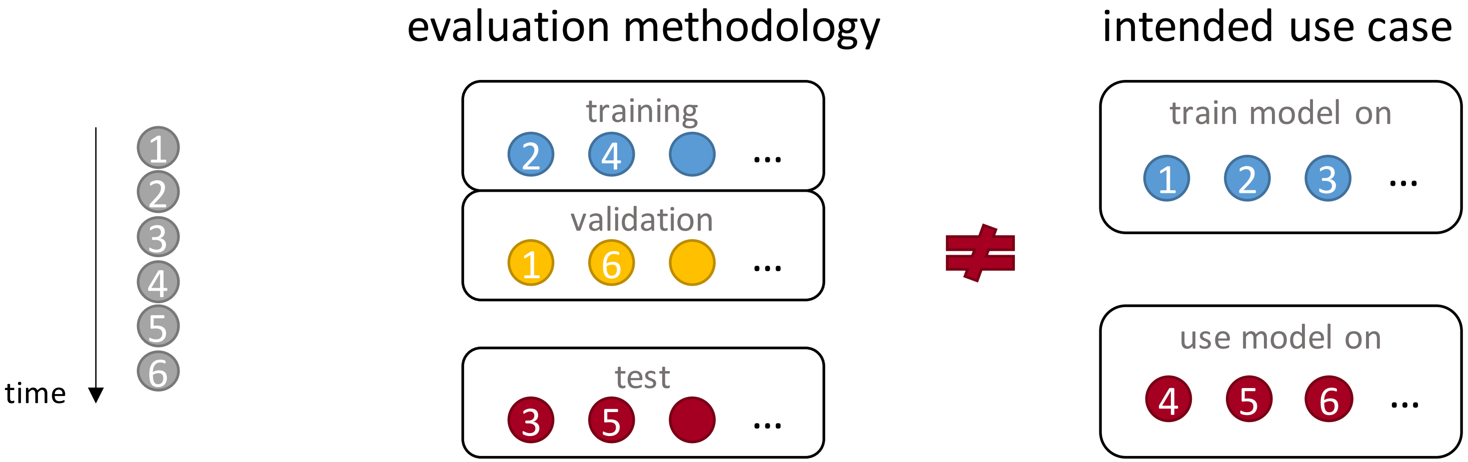
time to add documentations, with ML model



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



Experiments Setup

Task	comment generation	method naming