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

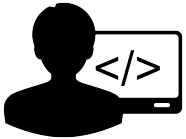
Problem #2: Sheer volume of needed changes and tight project schedules can delay code changes

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

Supporting and Driving Software Evolution through **Natural Language**

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]

Natural Language & Software



Developers use natural language in various ways...

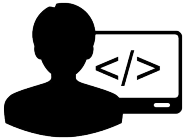
Queries for search

Source code ***comments***

Commit ***messages***

Bug report ***discussions***

Natural Language & Software



Developers use natural language in various ways...

Queries for search

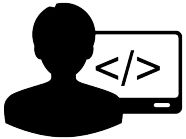
Source code ***comments***

Supporting Software Evolution

Commit ***messages***

Bug report ***discussions***

Natural Language & Software



Developers use natural language in various ways...

Queries for search

Source code ***comments***

Supporting Software Evolution

Commit ***messages***

Bug report ***discussions***

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

Post Hoc: (Old Comment, New Method)

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?

[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]

Inconsistency Detection

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

Leads to time-wasting confusion and vulnerability to bugs

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

Old Comment ✘

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

Old Method

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

Is this comment inconsistent?

New Method

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

[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

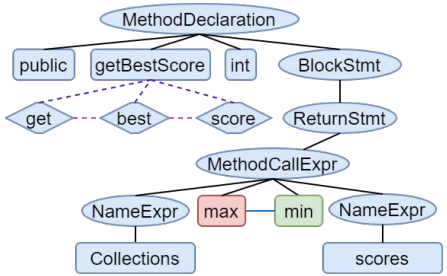
Softmax

Inconsistent?

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>
```

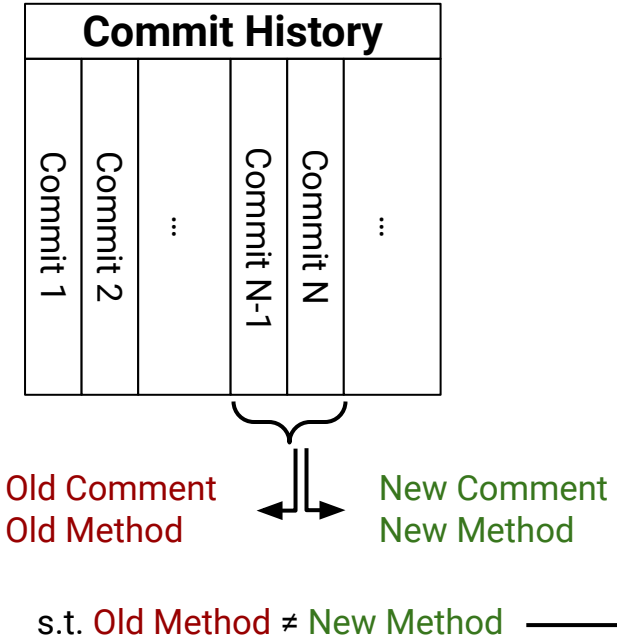
GRAPH *(Encoded with GGNN)*
AST node edits as a graph structure

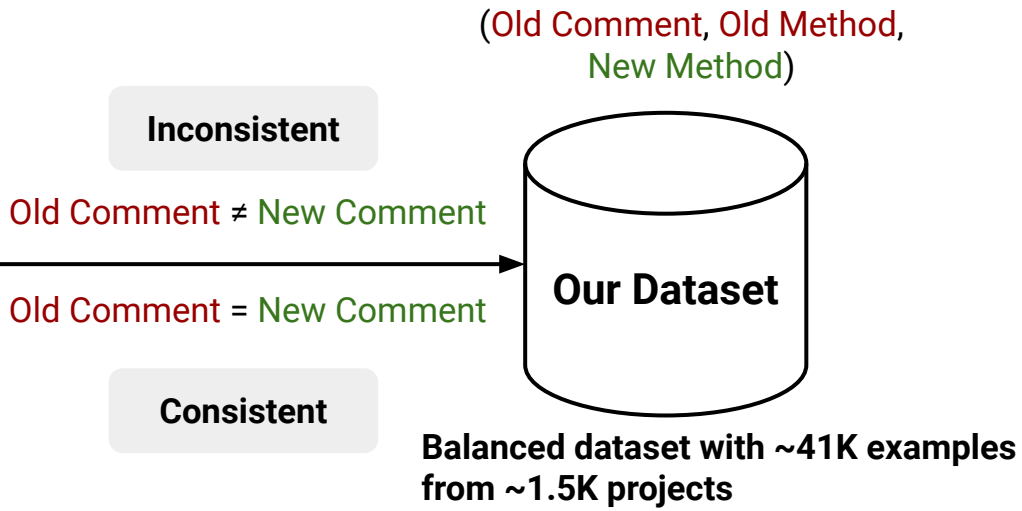


HYBRID
SEQ + GRAPH

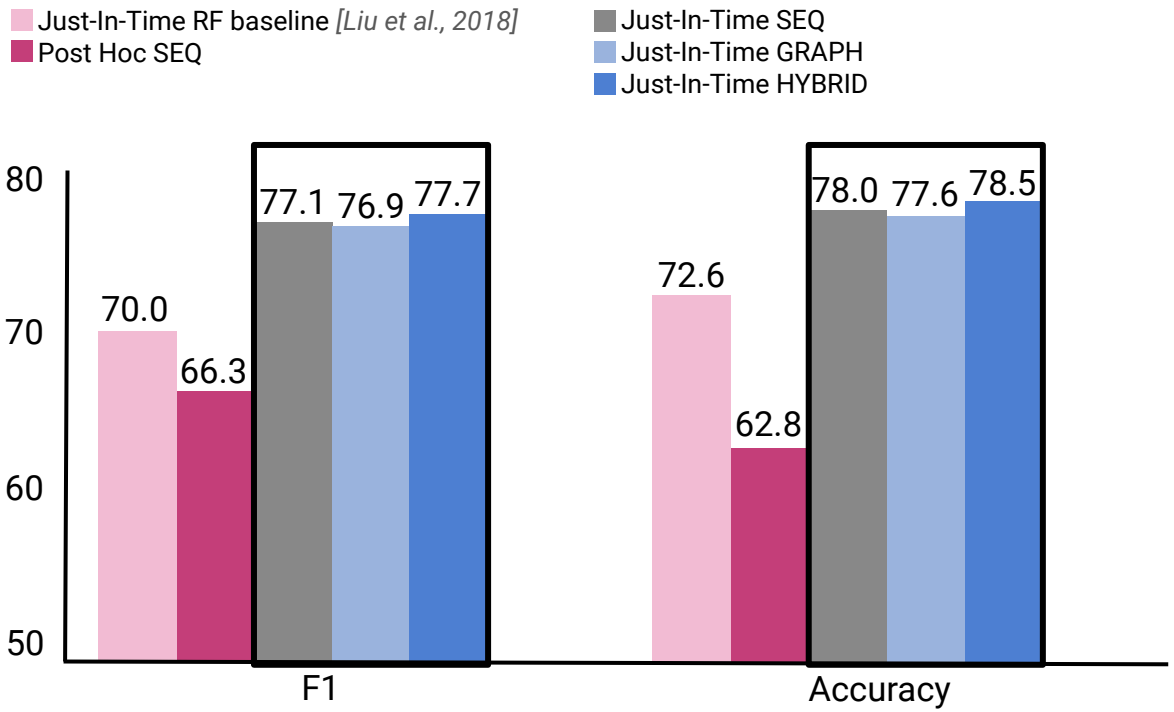
GRU [Cho et al., 2014]; Multi-head attention [Vaswani et al., 2017]; GGNN [Li et al., 2016]

Data Collection



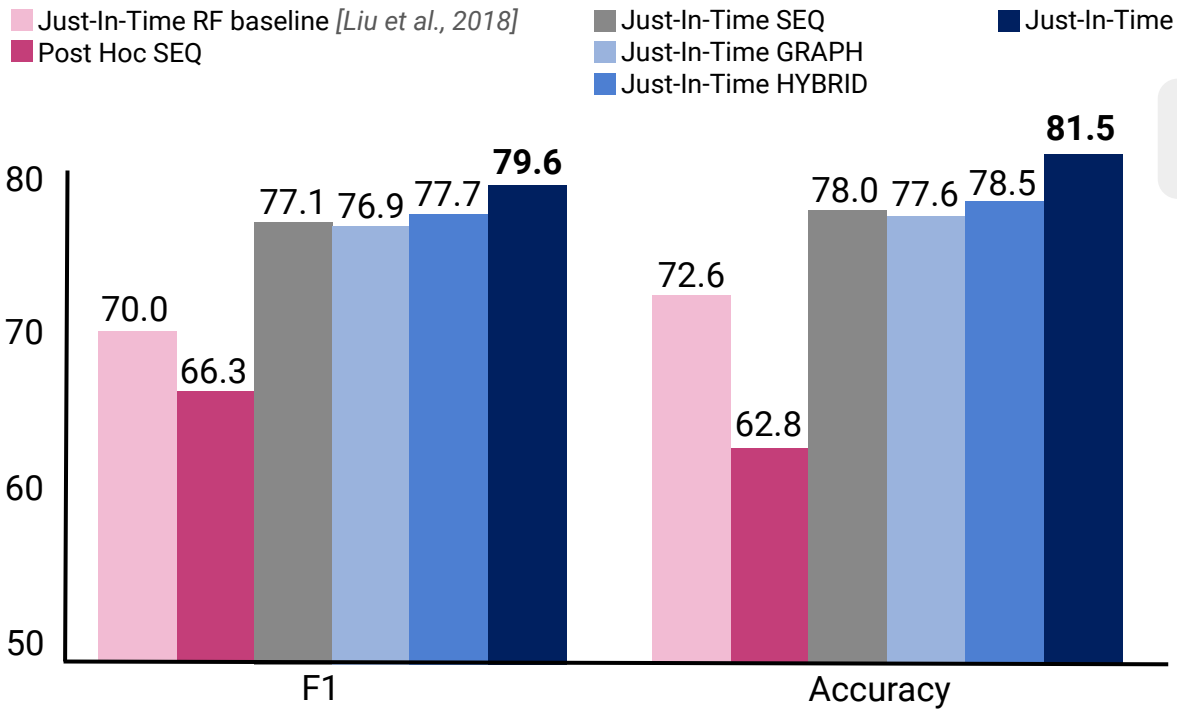


Results



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

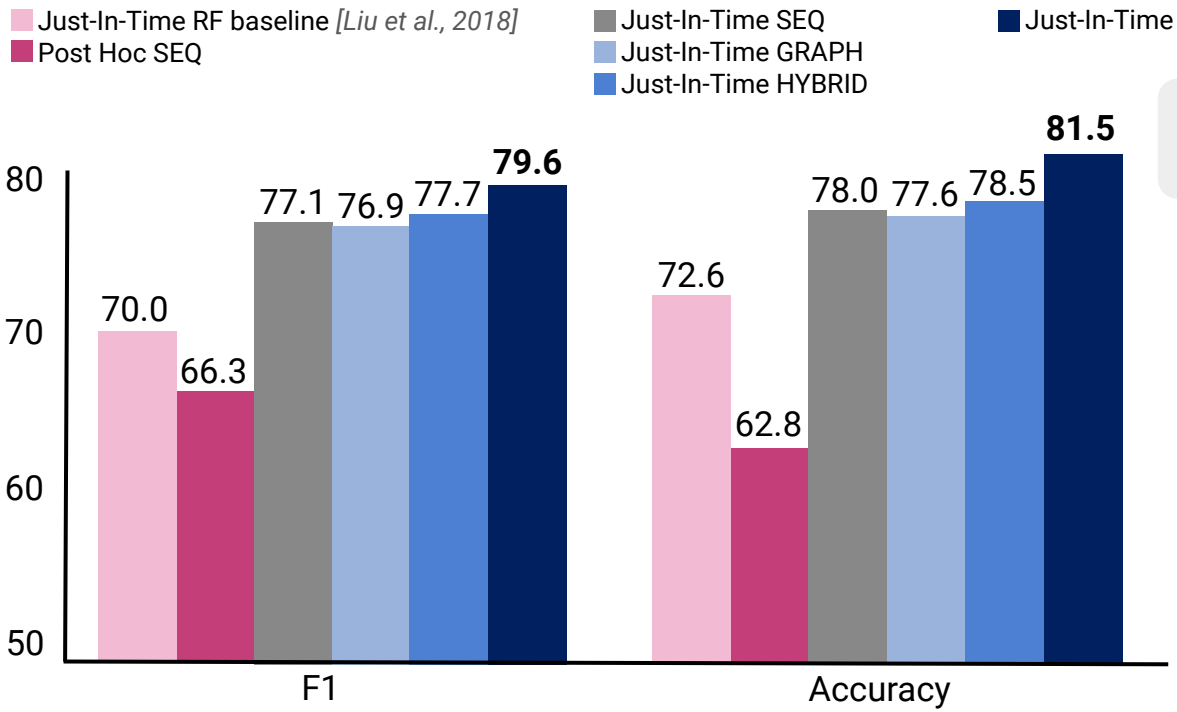
Results



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

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

Results



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

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

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?

Updating Comments Based on Code Changes

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

New Comment

~~Old Comment~~ ✘

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

lowest

Old Method

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

New Method

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

Is this comment inconsistent?

Code Summarization/Comment Generation

Code summarization and Comment generation

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

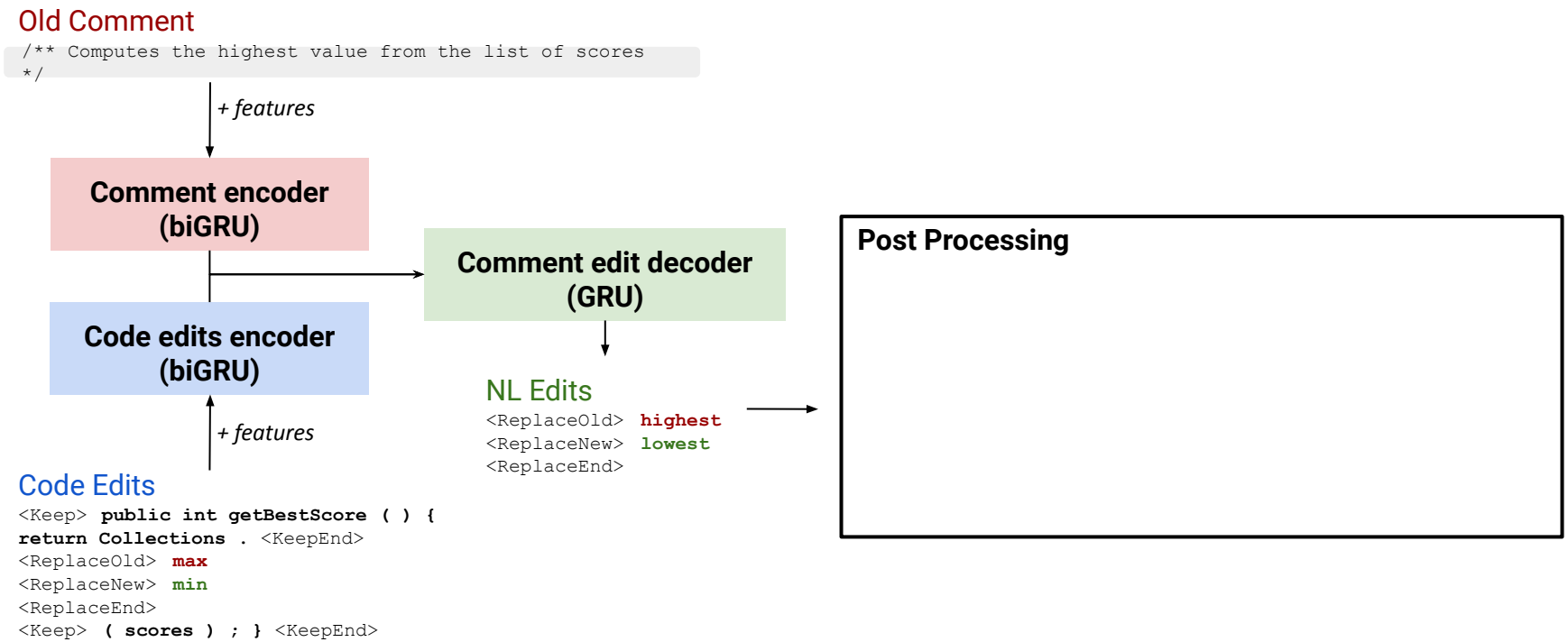
- Ignores rich context from Old Comment and code changes between Old Method and New Method
- Deviates from how developers update comments

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]

Edit Model


Old Comment
/** Computes the highest value from the list of scores
*/
+ features
Comment encoder (biGRU)
Code edits encoder (biGRU)
+ features
Code Edits
<Keep> public int getBestScore ( ) {
return Collections . <KeepEnd>
<ReplaceOld> max
<ReplaceNew> min
<ReplaceEnd>
<Keep> (scores) ; } <KeepEnd>
Comment edit decoder (GRU)
NL Edits
<ReplaceOld> highest
<ReplaceNew> lowest
<ReplaceEnd>
Post Processing

Edit Model

Old Comment

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

+ features

Comment encoder (biGRU)

Code edits encoder (biGRU)

+ features

Code Edits

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

Comment edit decoder (GRU)

NL Edits

```
<ReplaceOld> highest
<ReplaceNew> lowest
<ReplaceEnd>
```

Post Processing

Step #1: Align predicted NL edits with Old Comment

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

Edit Model

Old Comment

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

+ features

Comment encoder (biGRU)

Code edits encoder (biGRU)

+ features

Code Edits

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

Comment edit decoder (GRU)

NL Edits

```
<ReplaceOld> highest
<ReplaceNew> lowest
<ReplaceEnd>
```

Post Processing

Step #1: Align predicted NL edits with Old Comment

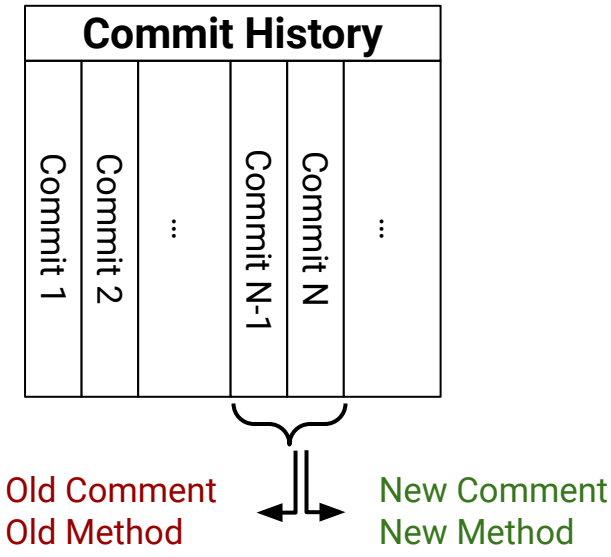
```
/** Computes the lowest value from the list of scores
*/
```

Step #2: Rerank

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

METEOR [Banerjee and Lavie 2005]

Data Collection



s.t. Old Method ≠ New Method

Old Comment ≠ New Comment

Inconsistent

Old Comment = New Comment

Consistent

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

Our Dataset

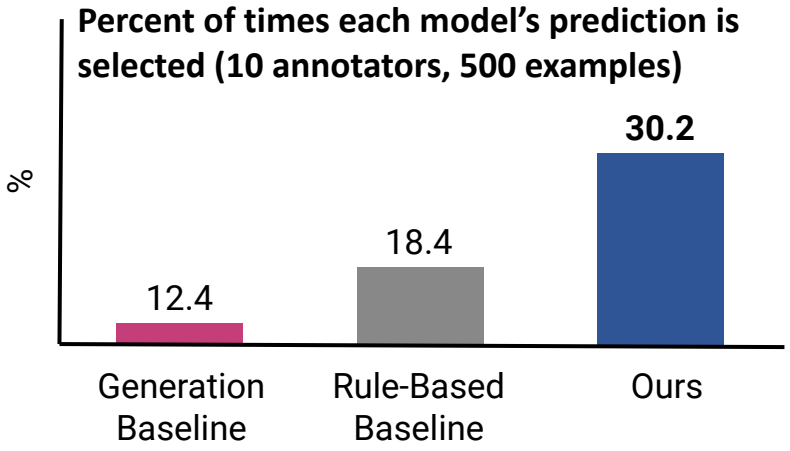
Balanced dataset with ~41K examples from ~1.5K projects

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.

devC added a commit that referenced this issue

1) User reports bug

2) Developers engage in the discussion
(understand problem, diagnose cause, propose solution)

3) Bug is resolved with code changes

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

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.

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

devC added a commit that referenced this issue

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

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`

devC added a commit that referenced this issue

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

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`

Commit message/PR title

Time step of commit/PR

***devC** added a commit that referenced this issue*

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

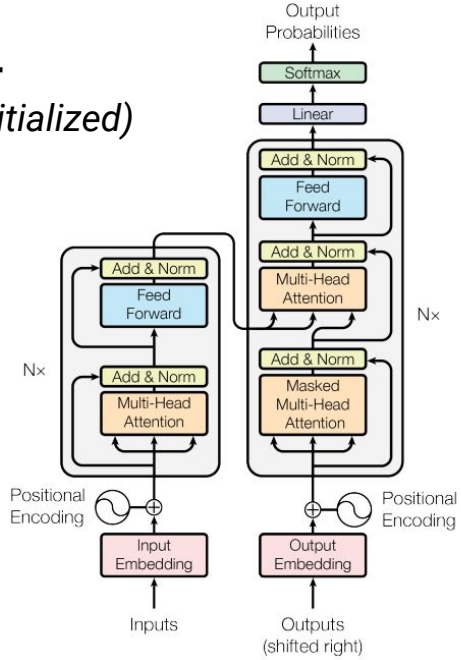
Task: Generate concise natural language description of the solution by synthesizing relevant content in the discussion when it emerges in real-time

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

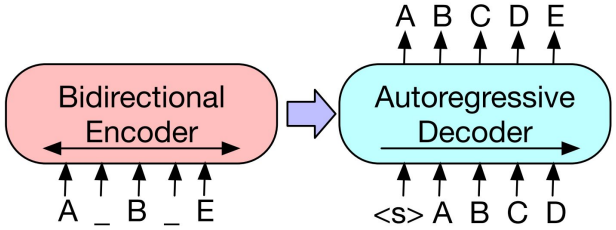
Copy Title: *Brief description of problem (e.g., Incorrect Distance)*

Transformer
(randomly initialized)



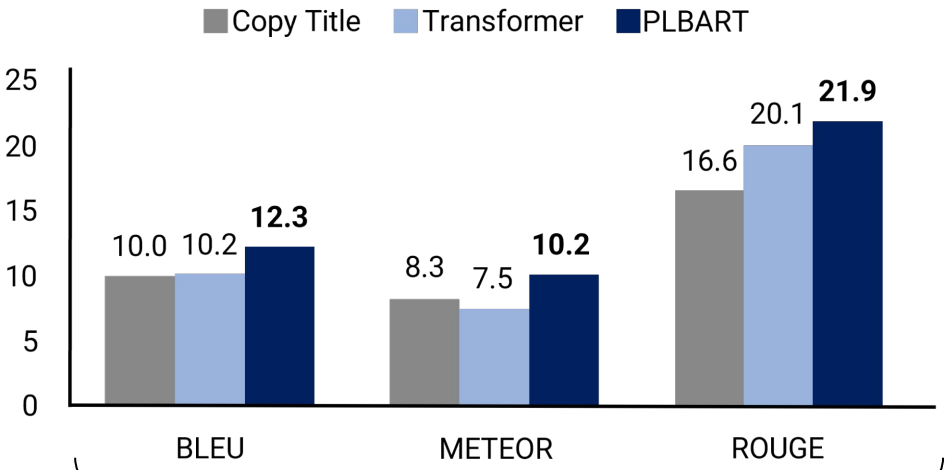
[Figure from *Vaswani et al., 2017*]

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

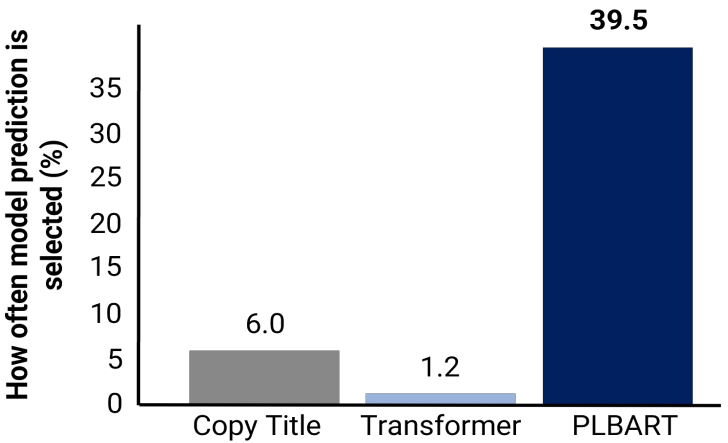


[Figure from Lewis et al., 2020]

Results: Generating Solution Descriptions



Automated metrics



Human evaluation
(8 annotators, 160 examples)
"Select the most <u>informative</u> generated description(s)"

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

PLBART [Ahmad et al., 2021]
METEOR [Banerjee and Lavie 2005]; BLEU-4 [Papineni et al., 2002]; ROUGE [Lin et al., 2004]

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`

Commit message/PR title

Time step of commit/PR

devC added a commit that referenced this issue

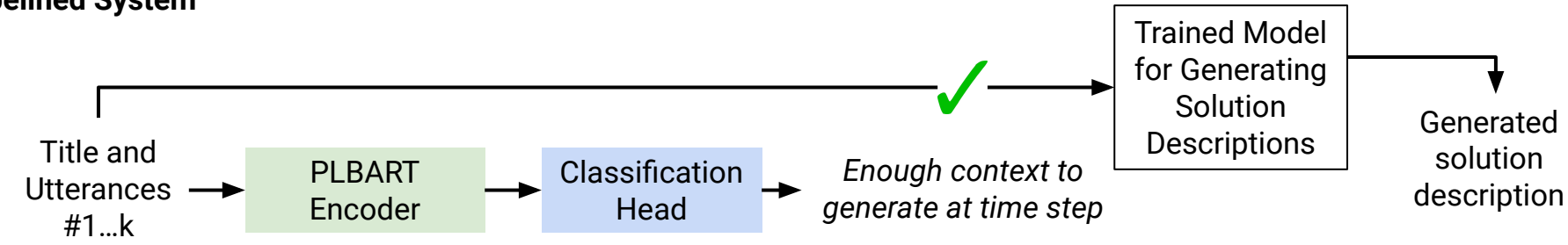
Solution is often formulated in discussion but buried under large amount of text.

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

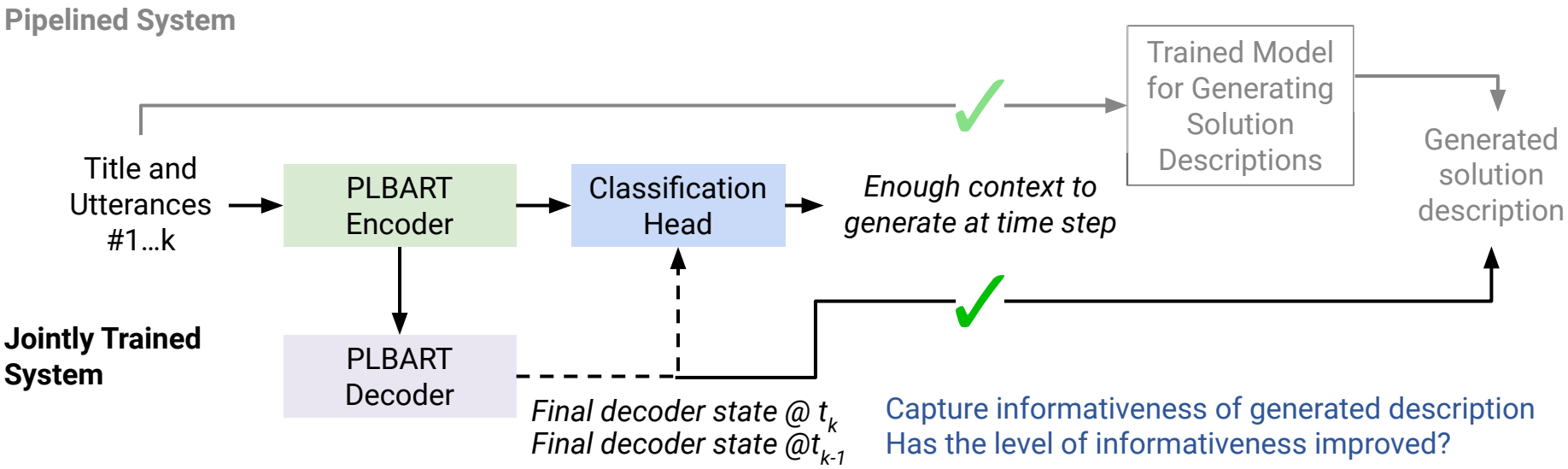
Data: 12K bug reports reports for open-source Java projects from GitHub Issues which are linked to a commit/PR

Generating Solution Descriptions in Real-Time

Pipelined System



Generating Solution Descriptions in Real-Time



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)	**Pipelined**	**Joint**

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)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%

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)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%
Is there sufficient context about the solution at time step k?	39.0%	33.8%

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)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%
Is there sufficient context about the solution at time step k?	39.0%	33.8%
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

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

Human Evaluation (60 annotators, 120 examples)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%
Is there sufficient context about the solution at time step k?	39.0%	33.8%
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
Scenario #2: System refrains from generating	35.4%	36.4%

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

Human Evaluation (60 annotators, 120 examples)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%
Is there sufficient context about the solution at time step k?	39.0%	33.8%
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
Scenario #2: System refrains from generating	35.4%	36.4%
Is there sufficient context about the solution at any point in the discussion?	34.2%	37.0%

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

Human Evaluation (60 annotators, 120 examples)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%
Is there sufficient context about the solution at time step k?	39.0%	33.8%
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
Scenario #2: System refrains from generating	35.4%	36.4%
Is there sufficient context about the solution at any point in the discussion?	34.2%	37.0%

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

Human Evaluation (60 annotators, 120 examples)	**Pipelined**	**Joint**
Scenario #1: System generates at time step k	64.6%	63.6%
~~*Is there sufficient context about the solution at time step k?*~~	~~39.0%~~	~~33.8%~~
*Is there **NOT** sufficient context about the solution at time step k?*	61.0%	66.2%
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
Scenario #2: System refrains from generating	35.4%	36.4%
Is there sufficient context about the solution at any point in the discussion?	34.2%	37.0%

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

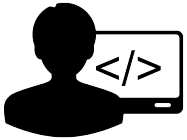
devA (Utterance #1)
Seeing negative distance when using 1D grid.

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

Bug-fixing code changes

```
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

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

→

Automated Bug-Fixing Model

→

Fixed Code

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

+

Full Buggy Method

```
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

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

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

MODIT [Chakraborty and Ray, 2021]

Deriving NL Context from Bug Report Discussions

Title: Parsing exception messages contain trailing newlines

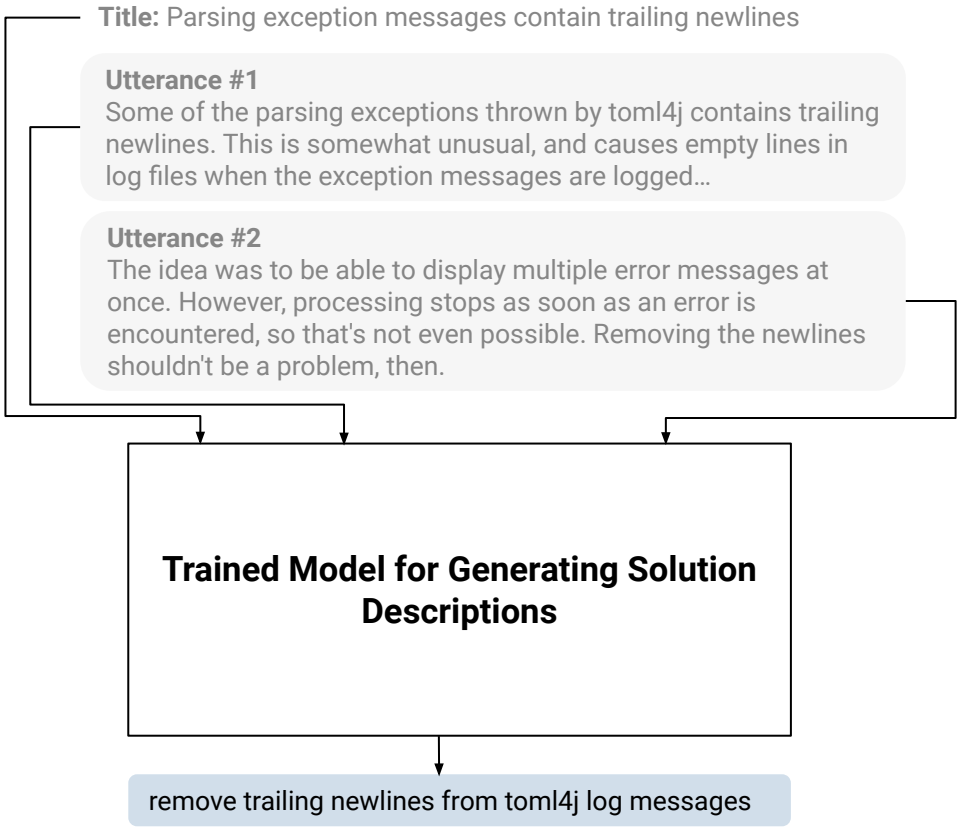
Utterance #1
Some of the parsing exceptions thrown by toml4j contains trailing newlines. This is somewhat unusual, and causes empty lines in log files when the exception messages are logged…

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



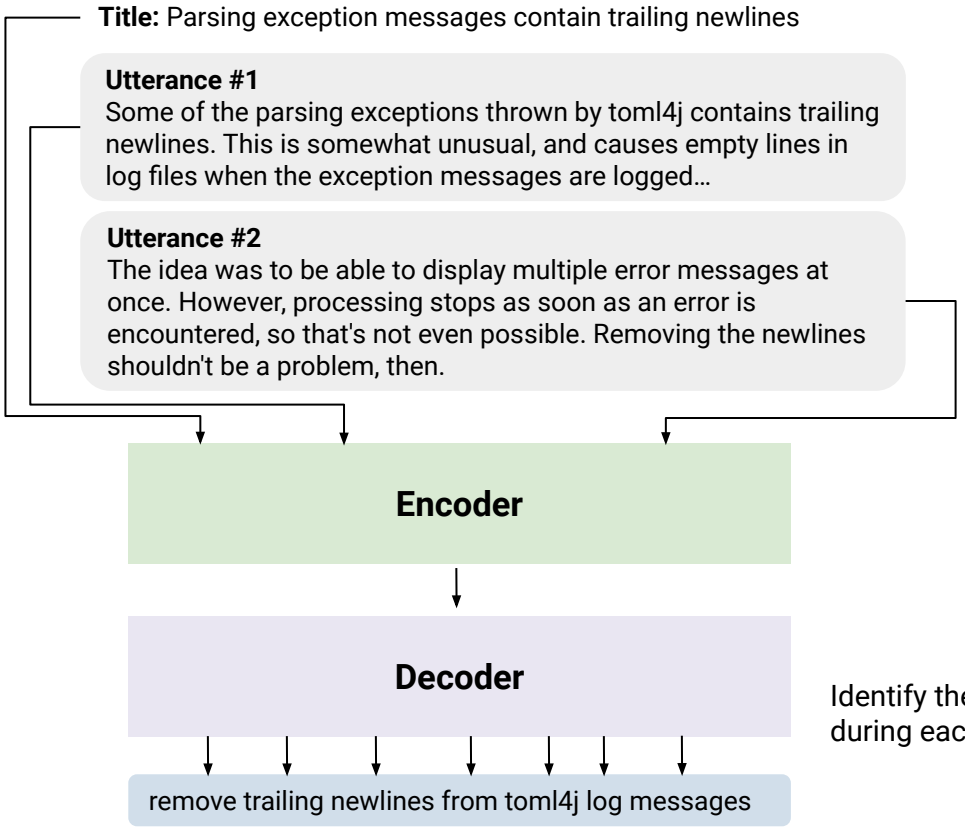
Deriving Context **Heuristically**

- Whole discussion
- Title
- Last utterance

Deriving Context **Algorithmically**

- Solution description
- Solution description + title

Deriving NL Context from Bug Report Discussions



Deriving Context **Heuristically**

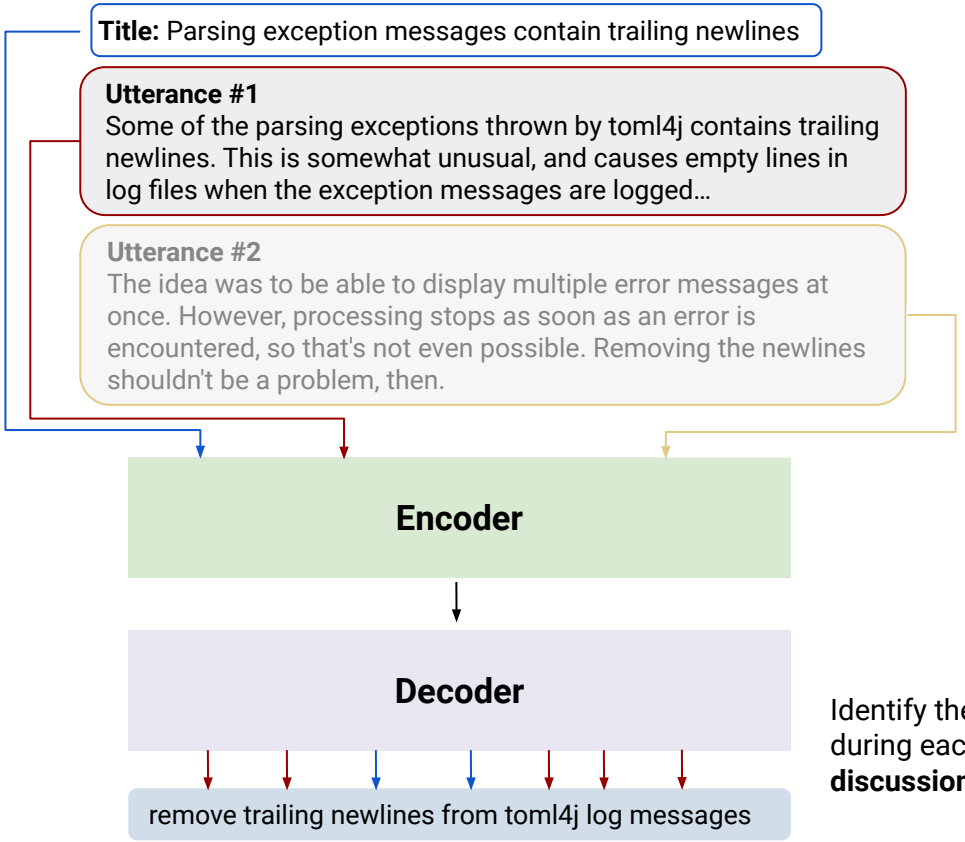
- Whole discussion
- Title
- Last utterance

Deriving Context **Algorithmically**

- Solution description
- Solution description + title

Identify the <u>most highly attended</u> input token during each step of decoding

Deriving NL Context from Bug Report Discussions



Deriving Context **Heuristically**

- Whole discussion
- Title
- Last utterance

Deriving Context **Algorithmically**

- Solution description
- Solution description + title
- Attended segments

Identify the <u>most highly attended</u> input token during each step of decoding **and the discussion segment to which it belongs**

Approach

Buggy Code

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

→ **Automated Bug-Fixing Model** →

Fixed Code

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

+

Full Buggy Method

```
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

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

→ **PLBART** →

Fixed Code

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

+

Full Buggy Method

```
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

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

→ **PLBART** →

Fixed Code

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

+

Full Buggy Method

```
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

Oracle commit message

MODIT [Chakraborty and Ray, 2021]

Approach

Buggy Code

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

→ **PLBART** →

Fixed Code

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

+

Full Buggy Method

```
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~~

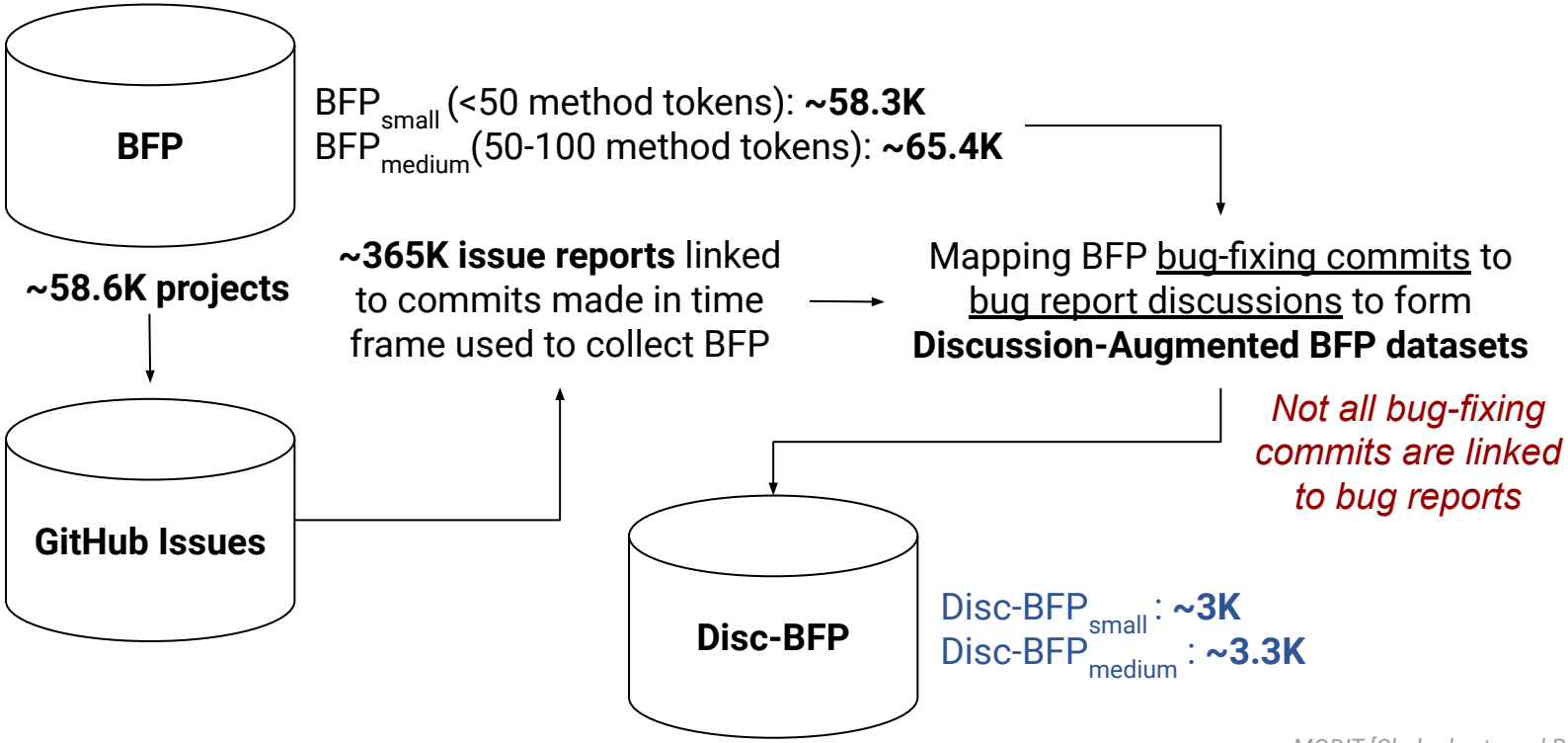
~~**Oracle commit message**~~

NL Context derived from bug report discussion:

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

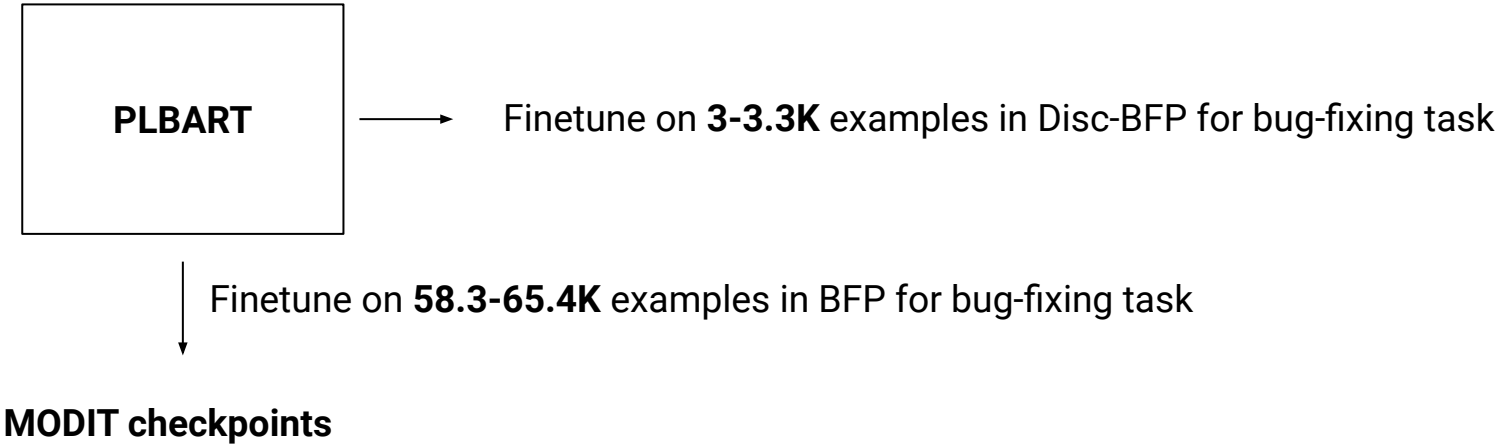
Data

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



MODIT [Chakraborty and Ray, 2021]

Initializing Model Parameters



MODIT [Chakraborty and Ray, 2021]

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

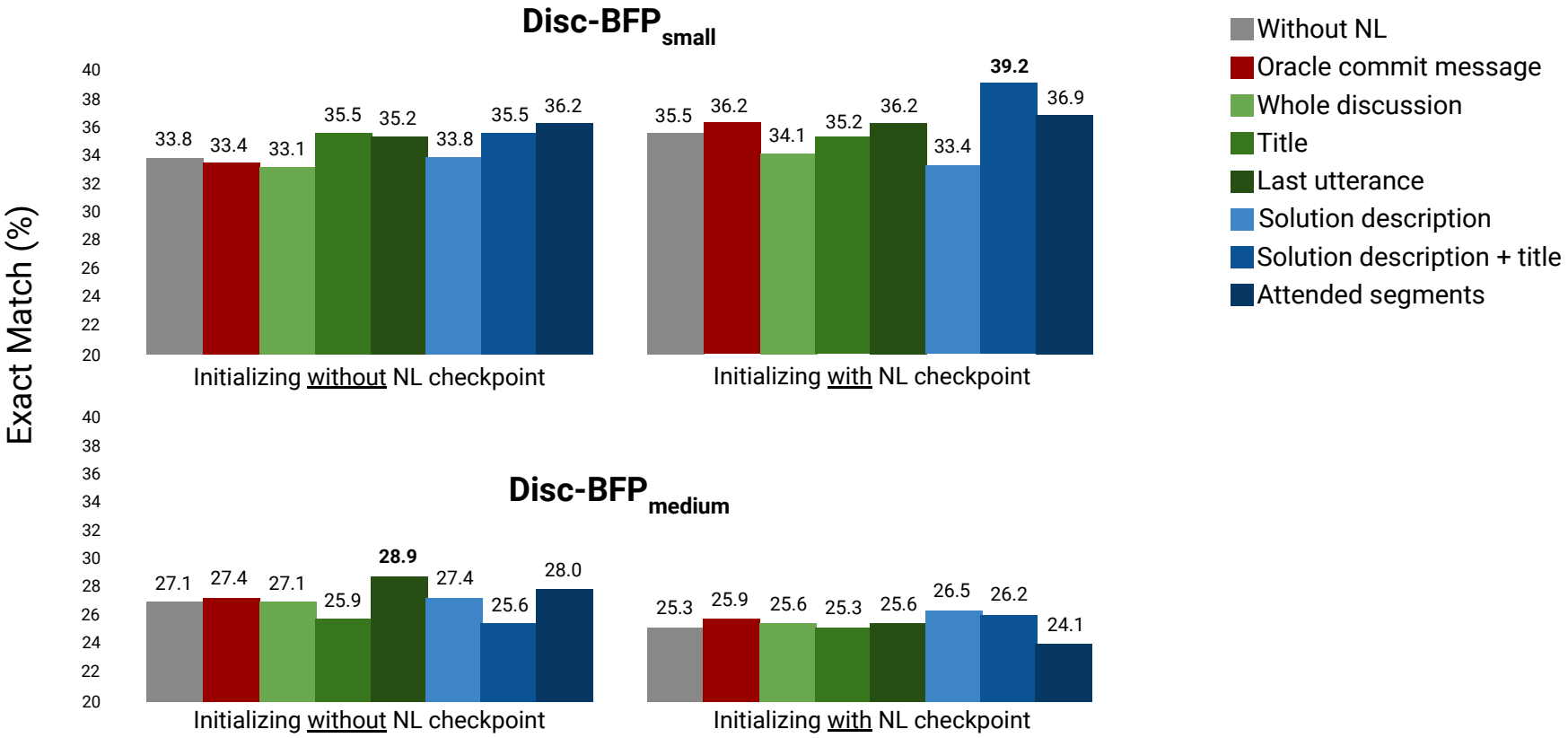
- **Without NL:** buggy + method
- **With NL:** buggy + method + oracle commit message

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

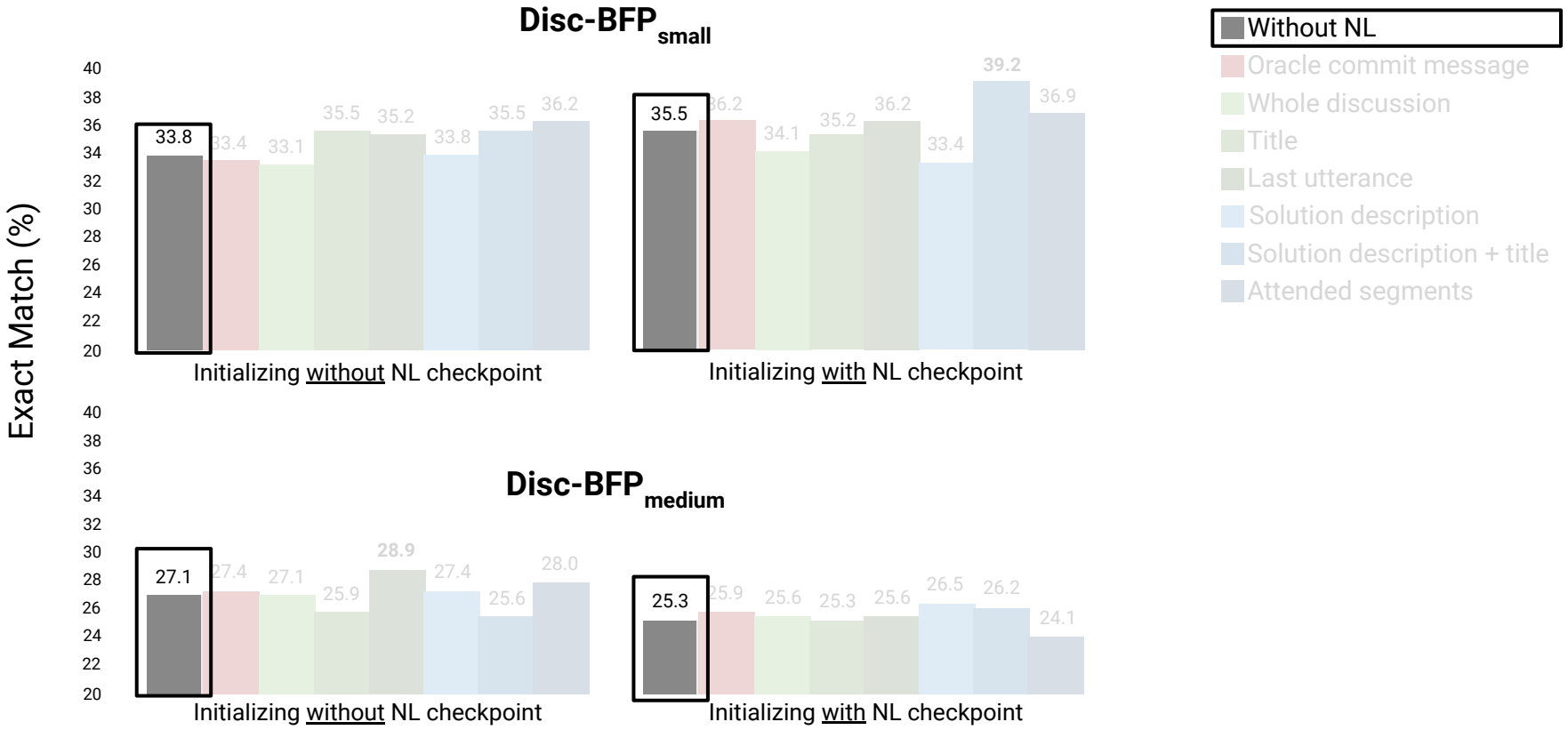
Our Models + Baselines

MODIT [Chakraborty and Ray, 2021]

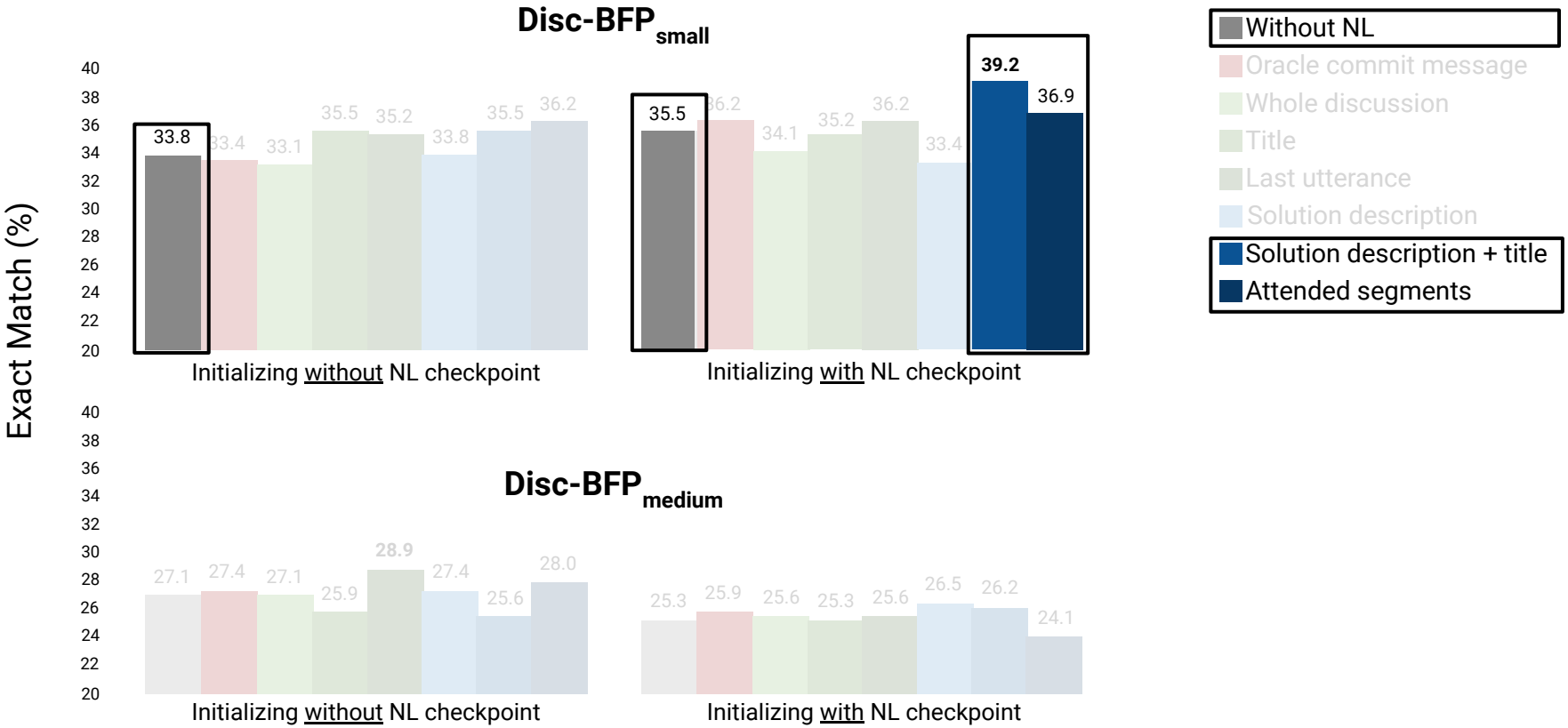
Results


Disc-BFP small
Exact Match (%)
40
38
36
34
32
30
28
26
24
22
20
33.8
33.4
33.1
35.5
35.2
33.8
35.5
36.2
35.5
36.2
34.1
35.2
36.2
33.4
39.2
36.9
Initializing without NL checkpoint
Initializing with NL checkpoint
Disc-BFP medium
27.1
27.4
27.1
25.9
28.9
27.4
25.6
28.0
25.3
25.9
25.6
25.3
25.6
26.5
26.2
24.1
Initializing without NL checkpoint
Initializing with NL checkpoint
Without NL
Oracle commit message
Whole discussion
Title
Last utterance
Solution description
Solution description + title
Attended segments

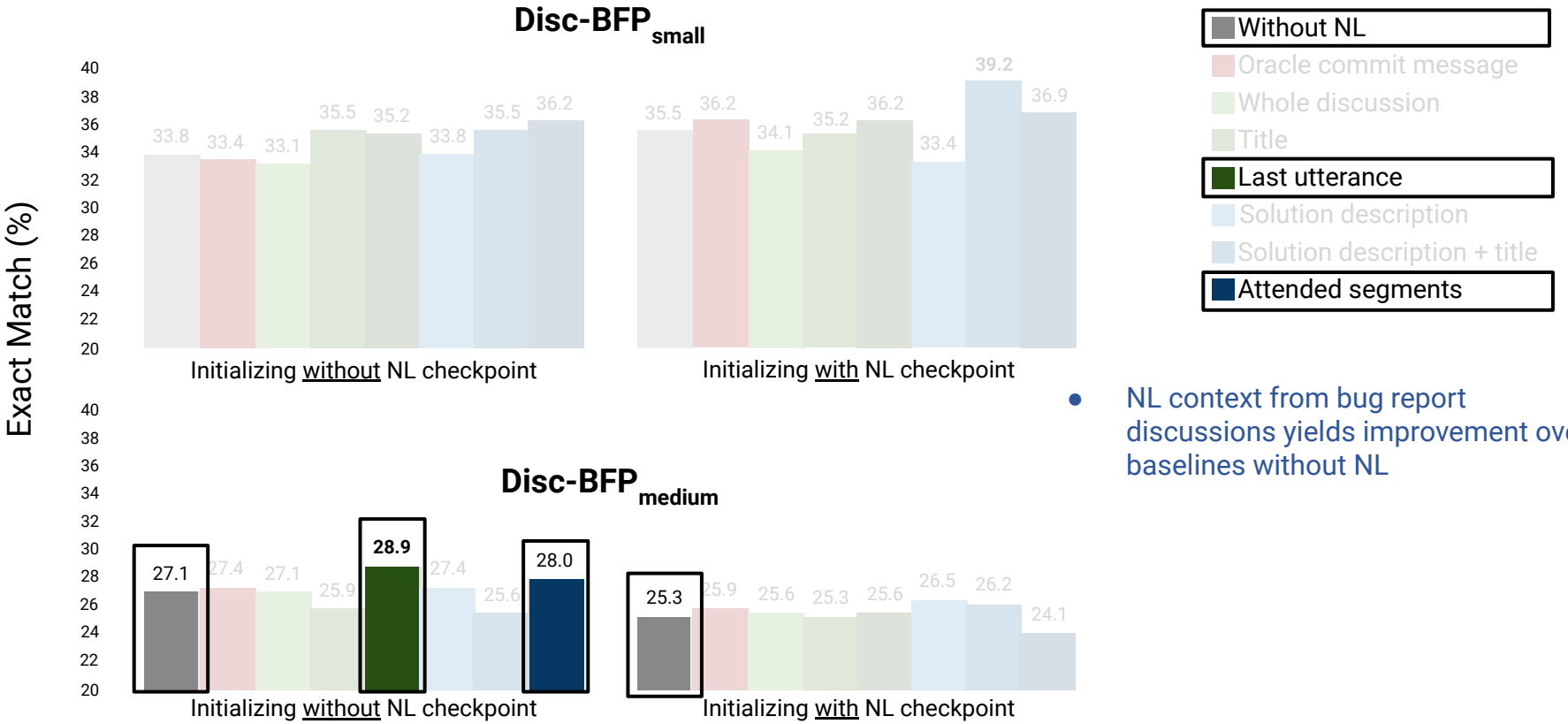
Results


Disc-BFP small
Exact Match (%)
40
38
36
34
32
30
28
26
24
22
20
33.8
33.4
33.1
35.5
35.2
33.8
35.5
36.2
35.5
36.2
34.1
35.2
36.2
33.4
39.2
36.9
Initializing without NL checkpoint
Initializing with NL checkpoint
Disc-BFP medium
27.1
27.4
27.1
25.9
28.9
27.4
25.6
28.0
25.3
25.9
25.6
25.3
25.6
26.5
26.2
24.1
Initializing without NL checkpoint
Initializing with NL checkpoint
Without NL
Oracle commit message
Whole discussion
Title
Last utterance
Solution description
Solution description + title
Attended segments

Results

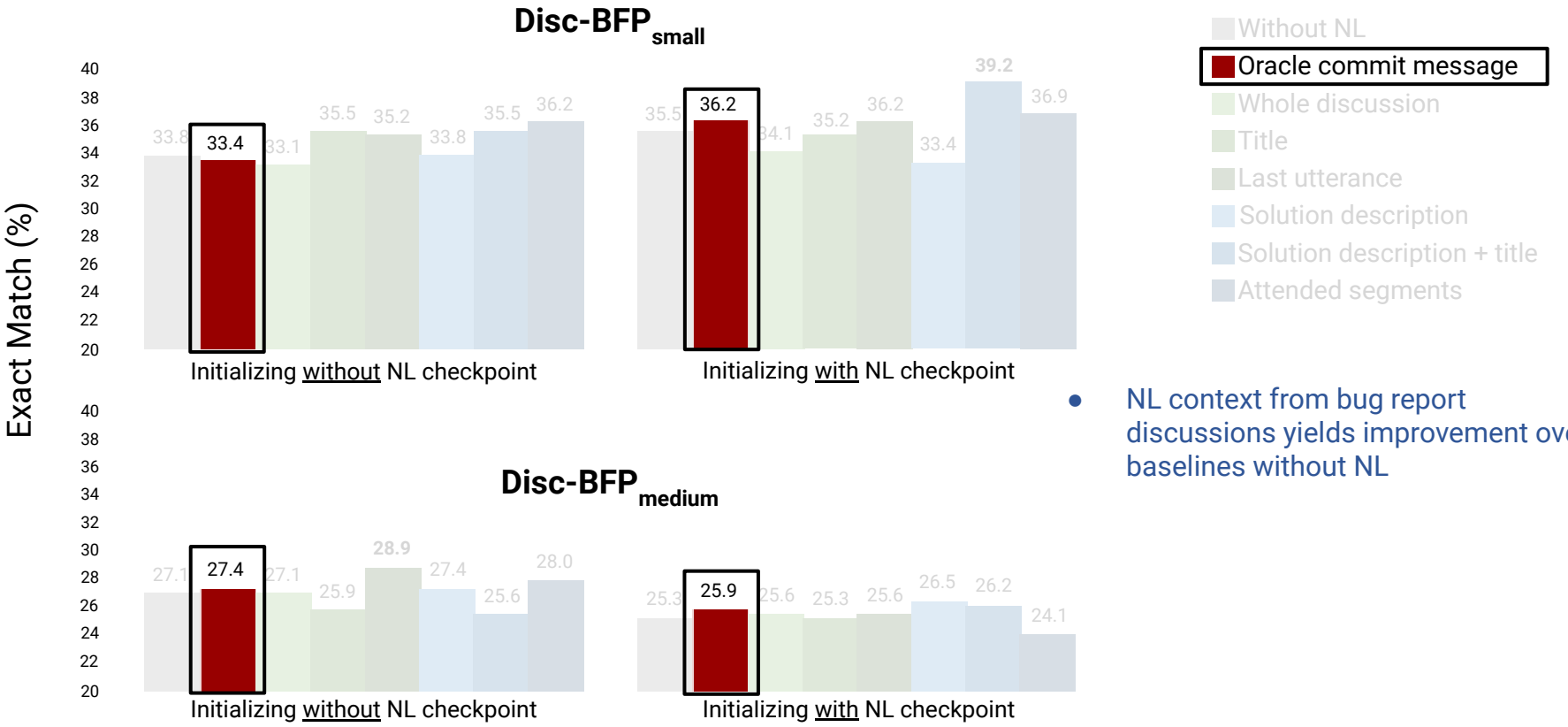

Disc-BFP small
Exact Match (%)
40
38
36
34
32
30
28
26
24
22
20
33.8
33.4
33.1
35.5
35.2
33.8
35.5
36.2
35.5
36.2
34.1
35.2
36.2
33.4
39.2
36.9
Initializing without NL checkpoint
Initializing with NL checkpoint
Disc-BFP medium
27.1
27.4
27.1
25.9
28.9
27.4
25.6
28.0
25.3
25.9
25.6
25.3
25.6
26.5
26.2
24.1
Initializing without NL checkpoint
Initializing with NL checkpoint
Without NL
Oracle commit message
Whole discussion
Title
Last utterance
Solution description
Solution description + title
Attended segments

Results



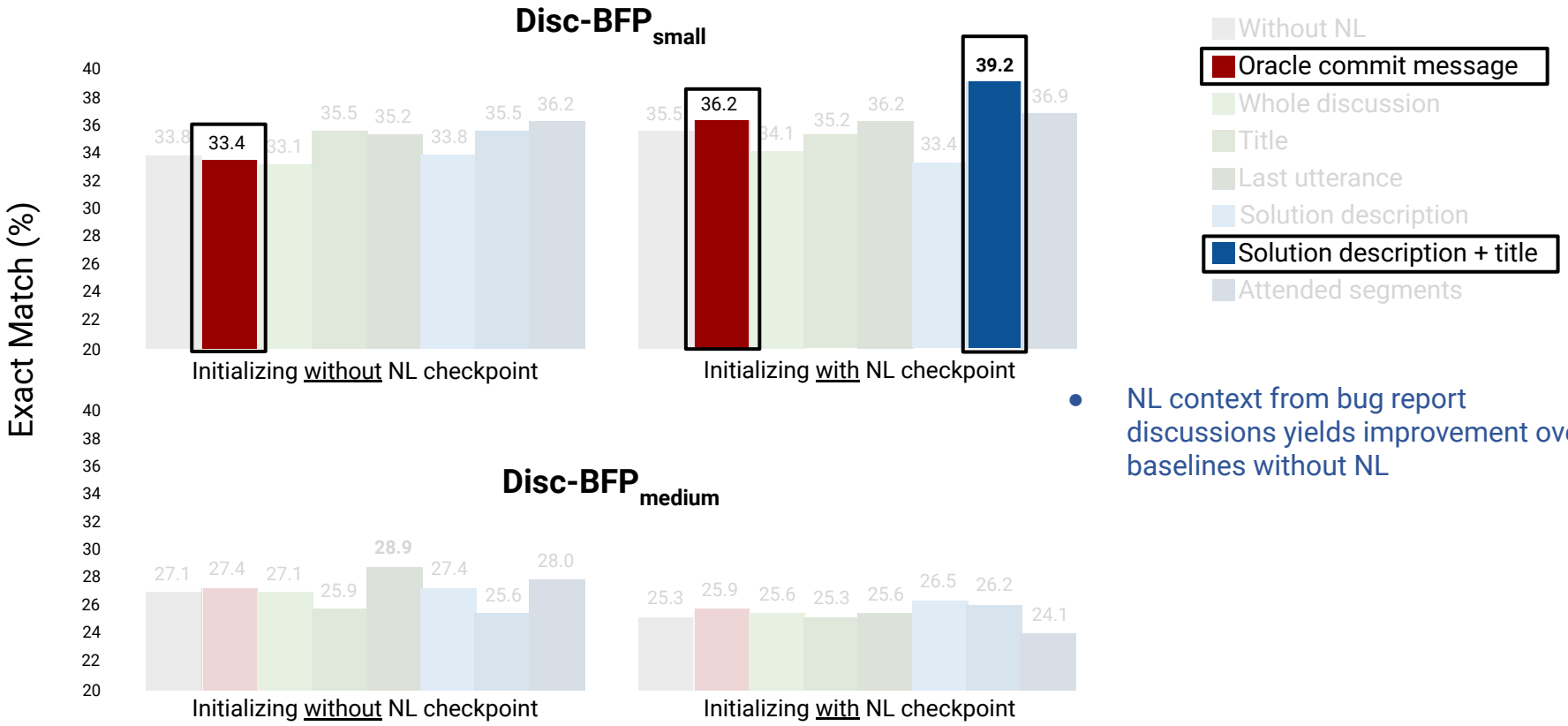
- NL context from bug report discussions yields improvement over baselines without NL

Results



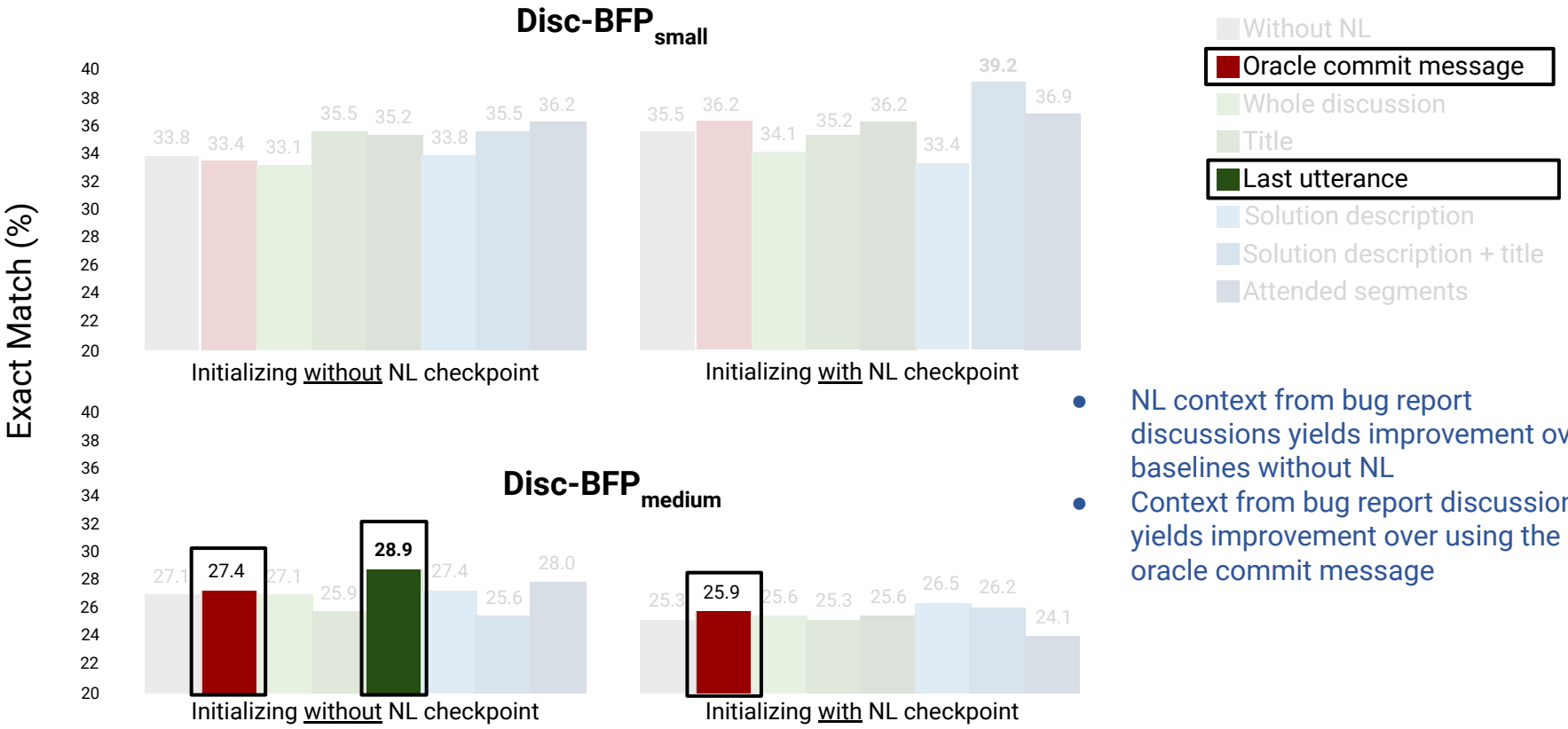
- NL context from bug report discussions yields improvement over baselines without NL

Results



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

Results



- 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

<u>Supporting</u> software evolution by upholding software quality amidst constant changes

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

→ **Code Edits Encoder** → **Task-Specific Decoder**

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]

Comment Inconsistency Detection/Update

✗ lowest

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

Commit Message Generation

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

PaLM: Scaling Language Modeling with Pathways [Chowdhery et al. 2022]

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

Commit message generation [Loyola et al., 2017; Xu et al., 2019]; Release note generation *[Moreno et al., 2014]*; Test suite update *[Daniel et al., 2011]*

Future Work: Driving Software Evolution

<u>Driving</u> software evolution by expediting critical code changes

```
/** 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("(%d)", distance));
+       log.debug(String.format("L1 Distance in 1D (%d)", distance));
}
```

Can an intelligent agent collaborate with human developers for more efficient/effective <u>code review</u>?

Future Work: Driving Software Evolution

Driving software evolution by expediting critical code changes

```
/** 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("(%d)", distance));
+       log.debug(String.format("L1 Distance in 1D (%d)", distance));
}
```

Can an intelligent agent collaborate with human developers for more efficient/effective code review?

By interactively providing…

- PR review comments

Future Work: Driving Software Evolution

Driving software evolution by expediting critical code changes

```
/** 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("(%d)", distance));
+       log.debug(String.format("L1 Distance in 1D (%d)", distance));
}
```

Can an intelligent agent collaborate with human developers for more efficient/effective code review?

By interactively providing…

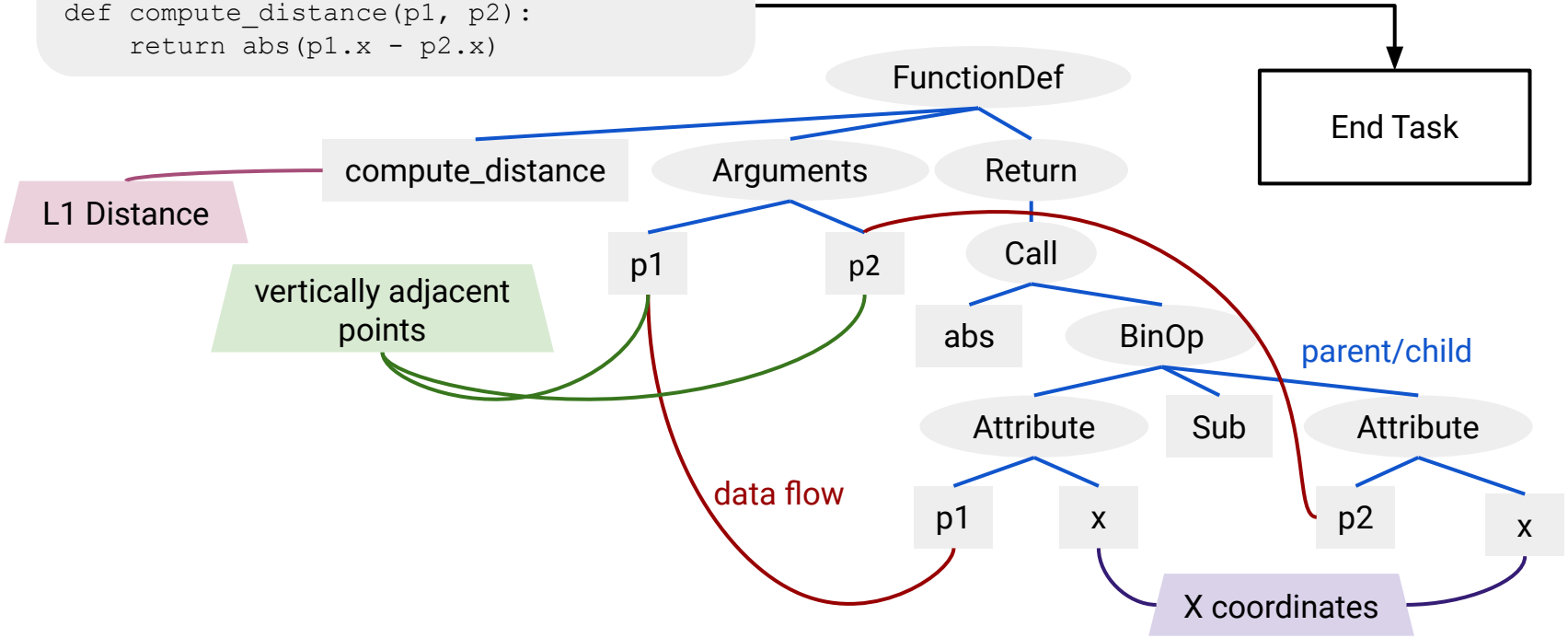
- PR review comments
  Suggested code changes

Future Work: Enhancing Code Representations with Natural Language

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

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

Code Representation



End Task

Acknowledgements

NSF

Google

Bloomberg