Facilitating Software Evolution through Natural Language Comments and Dialogue

Sheena Panthaplackel
Dissertation Defense

Committee: Ray Mooney, Jessy Li, Milos Gligoric, Greg Durrett, Charles Sutton
Software Evolution

Software is constantly evolving as developers...
- Incorporate new functionality
- Refactor the code base
- Fix bugs

Problem #1: Developers may unintentionally introduce vulnerabilities when making code changes

Goal #1: Supporting software evolution by upholding software quality amidst constant changes

Problem #2: Sheer volume of needed changes and tight project schedules can delay code changes

Goal #2: Driving software evolution by expediting critical code changes

Identifying the Characteristics of Vulnerable Code Changes: An Empirical Study [Bosu et al., 2014]
Bug Introducing Changes: A Study with Android [Asaduzzaman et al., 2019]
Will this be Quick? A Case Study of Bug Resolution Times across Industrial Projects [Datta et al., 2015]
Developers use natural language in various ways...

- **Queries** for search
- Source code **comments**
- **Commit messages**
- **Bug report discussions**
Developers use natural language in various ways...

- **Queries** for search
- Source code **comments**
- Commit **messages**
- Bug report **discussions**

Supporting Software Evolution
Natural Language & Software

Developers use natural language in various ways...

- **Queries** for search
- **Source code comments**
- **Commit messages**
- **Bug report discussions**

Supporting Software Evolution

Driving Software Evolution
Overview

**Supporting Software Evolution Using Comments**
- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving Software Evolution Using Dialogue**
- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Overview

**Supporting Software Evolution Using Comments**

- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving Software Evolution Using Dialogue**

- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Overview

**Supporting** Software Evolution Using Comments

- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving** Software Evolution Using Dialogue

- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Inconsistency Detection

Document functionality, usage, implementation, error cases, ...

/** Computes the highest value from the list of scores */

public int getBestScore() {
    return Collections.max(scores);
}
Inconsistency Detection

When developers make code changes, they often fail to update comments accordingly.

Leads to time-wasting confusion and vulnerability to bugs.

Old Comment
/** Computes the highest value from the list of scores */

Old Method
public int getBestScore() {
    return Collections.max(scores);
}

New Method
public int getBestScore() {
    return Collections.min(scores);
}

Post Hoc: (Old Comment, New Method)

Is this comment inconsistent?
Inconsistency Detection

When developers make code changes, they often fail to update comments accordingly.

Leads to time-wasting confusion and vulnerability to bugs

Old Comment

/** Computes the **highest** value from the list of scores */

Old Method

```java
public int getBestScore() {
    return Collections.max(scores);
}
```

New Method

```java
public int getBestScore() {
    return Collections.min(scores);
}
```

Post Hoc: (Old Comment, New Method)

Just-In-Time: (Old Comment, New Method, Old Method)

Is this comment inconsistent?

[Tan et al., 2007; Malik et al., 2008; Tan, Zhou, and Padioleau 2011; Tan et al., 2012; Ratol and Robillard 2017; Zhou et al., 2017; Corazza, Maggio, and Scanniello 2018; Liu et al., 2018; Cimasa et al., 2019; Sadu 2019]
Architecture

/** Computes the highest value from the list of scores */

Comment encoder (biGRU) → Self-attention → Contextualized comment encoder (biGRU)

Code edits encoder → Multi-head attention → Fully connected layer

SEQ (Encoded with biGRU)
Code edits as a sequence of tokens
<Keep> public int getBestScore(){ return Collections. <KeepEnd>
<ReplaceOld> max <ReplaceNew> min <ReplaceEnd>
<Keep> (scores);} <KeepEnd>

GRAPH (Encoded with GGNN)
AST node edits as a graph structure

HYBRID
SEQ + GRAPH

GRU [Cho et al., 2014]; Multi-head attention [Vaswani et al., 2017]; GGNN [Li et al., 2016]
Data Collection

<table>
<thead>
<tr>
<th>Commit History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit 1</td>
</tr>
</tbody>
</table>

Old Comment ≠ New Comment

Balanced dataset with ~41K examples from ~1.5K projects

Old Comment = New Comment

Our Dataset

s.t. Old Method ≠ New Method
Results

- Our Just-In-Time approaches can outperform baseline and post hoc models.
Results

- **Just-In-Time RF baseline** [Liu et al., 2018]
- **Post Hoc SEQ**
- **Just-In-Time SEQ**
- **Just-In-Time GRAPH**
- **Just-In-Time HYBRID**

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Just-In-Time RF baseline</td>
<td>70.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Post Hoc SEQ</td>
<td>66.3</td>
<td>66.3</td>
</tr>
<tr>
<td>Just-In-Time SEQ</td>
<td>77.1</td>
<td>77.1</td>
</tr>
<tr>
<td>Just-In-Time GRAPH</td>
<td>76.9</td>
<td>76.9</td>
</tr>
<tr>
<td>Just-In-Time HYBRID</td>
<td>77.7</td>
<td>77.7</td>
</tr>
<tr>
<td>Just-In-Time HYBRID + features</td>
<td>79.6</td>
<td>79.6</td>
</tr>
<tr>
<td>Just-In-Time HYBRID</td>
<td>81.5</td>
<td>81.5</td>
</tr>
</tbody>
</table>

- Just-In-Time approaches can outperform baseline and post hoc models.

E.g., lexical overlap, is Java keyword
Associating Natural Language Comment and Source Code Entities [Chapter 3]
Results

- Our Just-In-Time approaches can outperform baseline and post hoc models.
- Incorporating auxiliary features can further boost performance.

E.g., lexical overlap, is Java keyword.

Associating Natural Language Comment and Source Code Entities [Chapter 3]
Overview

**Supporting Software Evolution Using Comments**
- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving Software Evolution Using Dialogue**
- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Updating Comments Based on Code Changes

Old Comment
/** Computes the highest value from the list of scores */

Old Method
public int getBestScore() {
    return Collections.max(scores);
}

New Method
public int getBestScore() {
    return Collections.min(scores);
}

Is this comment inconsistent?
Generate an updated comment (New Comment) that is consistent with the new version of the code (New Method).

**Old Method**

```java
public int getBestScore() {
    return Collections.max(scores);
}
```

**Old Comment**

```
/** Computes the highest value from the list of scores */
```

**New Method**

```java
public int getBestScore() {
    return Collections.min(scores);
}
```

**New Comment**

```
/** Computes the lowest value from the list of scores */
```

Is this comment inconsistent?
Code Summarization/Comment Generation

Given a body of code (New Method), generate a NL summary/comment (New Comment).

- Ignores rich context from Old Comment and code changes between Old Method and New Method
- Deviates from how developers update comments

We studied...
 Learning to edit Old Comment → New Comment rather than generate New Comment from scratch.

[Iyer et al., 2016; Yao et al., 2018; Yin et al., 2018; Allamanis et al., 2016, Xu et al., 2019, Alon et al., 2019; Fernandes et al., 2019; Sridhara et al., 2011; Movshovitz-Attias and Cohen 2013; Hu et al., 2018; Liang and Zhu 2018; LeClair et al., 2019; Fernandes et al., 2019; Ahmad et al., 2020; Yu et al., 2020]
public int getBestScore () {
    return Collections.max(scores);}

/** Computes the highest value from the list of scores */

Post Processing

Step #2:
- Beam score
- Similarity to Old Comment with METEOR
- Likelihood from comment generation model

Comment encoder (biGRU)

+ features

Comment edit decoder (GRU)

+ features

NL Edits

- highest
- lowest

Code edits encoder (biGRU)

Code Edits

<Keep> public int getBestScore () {
    return Collections.max(scores); <KeepEnd>

<ReplaceOld> max <ReplaceNew> min
<ReplaceEnd>
Edit Model

Old Comment

```java
/** Computes the highest value from the list of scores */
+ features
Comment encoder (biGRU)

Code edits encoder (biGRU)
+ features
Comment edit decoder (GRU)
```

NL Edits

```java
+ features
```

Code Edits

```java
<Keep> public int getBestScore () {
    return Collections . <KeepEnd>
+ReplaceOld> max
+ReplaceNew> min
+ReplaceEnd>
<Keep> ( scores ) ; } <KeepEnd>
```

Post Processing

**Step #1:** Align predicted NL edits with Old Comment

```java
/** Computes the lowest value from the list of scores */
```
Edit Model

Old Comment

```java
/** Computes the highest value from the list of scores */
public int getBestScore () {
    return Collections.max(scores);
}
```

Post Processing

**Step #1:** Align predicted NL edits with Old Comment

```java
/** Computes the lowest value from the list of scores */
```

**Step #2:** Rerank

- Beam score
- Similarity to Old Comment with METEOR
- Likelihood from comment generation model

Code Edits

```java
<Keep> public int getBestScore ( ) { return Collections . max(scores) ; } <KeepEnd> 
```

Comment encoder (biGRU)

Comment edit decoder (GRU)

Code edits encoder (biGRU)

NL Edits

```java
<ReplaceOld> <ReplaceNew> <ReplaceEnd> 
```
Data Collection

### Commit History

<table>
<thead>
<tr>
<th>Commit</th>
<th></th>
<th>Commit</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit 1</td>
<td></td>
<td>Commit N-1</td>
<td></td>
</tr>
<tr>
<td>Commit 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Old Comment
- Old Method
- New Comment
- New Method

- s.t. Old Method ≠ New Method

Inconsistent
Old Comment ≠ New Comment

Consistent
Old Comment = New Comment

Our Dataset

(Old Comment, Old Method, New Comment, New Method)

Balanced dataset with ~41K examples from ~1.5K projects

Comment types:
@return, @param, summary comments
**Results: Human Evaluation**

**Task:** Given Old Comment and code diff:
- Select the most suitable comment
- Select **None** if all options are bad or if Old Comment does not need to be updated

- Our edit model outperforms pure generation and rule-based baselines
- Annotators selected **None** 55% of the time

*Not all code changes warrant a comment update*

Combined Detection and Update of Inconsistent Comments [*Chapter 6*]
Overview

**Supporting Software Evolution Using Comments**

- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving Software Evolution Using Dialogue**

- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Bug Report Discussions

Title: Incorrect distance

devA (Utterance #1)
Seeing negative distance when using 1D grid.

devB (Utterance #2)
Probably a bug in getL1Distance(int x1, int x2)

devC (Utterance #3)
We do $x1 - x2$, which will be negative if $x1 < x2$.

devB (Utterance #4)
We should compute its absolute value.

When a bug is reported, developers engage in a dialogue to collaboratively understand it and ultimately resolve it.

1) User reports bug
2) Developers engage in the discussion
   (understand problem, diagnose cause, propose solution)
3) Bug is resolved with code changes

devC added a commit that referenced this issue
Bug Report Discussions

Title: Incorrect distance

**devA (Utterance #1)**
Seeing negative distance when using 1D grid.

**devB (Utterance #2)**
Probably a bug in `getL1Distance(int x1, int x2)`

**devC (Utterance #3)**
We do `x1 - x2`, which will be negative if `x1 < x2`.

**devB (Utterance #4)**
We should compute its absolute value.

Solution is often formulated in discussion but buried under large amount of text.

*devC added a commit that referenced this issue*
Bug Report Discussions

Title: Incorrect distance

devA (Utterance #1)  
Seeing negative distance when using 1D grid.

devB (Utterance #2)  
Probably a bug in `getL1Distance(int x1, int x2)`.

devC (Utterance #3)  
We do `x1 - x2`, which will be negative if `x1 < x2`.

devB (Utterance #4)  
We should compute its absolute value.

Solution is often formulated in discussion but buried under large amount of text.

**Task:** Generate concise natural language description of the solution by synthesizing relevant content in the discussion when it emerges in real-time.

`devC` added a commit that referenced this issue
Bug Report Discussions

**Title:** Incorrect distance

**devA (Utterance #1)**
Seeing negative distance when using 1D grid.

**devB (Utterance #2)**
Probably a bug in `getL1Distance(int x1, int x2)`

**devC (Utterance #3)**
We do $x_1 - x_2$, which will be negative if $x_1 < x_2$.

**devB (Utterance #4)**
We should compute its absolute value.

**NL Solution Description**
Compute absolute value of $x_1 - x_2$ in `getL1Distance`

Solution is often formulated in discussion but buried under large amount of text.

**Task:** Generate concise natural language description of the solution by synthesizing relevant content in the discussion when it emerges in real-time

*devC added a commit that referenced this issue*
Title: Incorrect distance

**devA (Utterance #1)**
Seeing negative distance when using 1D grid.

**devB (Utterance #2)**
Probably a bug in `getL1Distance(int x1, int x2)`

**devC (Utterance #3)**
We do `x1 - x2`, which will be negative if `x1 < x2`.

**devB (Utterance #4)**
We should compute its absolute value.

**NL Solution Description**
Compute absolute value of `x1 - x2` in `getL1Distance`

---

Solution is often formulated in discussion but buried under large amount of text.

**Task:** Generate concise natural language description of the solution by synthesizing relevant content in the discussion when it emerges in real-time

**Data:** 12K bug reports reports for open-source Java projects from GitHub Issues which are linked to a commit/PR
**Benchmarking Models:** Generating Solution Descriptions

**Copy Title:** Brief description of problem (e.g., Incorrect Distance)

**Transformer**
(randomly initialized)

- **Output Probabilities**
  - Softmax
  - Linear

- **Add & Norm**
  - Feed Forward

- **Add & Norm**
  - Multi-Head Attention

- **Add & Norm**
  - Masked Multi-Head Attention

**PLBART** [Ahmad et al., 2021]
Pretrained as a denoising autoencoder on technical text and source code

**Bidirectional Encoder**

**Autoregressive Decoder**

[Figure from Vaswani et al., 2017]

[Figure from Lewis et al., 2020]
Results: Generating Solution Descriptions

Based on automated metrics and human evaluation, PLBART outperforms baselines.

**Automated metrics**

- **Copy Title**
  - BLEU: 10.0
  - METEOR: 8.3
  - ROUGE: 16.6

- **Transformer**
  - BLEU: 10.2
  - METEOR: 7.5
  - ROUGE: 20.1

- **PLBART**
  - BLEU: 12.3
  - METEOR: 10.2
  - ROUGE: 21.9

**Human evaluation**

(8 annotators, 160 examples)

- “Select the most informative generated description(s)”

- **Copy Title**: 6.0
- **Transformer**: 1.2
- **PLBART**: 39.5

Based on automated metrics and human evaluation, PLBART outperforms baselines.
Bug Report Discussions

Title: Incorrect distance

devA (Utterance #1)
Seeing negative distance when using 1D grid.

devB (Utterance #2)
Probably a bug in `getL1Distance(int x1, int x2)`

devC (Utterance #3)
We do `x1 - x2`, which will be negative if `x1 < x2`.

devB (Utterance #4)
We should compute its absolute value.

NL Solution Description
Compute absolute value of `x1 - x2` in `getL1Distance`

Solution is often formulated in discussion but buried under large amount of text.

Task: Generate concise natural language description of the solution by synthesizing relevant content in the discussion when it emerges in real-time

Data: 12K bug reports reports for open-source Java projects from GitHub Issues which are linked to a commit/PR
Generating Solution Descriptions in Real-Time

Pipelined System

Title and Utterances #1...k → PLBART Encoder → Classification Head → Enough context to generate at time step → Generated solution description

Trained Model for Generating Solution Descriptions
Generating Solution Descriptions in Real-Time

Pipelined System

Jointly Trained System

Title and Utterances #1...k → PLBART Encoder → Classification Head → Enough context to generate at time step

Final decoder state @ \( t_k \) → Final decoder state @ \( t_{k-1} \) → Capture informativeness of generated description

Has the level of informativeness improved?

NOTE: For a given discussion, generation is performed during at most 1 time step \( k \) (i.e., once generation is performed at \( t=k \), classification/generation will not be performed for \( t > k \).
### Results: Generating Solution Descriptions in Real-Time

<table>
<thead>
<tr>
<th>Human Evaluation (60 annotators, 120 examples)</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
</table>

**NOTE:** For a given discussion, generation is performed during at most 1 time step $k$ (i.e., once generation is performed at $t=k$, classification/generation will not be performed for $t > k$.)
### Results: Generating Solution Descriptions in Real-Time

<table>
<thead>
<tr>
<th>Human Evaluation (60 annotators, 120 examples)</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario #1: System generates at time step $k$</td>
<td>64.6%</td>
<td>63.6%</td>
</tr>
</tbody>
</table>

**NOTE:** For a given discussion, generation is performed during at most 1 time step $k$ (i.e., once generation is performed at $t=k$, classification/generation will not be performed for $t > k$.)
### Results: Generating Solution Descriptions in Real-Time

**Human Evaluation** (60 annotators, 120 examples)

<table>
<thead>
<tr>
<th>Scenario #1: System generates at time step k</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there sufficient context about the solution at time step k?</td>
<td>64.6%</td>
<td>63.6%</td>
</tr>
</tbody>
</table>

### NOTE:
For a given discussion, generation is performed during at most 1 time step $k$ (i.e., once generation is performed at $t=k$, classification/generation will not be performed for $t > k$.)
## Results: Generating Solution Descriptions in Real-Time

### Human Evaluation (60 annotators, 120 examples)

<table>
<thead>
<tr>
<th>Scenario #1: System generates at time step k</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there sufficient context about the solution at time step k?</td>
<td>64.6%</td>
<td>63.6%</td>
</tr>
</tbody>
</table>

Rate the informativeness of the generated description:

1 - Incomprehensible, completely incorrect, irrelevant
2 - Generic, rephrasing the problem
3 - Includes some useful information but does not capture the solution
4 - Partially captures solution
5 - Completely captures solution

When sufficient context is available, system output is useful.

NOTE: For a given discussion, generation is performed during at most 1 time step k (i.e., once generation is performed at t=k, classification/generation will not be performed for t > k.)
**Results: Generating Solution Descriptions in Real-Time**

<table>
<thead>
<tr>
<th>Human Evaluation (60 annotators, 120 examples)</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario #1: System generates at time step k</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is there sufficient context about the solution at time step k?</td>
<td>39.0%</td>
<td>33.8%</td>
</tr>
<tr>
<td>Rate the informativeness of the generated description:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - Incomprehensible, completely incorrect, irrelevant</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - Generic, rephrasing the problem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - Includes some useful information but does not capture the solution</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>4 - Partially captures solution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 - Completely captures solution</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Scenario #2: System refrains from generating</strong></td>
<td>35.4%</td>
<td>36.4%</td>
</tr>
</tbody>
</table>

NOTE: For a given discussion, generation is performed during at most 1 time step k (i.e., once generation is performed at t=k, classification/generation will not be performed for t > k.)
## Results: Generating Solution Descriptions in Real-Time

**Human Evaluation** (60 annotators, 120 examples)

<table>
<thead>
<tr>
<th>Scenario #1: System generates at time step k</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there sufficient context about the solution at time step k?</td>
<td>64.6%</td>
<td>63.6%</td>
</tr>
</tbody>
</table>

*Rate the informativeness of the generated description:*
1. Incomprehensible, completely incorrect, irrelevant
2. Generic, rephrasing the problem
3. Includes some useful information but does not capture the solution
4. Partially captures solution
5. Completely captures solution

<table>
<thead>
<tr>
<th>Rate</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario #2: System refrains from generating</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there sufficient context about the solution at any point in the discussion?</td>
<td>35.4%</td>
<td>36.4%</td>
</tr>
</tbody>
</table>

When sufficient context is available, system output is useful.

**NOTE:** For a given discussion, generation is performed during at most 1 time step $k$ (i.e., once generation is performed at $t=k$, classification/generation will not be performed for $t > k$.}
**Results: Generating Solution Descriptions in Real-Time**

<table>
<thead>
<tr>
<th>Human Evaluation (60 annotators, 120 examples)</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario #1: System generates at time step k</strong></td>
<td>64.6%</td>
<td>63.6%</td>
</tr>
<tr>
<td>Is there sufficient context about the solution at time step k?</td>
<td>39.0%</td>
<td>33.8%</td>
</tr>
<tr>
<td>Rate the informativeness of the generated description:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - Incomprehensible, completely incorrect, irrelevant</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>2 - Generic, rephrasing the problem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - Includes some useful information but does not capture the solution</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>4 - Partially captures solution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 - Completely captures solution</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Scenario #2: System refrains from generating</strong></td>
<td>35.4%</td>
<td>36.4%</td>
</tr>
<tr>
<td>Is there sufficient context about the solution at any point in the discussion?</td>
<td>34.2%</td>
<td>37.0%</td>
</tr>
</tbody>
</table>

*NOTE: For a given discussion, generation is performed during at most 1 time step k (i.e., once generation is performed at t=k, classification/generation will not be performed for t > k.*
# Results: Generating Solution Descriptions in Real-Time

<table>
<thead>
<tr>
<th>Human Evaluation (60 annotators, 120 examples)</th>
<th>Pipelined</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario #1: System generates at time step $k$</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is there sufficient context about the solution at time step $k$?</td>
<td>64.6%</td>
<td>63.6%</td>
</tr>
<tr>
<td>Is there NOT sufficient context about the solution at time step $k$?</td>
<td>39.0%</td>
<td>33.8%</td>
</tr>
</tbody>
</table>

Rate the informativeness of the generated description:

1 - Incomprehensible, completely incorrect, irrelevant
2 - Generic, rephrasing the problem
3 - Includes some useful information but does not capture the solution
4 - Partially captures solution
5 - Completely captures solution

- **3.3**
- **3.3**

<table>
<thead>
<tr>
<th><strong>Scenario #2: System refrains from generating</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there sufficient context about the solution at any point in the discussion?</td>
<td>35.4%</td>
<td>36.4%</td>
</tr>
</tbody>
</table>

When sufficient context is available, system output is useful.

Balancing the trade-off between generating too early and deferring to later time steps is an open challenge.

**NOTE:** For a given discussion, generation is performed during at most 1 time step $k$ (i.e., once generation is performed at $t=k$, classification/generation will not be performed for $t > k$.}
Overview

**Supporting Software Evolution Using Comments**

- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving Software Evolution Using Dialogue**

- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Implementing Bug-Fixing Code Changes

**Title:** Incorrect distance

- **devA (Utterance #1)**
  Seeing negative distance when using 1D grid.

- **devB (Utterance #2)**
  Probably a bug in `getL1Distance(int x1, int x2)`.

- **devC (Utterance #3)**
  We do `x1 - x2`, which will be negative if `x1 < x2`.

- **devB (Utterance #4)**
  We should compute its absolute value.

**NL Solution Description**
Compute absolute value of `x1 - x2` in `getL1Distance`.

**Automated Bug-Fixing Model**

**Suggested Bug-Fix**

```java
public int getL1Distance (int x1, int x2) {
    - return x1-x2;
    + return Math.abs(x1-x2);
}
```

**Bug-fixing commit**

[Le Goues et al., 2012; Kim et al., 2013; Ke et al., 2015; Le et al., 2017; Tufano et al., 2019; Chen et al., 2019; Lutellier et al., 2020; Mashhadi and Hemmati, 2021; Allamanis et al., 2021; Chakraborty and Ray, 2021]
Automated Bug-Fixing Models

Buggy Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
.append(table)
.append("\n");
```

Fixed Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
.append(table);
```

- Extremely challenging task with such limited context
- MODIT incorporates two additional sources of input

Full Buggy Method

```java
void emptyImplicitTable(String table, int line) {
    sb.append("Invalid table definition due to empty implicit table name: ")
    .append(table)
    .append("\n");
}
```

Natural Language Context

Removed trailing newlines from error messages

MODIT [Chakraborty and Ray, 2021]
Sources of Natural Language Context

Natural Language Context

- Removed trailing newlines from error messages

MODIT:

- Requires prompting developers. Burdensome for developers
- Simulated through oracle commit messages. Inaccurately reflect the available context since they are written after the bug-fixing commits

Is there a source of naturally-occurring natural language context that is available before the task is to be performed?

Bug Report Discussions

Title: Parsing exception messages contain trailing newlines

Utterance #1
Some of the parsing exceptions thrown by toml4j contains trailing newlines. This is somewhat unusual, and causes empty lines in log files when the exception messages are logged...

Utterance #2
The idea was to be able to display multiple error messages at once. However, processing stops as soon as an error is encountered, so that's not even possible. Removing the newlines shouldn't be a problem, then.

NL Solution Description
remove trailing newlines from toml4j log messages
Deriving NL Context from Bug Report Discussions

Title: Parsing exception messages contain trailing newlines

Utterance #1
Some of the parsing exceptions thrown by toml4j contains trailing newlines. This is somewhat unusual, and causes empty lines in log files when the exception messages are logged...

Utterance #2
The idea was to be able to display multiple error messages at once. However, processing stops as soon as an error is encountered, so that's not even possible. Removing the newlines shouldn't be a problem, then.

Deriving Context Heuristically
- Whole discussion
- Title
- Last utterance
Deriving NL Context from Bug Report Discussions

Title: Parsing exception messages contain trailing newlines

Utterance #1
Some of the parsing exceptions thrown by toml4j contains trailing newlines. This is somewhat unusual, and causes empty lines in log files when the exception messages are logged...

Utterance #2
The idea was to be able to display multiple error messages at once. However, processing stops as soon as an error is encountered, so that's not even possible. Removing the newlines shouldn't be a problem, then.

Trained Model for Generating Solution Descriptions

remove trailing newlines from toml4j log messages

Deriving Context Heuristically
- Whole discussion
- Title
- Last utterance

Deriving Context Algorithmically
- Solution description
- Solution description + title
Deriving NL Context from Bug Report Discussions

**Title:** Parsing exception messages contain trailing newlines

**Utterance #1**
Some of the parsing exceptions thrown by toml4j contains trailing newlines. This is somewhat unusual, and causes empty lines in log files when the exception messages are logged...

**Utterance #2**
The idea was to be able to display multiple error messages at once. However, processing stops as soon as an error is encountered, so that's not even possible. Removing the newlines shouldn't be a problem, then.

Deriving Context **Heuristically**
- Whole discussion
- Title
- Last utterance

Deriving Context **Algorithmically**
- Solution description
- Solution description + title

Identify the most highly attended input token during each step of decoding
Deriving NL Context from Bug Report Discussions

**Title:** Parsing exception messages contain trailing newlines

**Utterance #1**
Some of the parsing exceptions thrown by toml4j contains trailing newlines. This is somewhat unusual, and causes empty lines in log files when the exception messages are logged...

**Utterance #2**
The idea was to be able to display multiple error messages at once. However, processing stops as soon as an error is encountered, so that's not even possible. Removing the newlines shouldn't be a problem, then.

Deriving Context **Heuristically**
- Whole discussion
- Title
- Last utterance

Deriving Context **Algorithmically**
- Solution description
- Solution description + title
- Attended segments

Identify the most highly attended input token during each step of decoding and the discussion segment to which it belongs.
Approach

Buggy Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
    .append(table)
    .append("\n");
```

Fixed Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
    .append(table);
```

Full Buggy Method

```java
void emptyImplicitTable(String table, int line) {
    sb.append("Invalid table definition due to empty implicit table name: ")
        .append(table)
        .append("\n");
}
```

Natural Language Context

Removed trailing newlines from error messages

MODIT [Chakraborty and Ray, 2021]
Approach

Buggy Code

```java
sb.append("Invalid table definition due to empty implicit table name: ");
.append(table);
.append("\n");
```

Full Buggy Method

```java
void emptyImplicitTable(String table, int line) {
    sb.append("Invalid table definition due to empty implicit table name: ");
    .append(table);
    .append("\n");
}
```

Natural Language Context

Removed trailing newlines from error messages
Approach

Buggy Code

```java
sb.append("Invalid table definition due to
empty implicit table name: ")
    .append(table)
    .append("\n");
```

Full Buggy Method

```java
void emptyImplicitTable(String table, int line) {
    sb.append("Invalid table definition due to
empty implicit table name: ")
        .append(table)
        .append("\n");
}
```

Fixed Code

```java
sb.append("Invalid table definition due to
empty implicit table name: ")
    .append(table);
```

Natural Language Context

Removed trailing newlines from error messages

**Oracle commit message**
Approach

Buggy Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
.append(table)
.append("\n");
```

Full Buggy Method

```java
void emptyImplicitTable(String table, int line) {
    sb.append("Invalid table definition due to empty implicit table name: ")
        .append(table)
        .append("\n");
}
```

Fixed Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
.append(table);
```

Full Buggy Method

```java
void emptyImplicitTable(String table, int line) {
    sb.append("Invalid table definition due to empty implicit table name: ")
        .append(table)
        .append("\n");
}
```

Fixed Code

```java
sb.append("Invalid table definition due to empty implicit table name: ")
.append(table);
```

NL Context derived from bug report discussion:

- Whole discussion
- Title
- Last utterance
- Solution description
- Solution description + title
- Attended segments

Natural Language Context

- Removed trailing newlines from error messages

Oracle commit message

Data

MODIT is built using the **Bug-Fix Patches (BFP) datasets** [Tufano et al., 2019]

- BFP
  - $\text{BFP}_{\text{small}}$ (<50 method tokens): $\sim58.3K$
  - $\text{BFP}_{\text{medium}}$ (50-100 method tokens): $\sim65.4K$

- ~58.6K projects
- ~365K issue reports linked to commits made in time frame used to collect BFP

Mapping BFP bug-fixing commits to bug report discussions to form **Discussion-Augmented BFP datasets**

- Not all bug-fixing commits are linked to bug reports

- Disc-BFP
  - $\text{Disc-BFP}_{\text{small}}$: $\sim3K$
  - $\text{Disc-BFP}_{\text{medium}}$: $\sim3.3K$
Initializing Model Parameters

PLBART

→ Finetune on 3-3.3K examples in Disc-BFP for bug-fixing task

→ Finetune on 58.3-65.4K examples in BFP for bug-fixing task

MODIT checkpoints
Initializing Model Parameters

**PLBART**

- Finetune on **3-3.3K** examples in Disc-BFP for bug-fixing task

- Finetune on **58.3-65.4K** examples in BFP for bug-fixing task

**MODIT checkpoints**

- **Without NL:** buggy + method
- **With NL:** buggy + method + oracle commit message

- Finetune on **3-3.3K** examples in Disc-BFP for bug-fixing task

**Our Models + Baselines**

MODIT [Chakraborty and Ray, 2021]
Results

Disc-BFP<sub>small</sub>

- Initializing without NL checkpoint:
  - Oracle commit message: 33.8
  - Whole discussion: 33.4
  - Title: 35.5
  - Last utterance: 35.2
  - Solution description: 35.5
  - Solution description + title: 36.2
  - Attended segments: 36.2

- Initializing with NL checkpoint:
  - Oracle commit message: 35.5
  - Whole discussion: 34.1
  - Title: 35.2
  - Last utterance: 36.2
  - Solution description: 33.4
  - Solution description + title: 39.2
  - Attended segments: 36.9

Disc-BFP<sub>medium</sub>

- Initializing without NL checkpoint:
  - Oracle commit message: 27.1
  - Whole discussion: 27.1
  - Title: 25.9
  - Last utterance: 27.4
  - Solution description: 27.4
  - Solution description + title: 28.0
  - Attended segments: 28.9

- Initializing with NL checkpoint:
  - Oracle commit message: 25.3
  - Whole discussion: 25.9
  - Title: 25.6
  - Last utterance: 25.3
  - Solution description: 25.6
  - Solution description + title: 26.5
  - Attended segments: 26.2

Legend:
- Without NL
- Oracle commit message
- Whole discussion
- Title
- Last utterance
- Solution description
- Solution description + title
- Attended segments
Results

Disc-BFP<sub>small</sub>

- Initializing <em>without</em> NL checkpoint:
  - Exact Match (%): 33.8

- Initializing <em>with</em> NL checkpoint:
  - Exact Match (%): 35.5

Disc-BFP<sub>medium</sub>

- Initializing <em>without</em> NL checkpoint:
  - Exact Match (%): 27.1

- Initializing <em>with</em> NL checkpoint:
  - Exact Match (%): 25.3
Results

**Disc-BFP\textsubscript{small}**

<table>
<thead>
<tr>
<th>Exact Match (%)</th>
<th>Initializing without NL checkpoint</th>
<th>Initializing with NL checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without NL</td>
<td>33.8, 33.4, 33.1, 35.5, 35.2, 33.8, 35.5, 36.2</td>
<td>35.5, 34.1, 35.2, 36.2, 33.8, 35.5, 36.2</td>
</tr>
<tr>
<td>Oracle commit message</td>
<td>36.9</td>
<td>39.2</td>
</tr>
<tr>
<td>Whole discussion</td>
<td>34.1</td>
<td>35.2</td>
</tr>
<tr>
<td>Title</td>
<td>27.1, 27.4, 27.1, 28.9, 27.4, 28.0</td>
<td>25.3, 25.9, 25.6, 25.3, 25.6, 26.2</td>
</tr>
<tr>
<td>Last utterance</td>
<td>27.1</td>
<td>25.3</td>
</tr>
<tr>
<td>Solution description</td>
<td>36.9</td>
<td>24.1</td>
</tr>
<tr>
<td>Solution description + title</td>
<td>39.2</td>
<td></td>
</tr>
<tr>
<td>Attended segments</td>
<td>33.8</td>
<td></td>
</tr>
</tbody>
</table>

**Disc-BFP\textsubscript{medium}**

<table>
<thead>
<tr>
<th>Exact Match (%)</th>
<th>Initializing without NL checkpoint</th>
<th>Initializing with NL checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without NL</td>
<td>27.1, 27.4, 27.1, 25.9, 27.4, 25.6, 25.6</td>
<td>25.9, 25.6, 25.3, 25.6, 26.5, 26.2</td>
</tr>
<tr>
<td>Oracle commit message</td>
<td>25.3</td>
<td>24.1</td>
</tr>
<tr>
<td>Whole discussion</td>
<td>27.1</td>
<td>25.9</td>
</tr>
<tr>
<td>Title</td>
<td>27.1</td>
<td>25.6</td>
</tr>
<tr>
<td>Last utterance</td>
<td>27.1</td>
<td>25.3</td>
</tr>
<tr>
<td>Solution description</td>
<td>24.1</td>
<td>25.6</td>
</tr>
<tr>
<td>Solution description + title</td>
<td>25.3</td>
<td></td>
</tr>
<tr>
<td>Attended segments</td>
<td>25.6</td>
<td></td>
</tr>
</tbody>
</table>
Results

Disc-BFP\textsubscript{small}

Initializing \textbf{without} NL checkpoint

Initializing \textbf{with} NL checkpoint

Disc-BFP\textsubscript{medium}

Initializing \textbf{without} NL checkpoint

Initializing \textbf{with} NL checkpoint

- **NL context from bug report discussions yields improvement over baselines without NL**
Results

Disc-BFP\textsubscript{small}

\begin{itemize}
\item Initializing without NL checkpoint
\item Initializing with NL checkpoint
\end{itemize}

Disc-BFP\textsubscript{medium}

\begin{itemize}
\item Initializing without NL checkpoint
\item Initializing with NL checkpoint
\end{itemize}

- NL context from bug report discussions yields improvement over baselines without NL

Oracle commit message

<table>
<thead>
<tr>
<th>Without NL</th>
<th>Whole discussion</th>
<th>Title</th>
<th>Last utterance</th>
<th>Solution description</th>
<th>Solution description + title</th>
<th>Attended segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.8</td>
<td>35.5</td>
<td>35.2</td>
<td>35.5</td>
<td>36.2</td>
<td>39.2</td>
<td>36.9</td>
</tr>
<tr>
<td>33.4</td>
<td>35.5</td>
<td>35.2</td>
<td>35.5</td>
<td>36.2</td>
<td>39.2</td>
<td>36.9</td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>24</td>
<td>26</td>
<td>28</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td>27.1</td>
<td>27.1</td>
<td>27.4</td>
<td>27.4</td>
<td>28.0</td>
<td>25.6</td>
<td>24.1</td>
</tr>
<tr>
<td>25.9</td>
<td>25.6</td>
<td>25.3</td>
<td>25.6</td>
<td>26.5</td>
<td>26.2</td>
<td>25.9</td>
</tr>
</tbody>
</table>

The University of Texas at Austin
Results

Disc-BFP\textsubscript{small}

- Initializing without NL checkpoint
  - Without NL: 33.4
  - Oracle commit message: 36.2
  - Whole discussion: 35.5
  - Title: 35.5
  - Last utterance: 36.2
  - Solution description: 33.1
  - Attended segments: 33.8

- Initializing with NL checkpoint
  - Without NL: 36.9
  - Oracle commit message: 39.2
  - Whole discussion: 36.2
  - Title: 36.2
  - Last utterance: 35.5
  - Solution description: 35.2
  - Attended segments: 36.2

Disc-BFP\textsubscript{medium}

- Initializing without NL checkpoint
  - Without NL: 27.1
  - Oracle commit message: 25.3
  - Whole discussion: 27.4
  - Title: 25.6
  - Last utterance: 26.5
  - Solution description: 28.9
  - Attended segments: 26.2

- Initializing with NL checkpoint
  - Without NL: 24.1
  - Oracle commit message: 26.2
  - Whole discussion: 27.1
  - Title: 27.1
  - Last utterance: 26.9
  - Solution description: 25.9
  - Attended segments: 25.6

- NL context from bug report discussions yields improvement over baselines without NL
Results

Disc-BFP<sub>small</sub>

- Initializing <u>without</u> NL checkpoint:
  - Without NL: 33.8%
  - Whole discussion: 33.4%
  - Title: 33.1%
  - Last utterance: 33.8%
  - Solution description: 35.5%
  - Solution description + title: 36.2%
  - Attended segments: 36.9%

- Initializing <u>with</u> NL checkpoint:
  - Without NL: 35.5%
  - Whole discussion: 36.2%
  - Title: 34.1%
  - Last utterance: 35.2%
  - Solution description: 36.2%
  - Solution description + title: 33.4%
  - Attended segments: 36.2%

Disc-BFP<sub>medium</sub>

- Initializing <u>without</u> NL checkpoint:
  - Without NL: 27.4%
  - Whole discussion: 28.9%
  - Title: 27.4%
  - Last utterance: 28.0%
  - Solution description: 25.9%
  - Solution description + title: 26.0%
  - Attended segments: 25.9%

- Initializing <u>with</u> NL checkpoint:
  - Without NL: 25.4%
  - Whole discussion: 25.6%
  - Title: 25.3%
  - Last utterance: 25.6%
  - Solution description: 26.5%
  - Solution description + title: 26.2%
  - Attended segments: 26.2%

- **NL context from bug report discussions yields improvement over baselines without NL**
- **Context from bug report discussions yields improvement over using the oracle commit message**
Overview

**Supporting Software Evolution Using Comments**
- Associating Natural Language Comment and Source Code Entities
- Just-In-Time Inconsistency Detection Between Comments and Source Code
- Updating Natural Language Comments Based on Code Changes
- Combined Detection and Update of Inconsistent Comments

**Driving Software Evolution Using Dialogue**
- Describing Solutions for Bug Reports Based on Developer Discussions
- Using Bug Report Discussions to Guide Automated Bug Fixing
Future Work: Unifying Related Tasks for Supporting Software Evolution

Supporting software evolution by upholding software quality amidst constant changes

```java
public int getBestScore() {
    return Collections.max(scores);
    return Collections.min(scores);
}
```

Comment Inconsistency Detection/Update

```java
/** Computes the highest value from the list of scores */
```  

Commit Message Generation

```java
fixed bug in getBestScore() to return min score as best
```  

Develop a unified approach for addressing multiple tasks occurring upon code changes.

- General framework for multiple tasks
- Joint/multi-task learning
- Few shot learning and prompt engineering with large pretrained autoregressive models

PLUR: A Unifying, Graph-Based View of Program Learning, Understanding, and Repair [Chen et al. 2021]

Jointly Learning to Repair Code and Generate Commit Message [Bai et al. 2021]

PaLM: Scaling Language Modeling with Pathways [Chowdhery et al. 2022]
Future Work: Driving Software Evolution

Driving software evolution by expediting critical code changes

```java
/** Computes distance as difference between x1 and x2 */
/** Computes distance as magnitude of difference between x1 and x2 */
public int getL1Distance (int x1, int x2) {
    return x1-x2;
    int distance = Math.abs(x1-x2);
    log.debug(String.format("(%d)", distance));
    return distance;
}
```

Reviewer
Please make the log message more descriptive.

Author
Will add in something about it being L1 distance. Anything else that should be included?

Reviewer
Maybe that it’s for the 1D grid?

Author
log.debug(String.format("L1 Distance in 1D (%d)", distance));

Can an intelligent agent collaborate with human developers for more efficient/effective code review?
**Future Work:** Driving Software Evolution

Driving software evolution by expediting critical code changes

```java
/** Computes distance as difference between x1 and x2 */
/** Computes distance as magnitude of difference between x1 and x2 */
public int getL1Distance (int x1, int x2) {
    return x1-x2;
    + int distance = Math.abs(x1-x2);
    + log.debug(String.format("(%d)", distance));
    + return distance;
}
```

**Reviewer**
Please make the log message more descriptive.

**Author**
Will add in something about it being L1 distance. Anything else that should be included?

**Reviewer**
Maybe that it’s for the 1D grid?

**Author**
+ log.debug(String.format("L1 Distance in 1D (%d)", distance));
```
**Future Work:** Driving Software Evolution

**Driving software evolution by expediting critical code changes**

```java
/** Computes distance as difference between x1 and x2 */
/** Computes distance as magnitude of difference between x1 and x2 */
public int getL1Distance (int x1, int x2) {
    return x1-x2;
    +  int distance = Math.abs(x1-x2);
    +  log.debug(String.format("(%d)", distance));
    +  return distance;
}
```

Reviewer
Please make the log message more descriptive.

Author
Will add in something about it being L1 distance. Anything else that should be included?

Reviewer
Maybe that it’s for the 1D grid?

Author
```java
+  log.debug(String.format("(%d)", distance));
+  log.debug(String.format("L1 Distance in 1D (%d)", distance));
```
Future Work: Enhancing Code Representations with Natural Language

```python
def compute_distance(p1, p2):
    return abs(p1.x - p2.x)
```

L1 Distance

vertically adjacent points

Code Representation

"""Computes the distance between two vertically adjacent points as the L1 distance between their X coordinates."""

X coordinates
Acknowledgements