The University of Texas at Austin

Deep Just-In-Time Inconsistency Detection Between Comments and Source Code

Sheena Panthaplackel, Junyi Jessy Li, Milos Gligoric, Raymond J. Mooney

Source Code Comments

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

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

Source Code Comments

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

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

```
-        return Collections.max(scores);
+        return Collections.min(scores);
```

✗

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

Leads to time-wasting confusion and vulnerability to bugs

Automatically detect inconsistent comments

Detecting Inconsistent Comments

Just-In-Time Inconsistency Detection

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

```
-       return Collections.max(scores);
+       return Collections.min(scores);
```

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

```
-    public int getBestScore() {
+    public double getBestScore() {
```

Post Hoc Inconsistency Detection

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

Inconsistent ✗

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

Consistent ✓

Outline

- Task
- Architecture
- Data
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Outline

- Task
- Architecture
- Data
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Task

Determine whether a comment (C) is inconsistent with a method (M)

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

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

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

Post Hoc: Given **C** and **M**

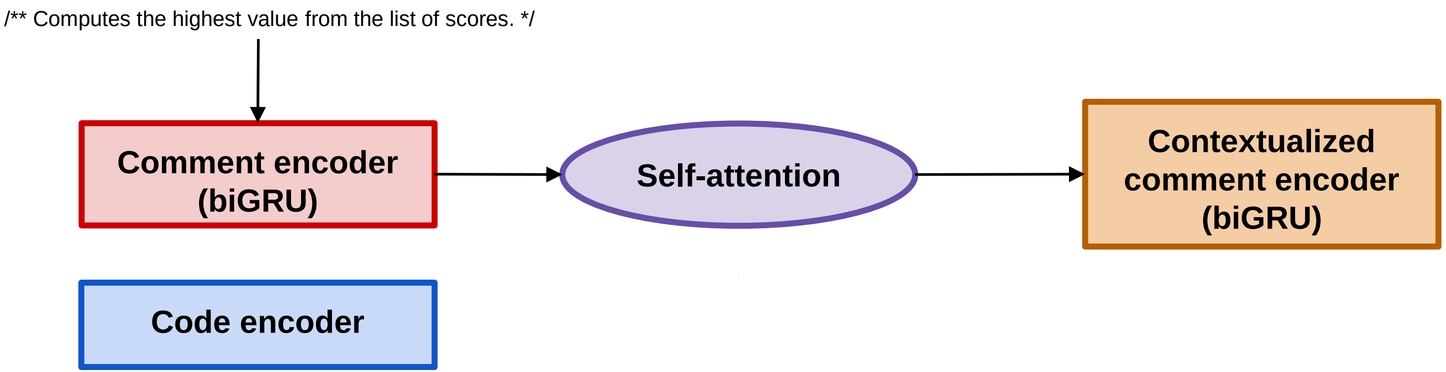
Just-In-Time: Given **C**, **M**, and M_{old}

The University of Texas at Austin

Outline

- Task
- Architecture
- Data
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Architecture



Code Representations

Sequence-based

Post hoc (**M**): M as a sequence of tokens

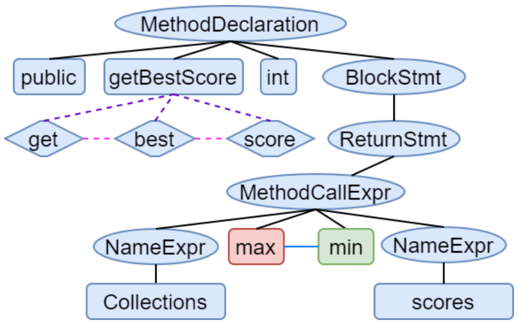
```
public int getBestScore ( ) { return Collections . min ( scores ) ; }
```

Just-In-Time ($\mathbf{M_{edit}}$): Edits between M_{old} and M as a sequence of tokens

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

AST-based

Just-In-Time ($\mathbf{T_{edit}}$):
Graph representation of AST node edits between M_{old} and M

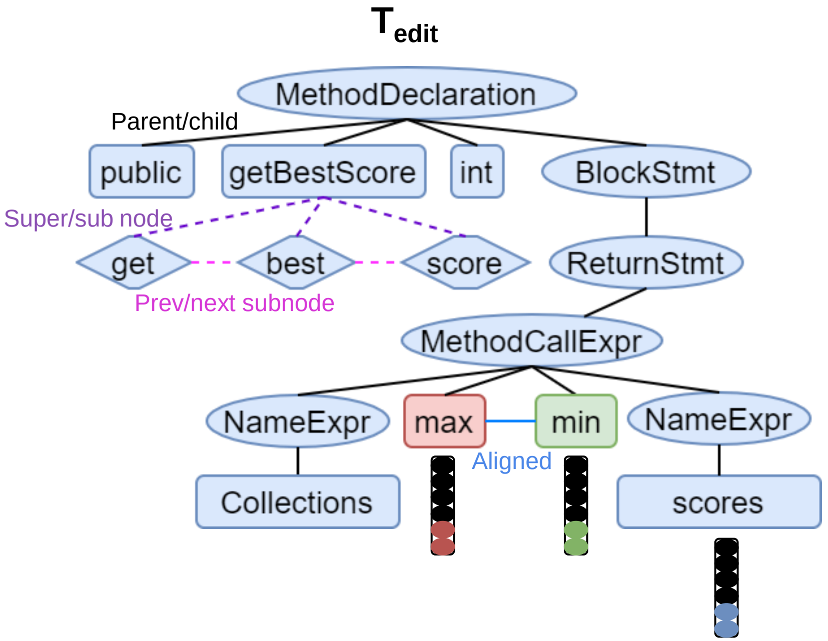


Encoding Code Representations

- Sequence-based representations (**M**, $\mathbf{M_{edit}}$) encoded with a biGRU.
- AST-based representations (**T**, $\mathbf{T_{edit}}$) encoded with GGNNs.

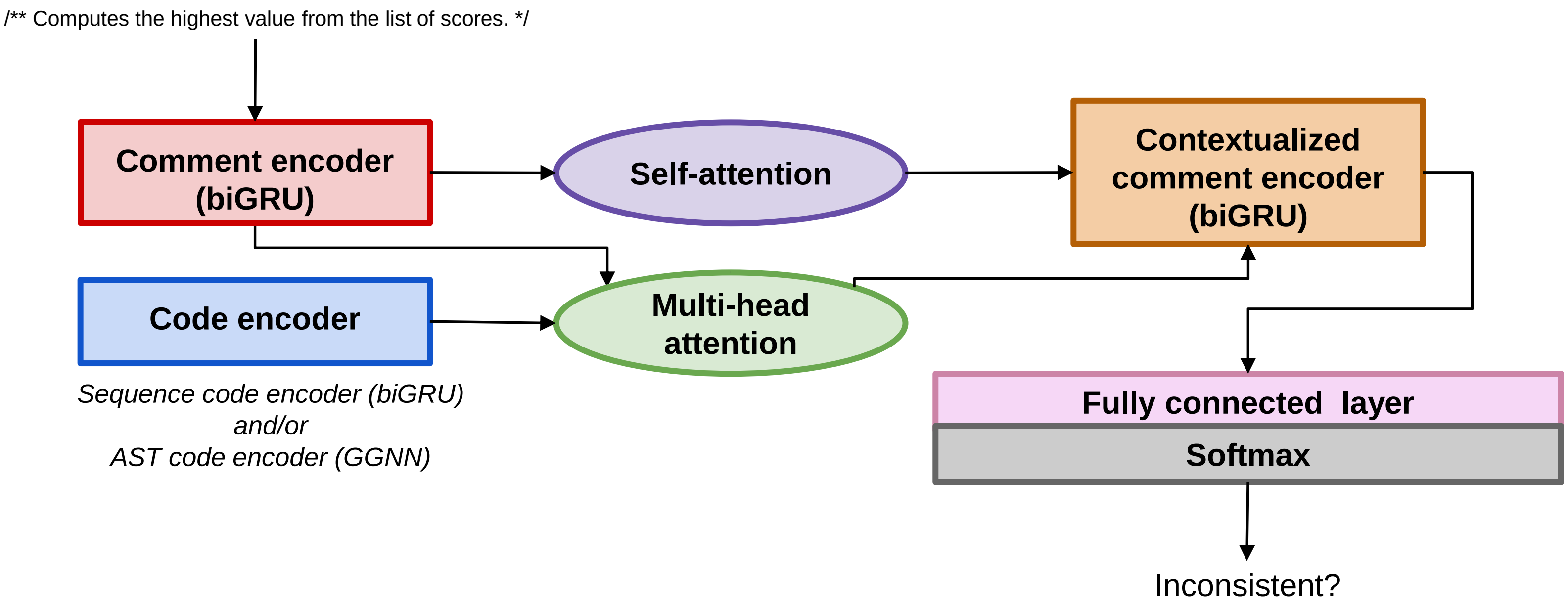
Gated Graph Neural Network (GGNN)

- Node state representations are updated across 8 steps of message passing
- Message passing is done through different edge types
- Initial state representations consist of a word embedding, concatenated with an edit embedding (for the just-in-time setting)



Gated Graph Sequence Neural Networks *(Li et al., ICLR 2016)*
Structured Neural Summarization *(Fernandes et al., ICLR 2019)*

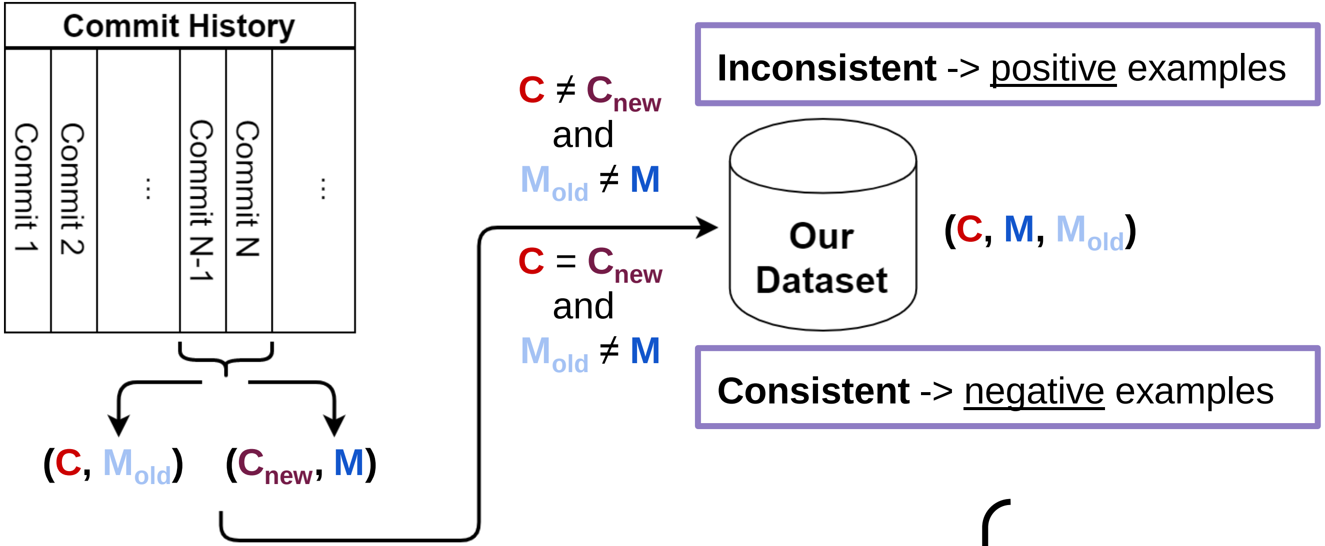
Architecture



Outline

- Task
- Architecture
- **Data**
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Data Collection



	Train	Valid	Test	Total
Examples	32,988	3,756	3,944	40,688
Projects	829	332	357	1,518

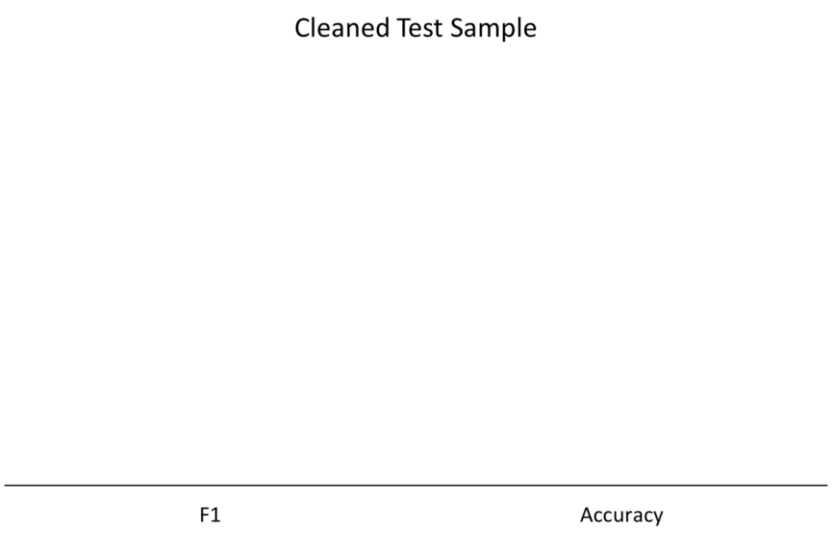
- @return, @param, and summary comments
- Balanced label distribution
- Cleaned 300 examples from test set for more reliable evaluation

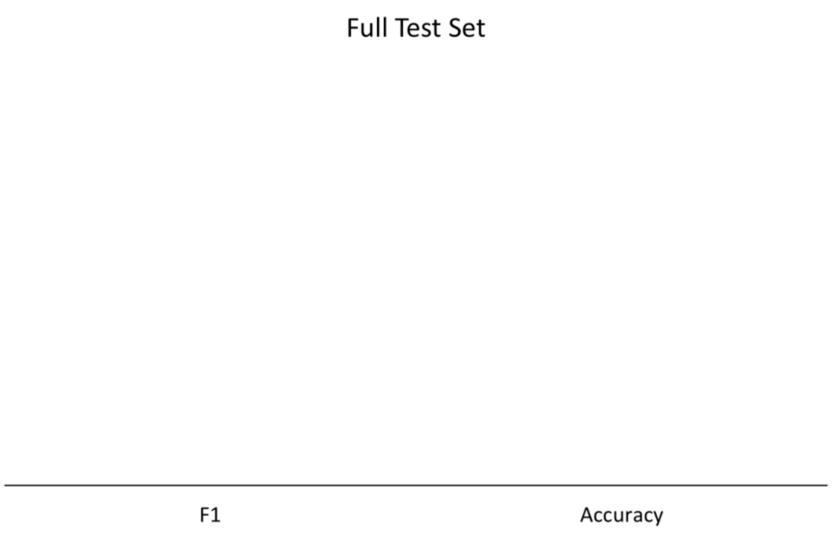
The University of Texas at Austin

Outline

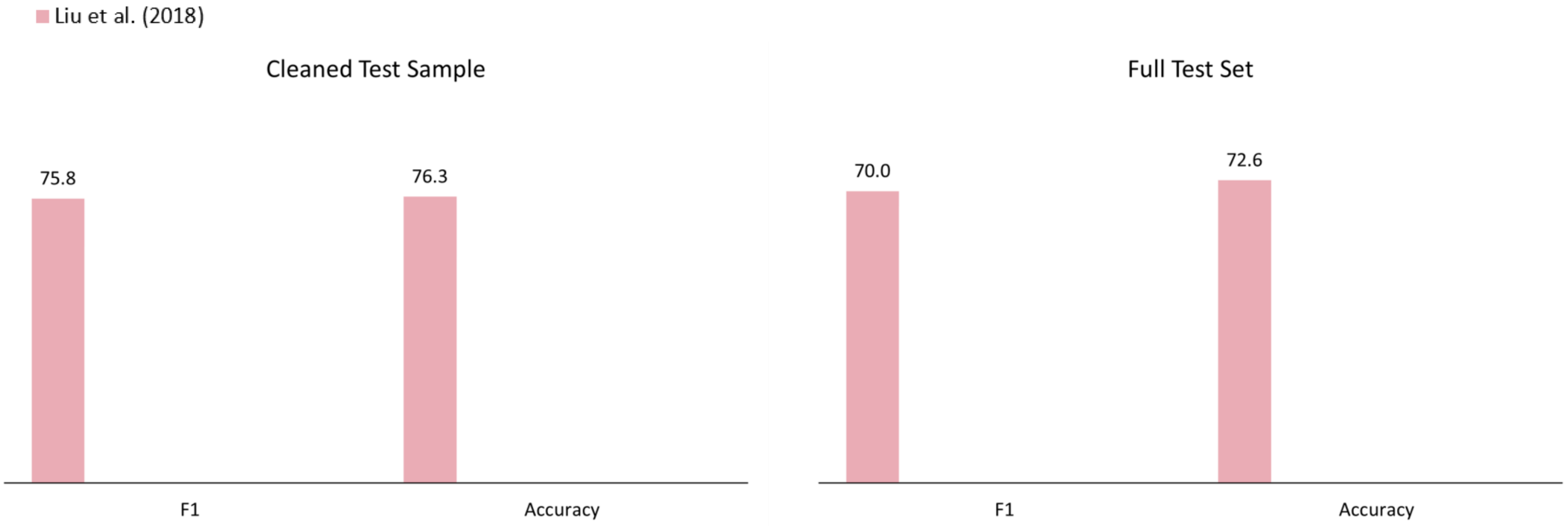
- Task
- Architecture
- Data
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Intrinsic Evaluation: Results

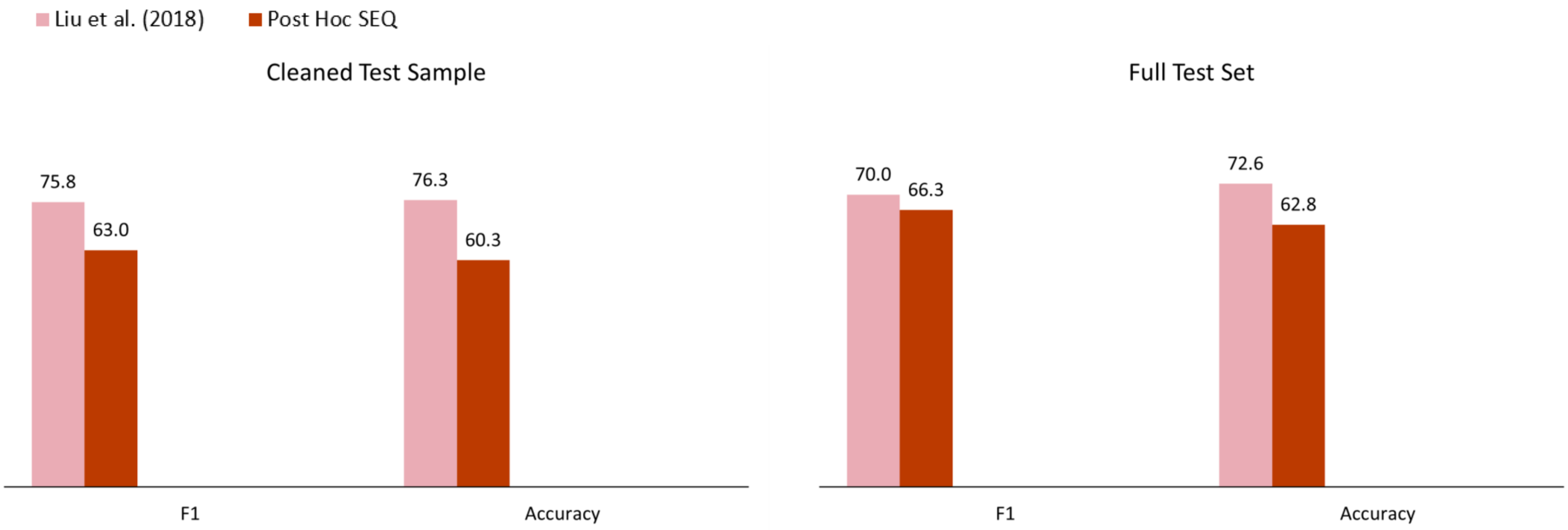




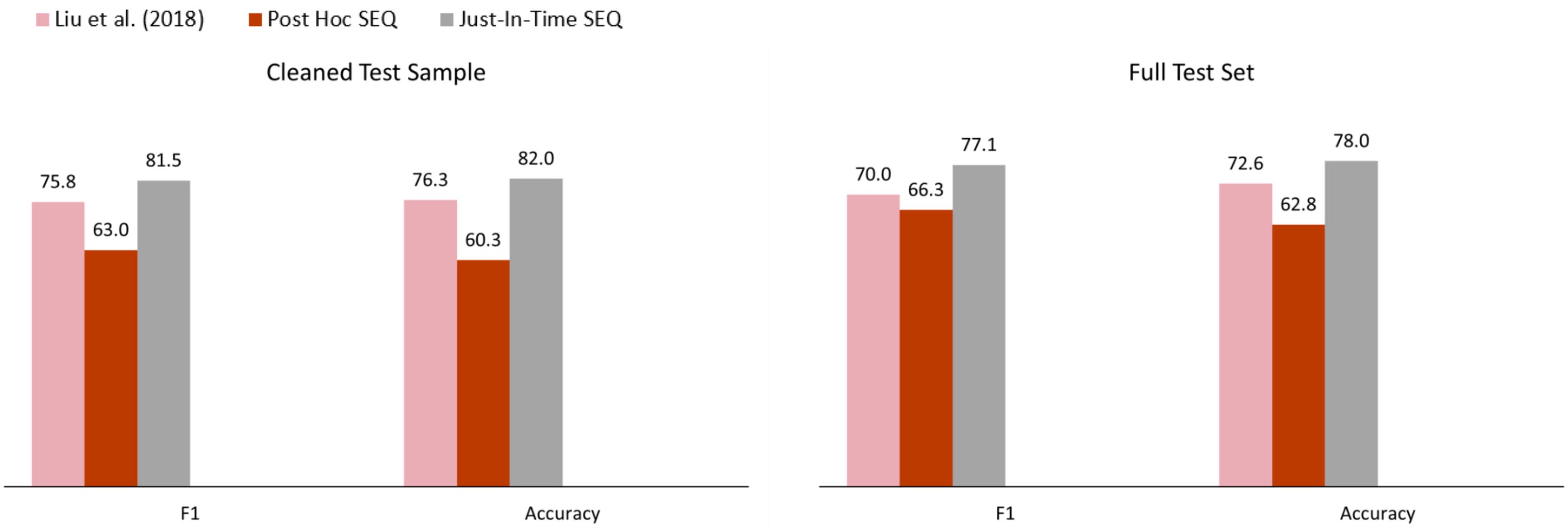
Intrinsic Evaluation: Results



Intrinsic Evaluation: Results

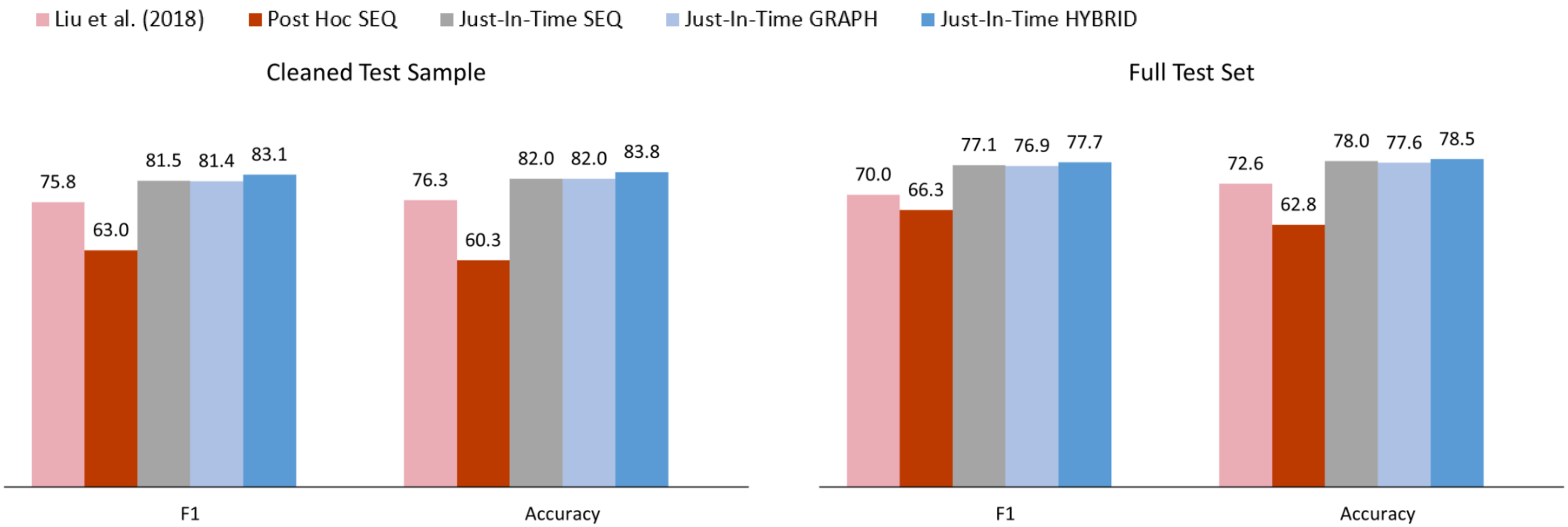


Intrinsic Evaluation: Results



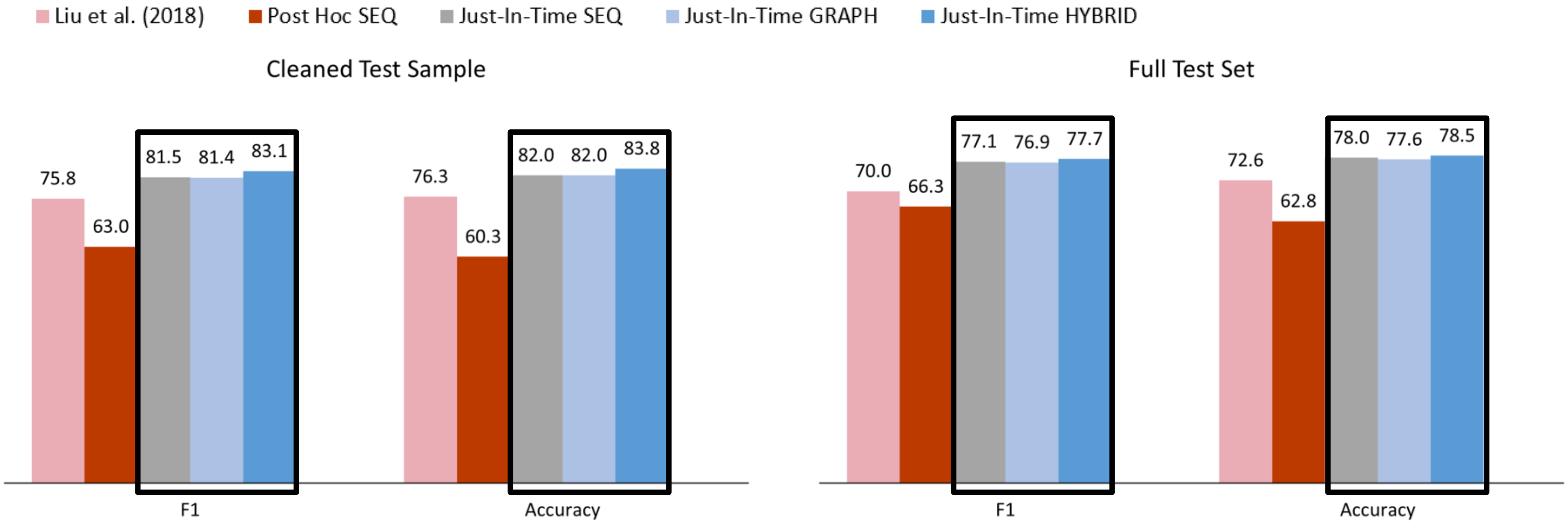
- Our Just-In-Time approach can outperform post hoc and baseline models

Intrinsic Evaluation: Results



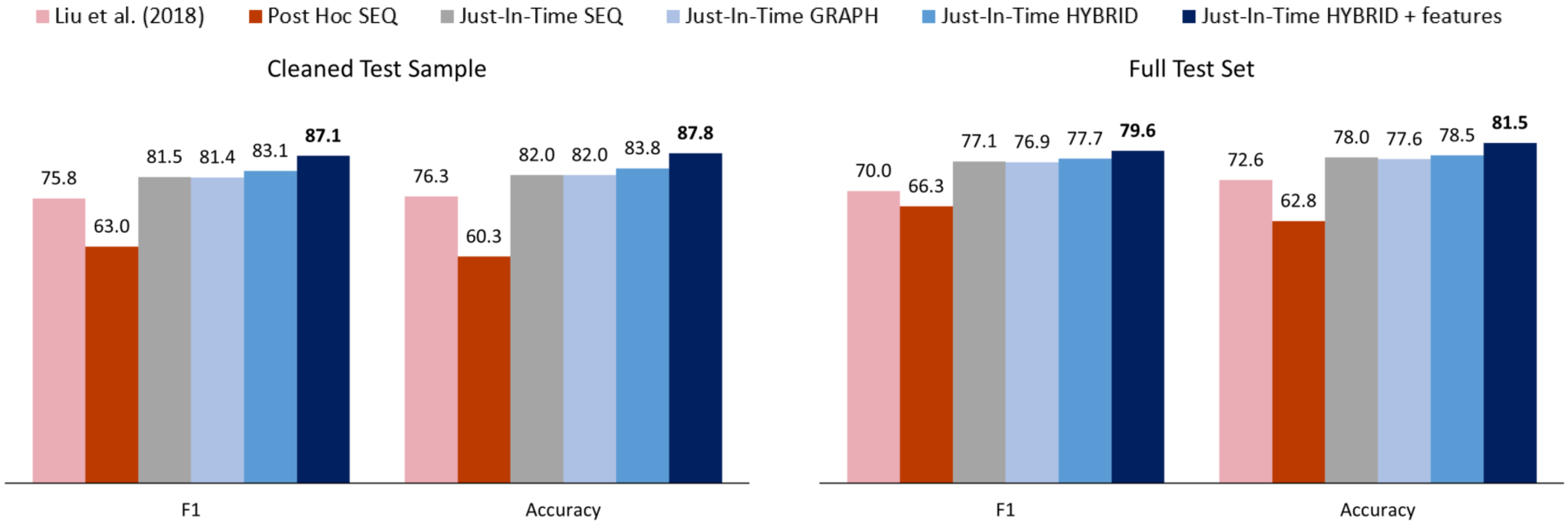
- Our Just-In-Time approach can outperform post hoc and baseline models

Intrinsic Evaluation: Results



- Our Just-In-Time approach can outperform post hoc and baseline models
- No significant difference between SEQ, GRAPH, and HYBRID approaches

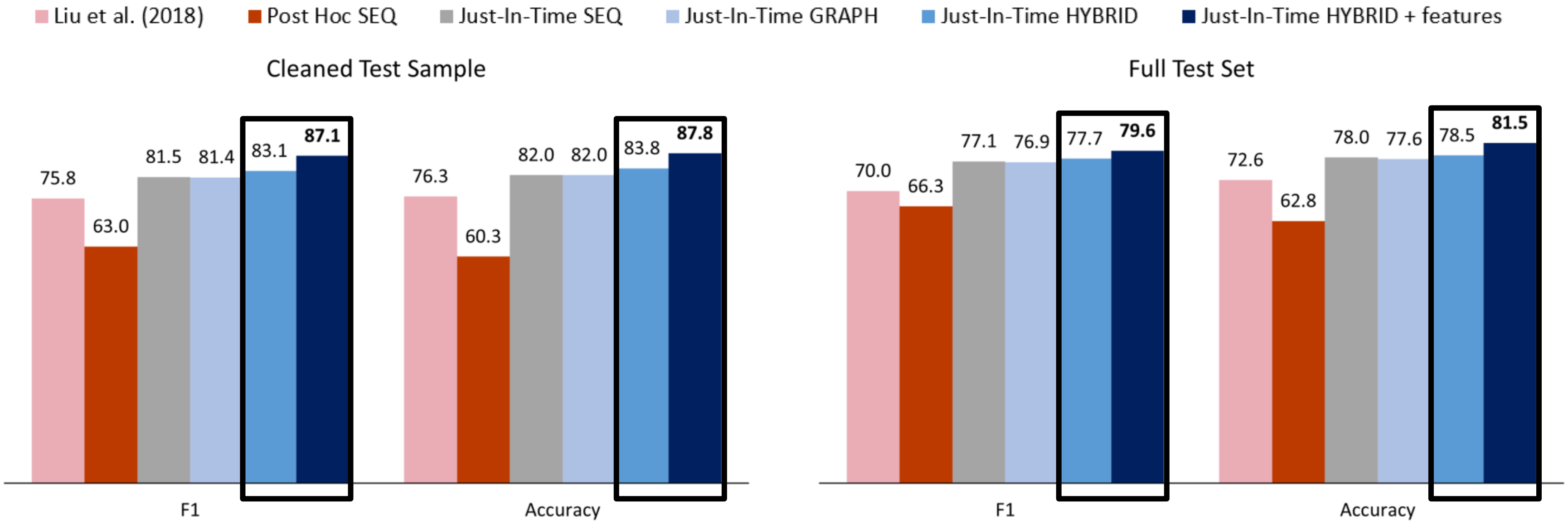
Intrinsic Evaluation: Results



- Our Just-In-Time approach can outperform post hoc and baseline models
- No significant difference between SEQ, GRAPH, and HYBRID approaches

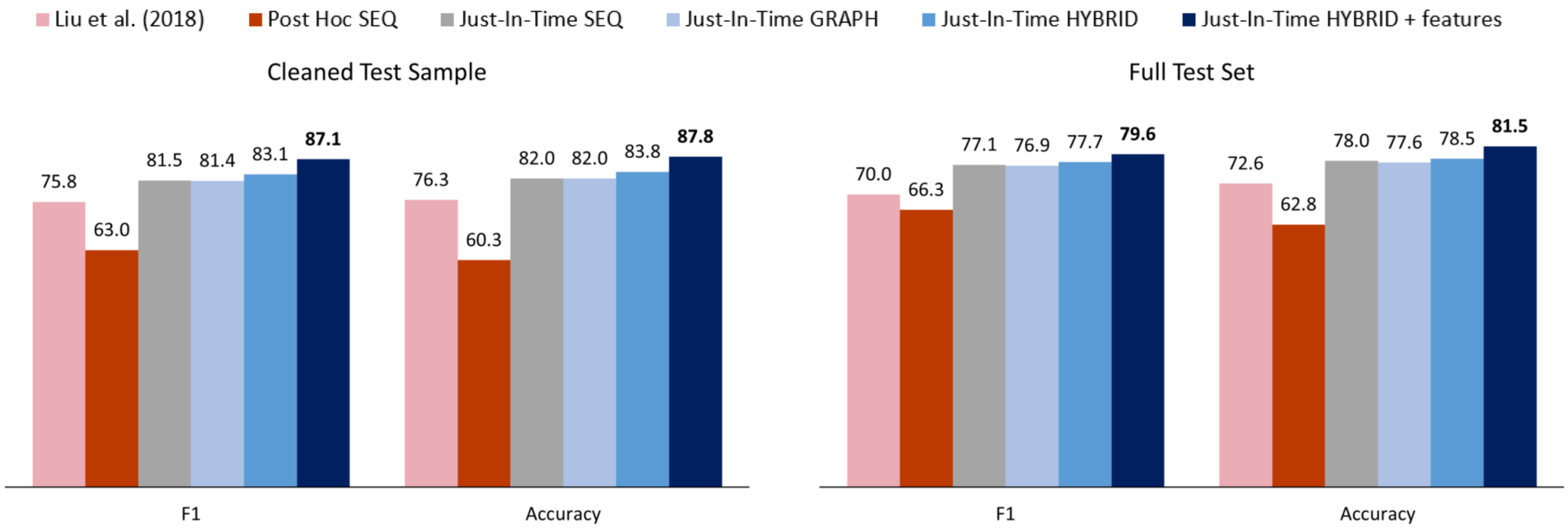
The University of Texas at Austin

Intrinsic Evaluation: Results



- Our Just-In-Time approach can outperform post hoc and baseline models
- No significant difference between SEQ, GRAPH, and HYBRID approaches
- Incorporating auxiliary features can further boost performance

Intrinsic Evaluation: Results



- Our Just-In-Time approach can outperform post hoc and baseline models
- No significant difference between SEQ, GRAPH, and HYBRID approaches
- Incorporating auxiliary features can further boost performance
- Analogous performance between cleaned and full test sets

Integrating with Update

On its own, inconsistency detection can only flag comments that developers failed to update.

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

```diff
-            return Collections.max(scores);
+            return Collections.min(scores);
```

✗

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

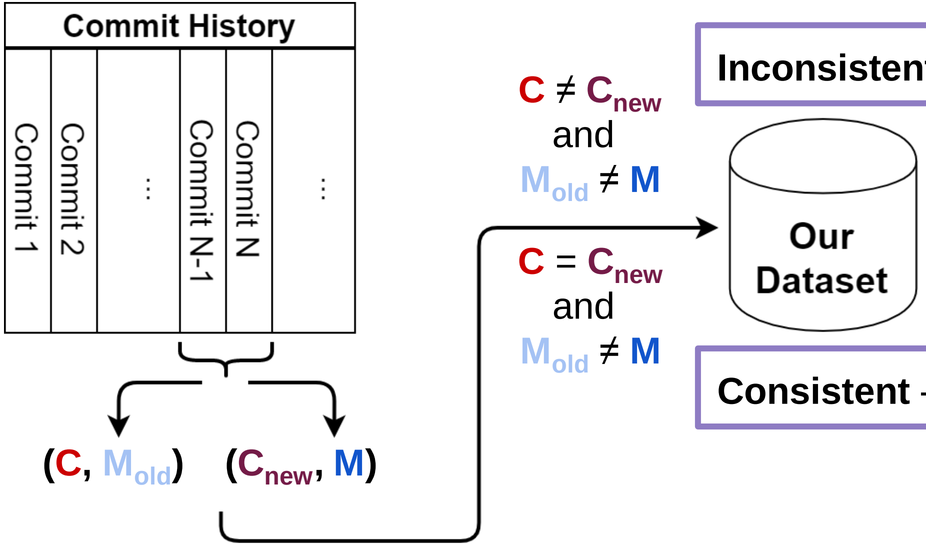
Automatically detect and update inconsistent comments

The University of Texas at Austin

Outline

- Task
- Architecture
- Data
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Extrinsic Evaluation: Integrating with Update



Inconsistent -> positive examples

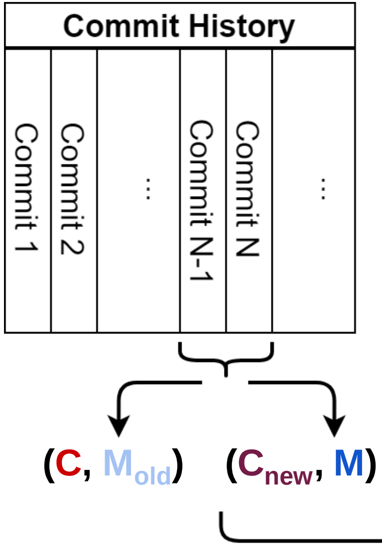
[(C_1, M_1), (C_2, M_2)] = [(C, M_{old}), (C_{new}, M)]

Given C, M_{old}, and M, produce C_{new}, which is consistent with M.

Consistent -> negative examples

(C, M_{old}), M -> C_{new} = { Generate C_{new} by updating C to reflect M *if C is inconsistent with M*; Copy C_{new} = C *else* }

Extrinsic Evaluation: Integrating with Update



$C \neq C_{new}$ and $M_{old} \neq M$

$C = C_{new}$ and $M_{old} \neq M$

Inconsistent -> <u>positive</u> examples

$[(C_1, M_1), (C_2, M_2)] = [(C, M_{old}), (C_{new}, M)]$

Given C, M_{old}, and M, produce C_{new}, which is consistent with M.

Consistent -> <u>negative</u> examples

Comment Update Model
(Panthaplackel et al., ACL 2020)

There are many cases in which code changes do not require a comment update.

(C, M_{old}), M -> C_{new} =
- Generate C_{new} by <u>updating</u> C to reflect M — *if C is inconsistent with M*
- Copy C_{new} = C — *else*

The University of Texas at Austin

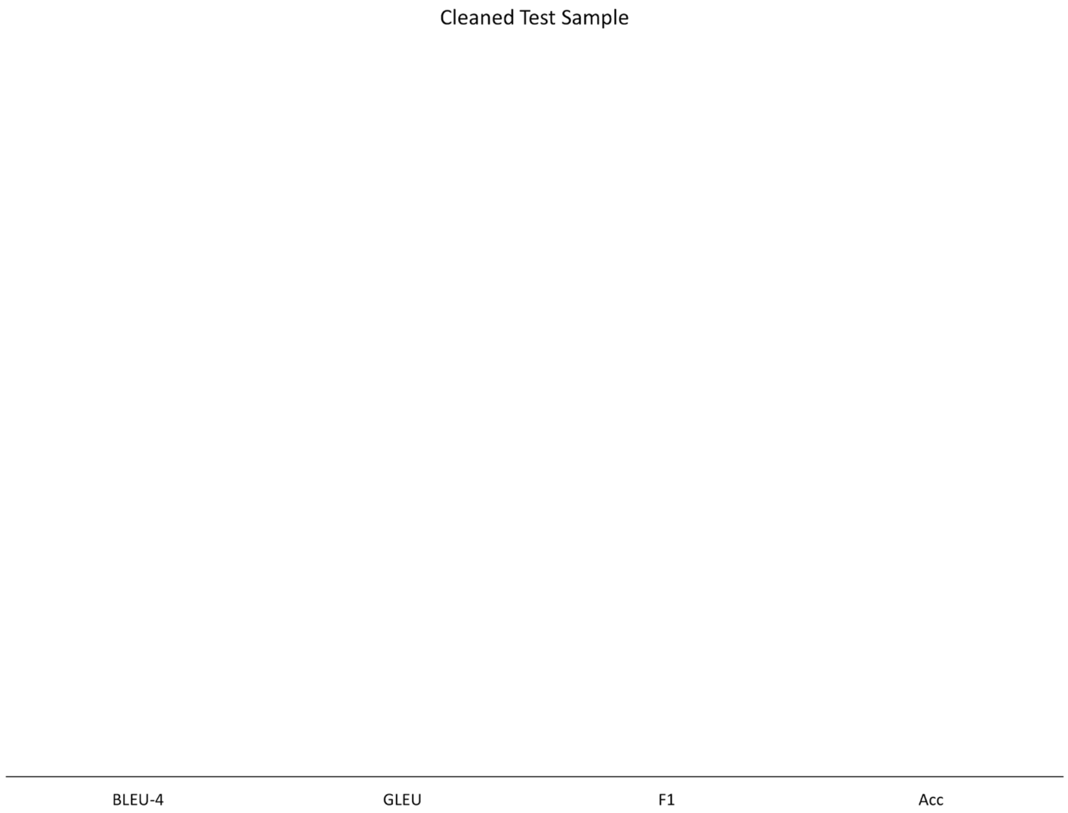
Extrinsic Evaluation: Integrating with Update

	Training		*Inference*
Update w/ Implicit Detection	+, - →	**Update** (Supervision: C_{new}) →	$P_{inconsistency\ label}$ = output(Update) != C $P_{comment}$ = output(Update)

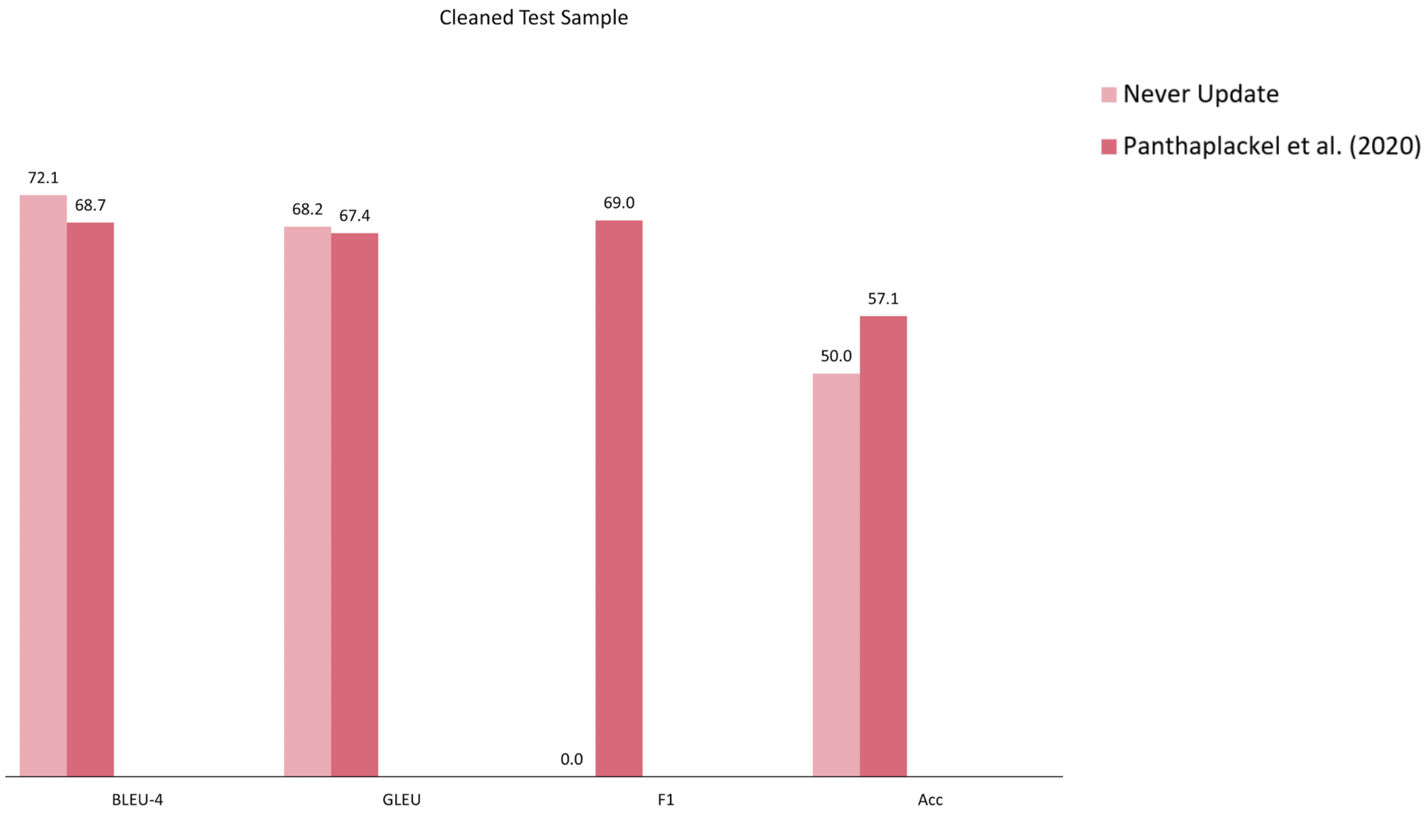
Pretrained Update and Detection	+ →	**Update** (Supervision: C_{new}) ↘	$P_{inconsistency\ label}$ = output(Detection)
			
			if output(Detection) is *Inconsistent*:
			$P_{comment}$ = output(Update)
			else:
			$P_{comment}$ = C

	+, - →	**Detection** (Supervision: $C \neq C_{new}$)	
Jointly Trained Update and Detection	+, - →	**Update** (Supervision: C_{new}) Shared: Embeddings, C encoder, M_{edit} encoder **Detection** (Supervision: $C \neq C_{new}$) ↗	

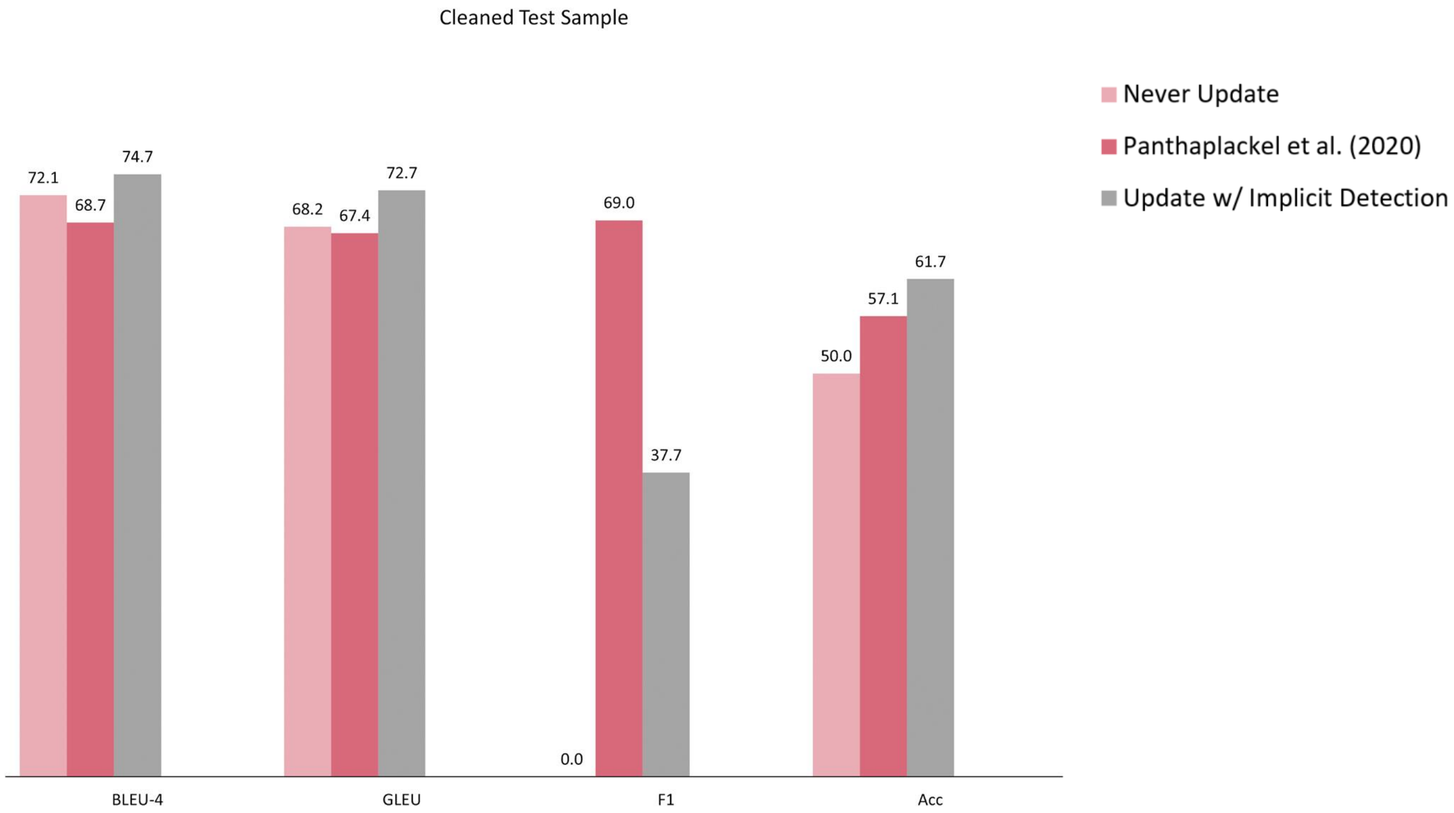
Extrinsic Evaluation: Results



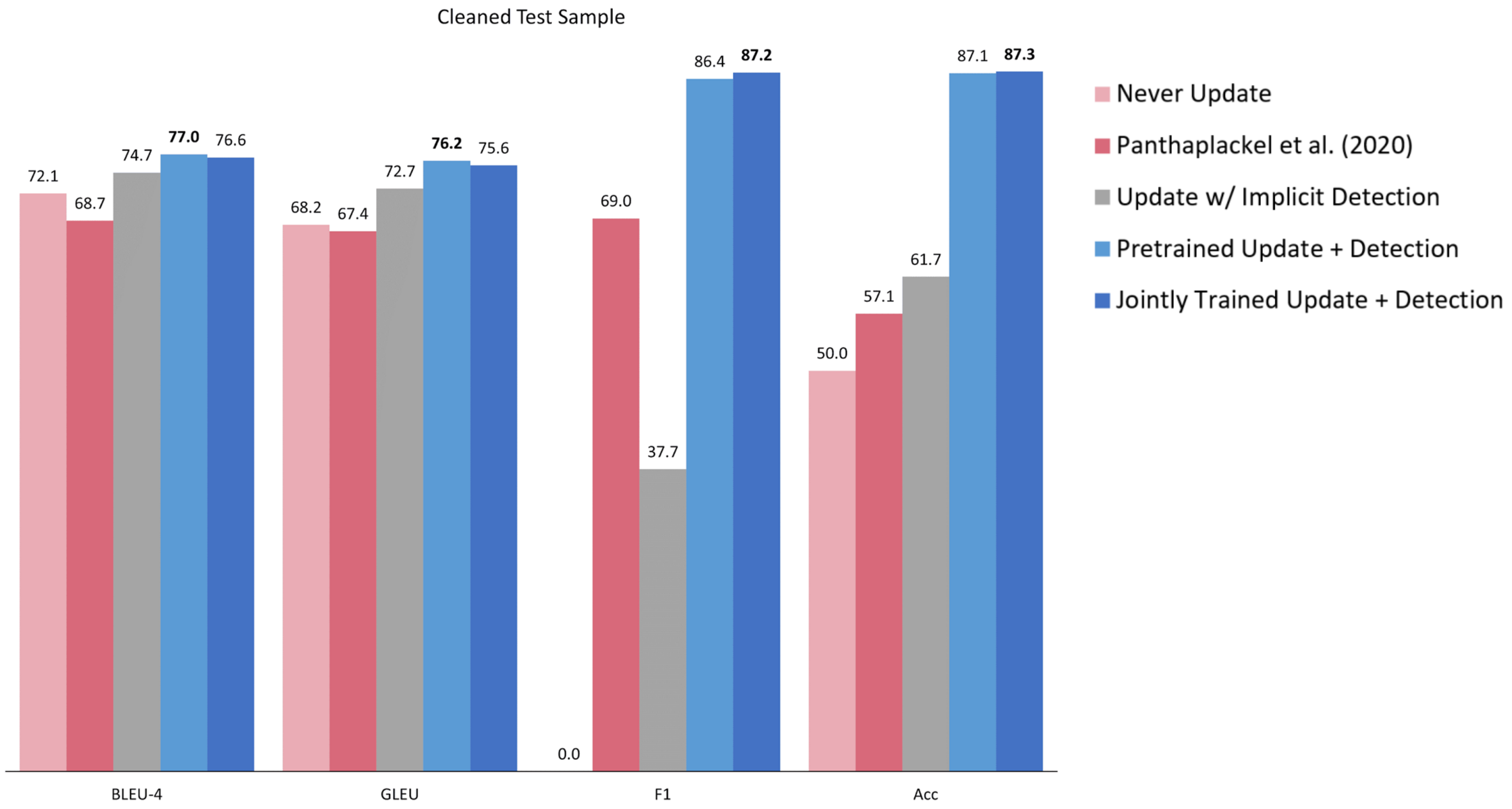
Extrinsic Evaluation: Results



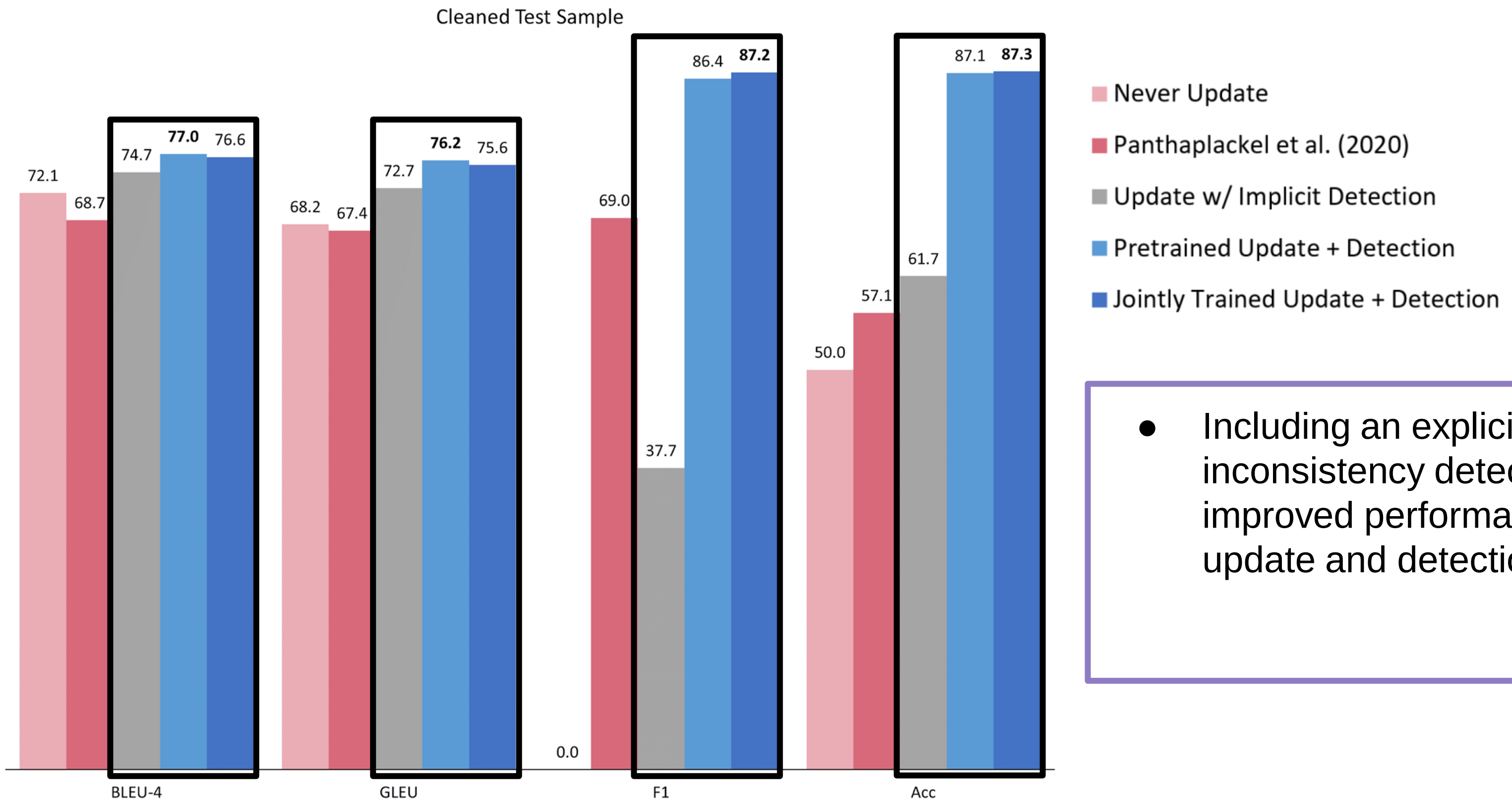
Extrinsic Evaluation: Results



Extrinsic Evaluation: Results



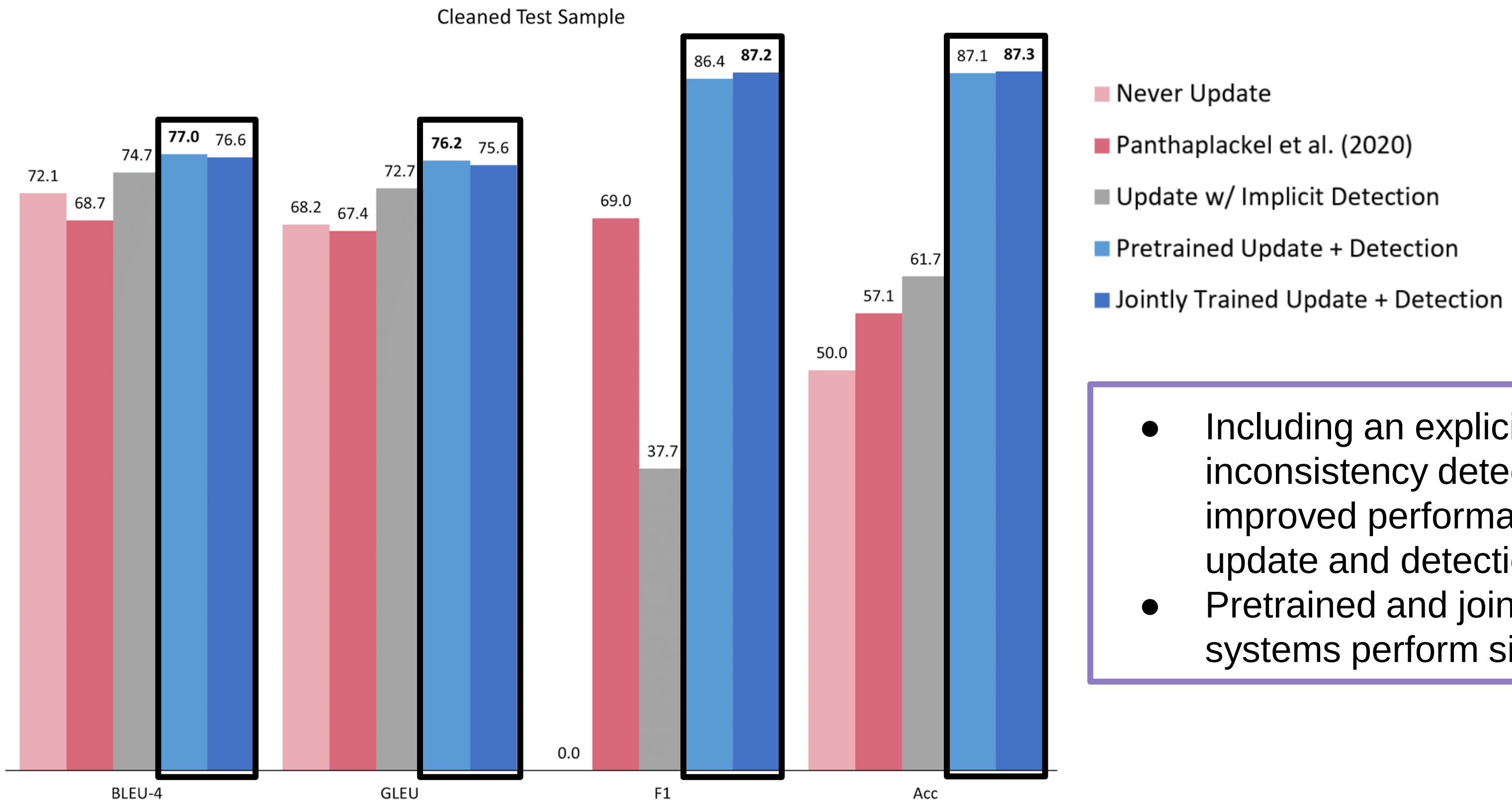
Extrinsic Evaluation: Results



- Including an explicit inconsistency detection leads to improved performance across update and detection metrics

The University of Texas at Austin

Extrinsic Evaluation: Results



- Including an explicit inconsistency detection leads to improved performance across update and detection metrics
- Pretrained and jointly trained systems perform similarly

Outline

- Task
- Architecture
- Data
- Intrinsic Evaluation
- Extrinsic Evaluation
- Summary

Summary

- We formulated a deep learning approach for just-in-time inconsistency detection between comments and code by learning to relate comments and code changes.
- We show that our approach can outperform multiple baselines as well as post hoc models.
- We also demonstrate that our approach can be used to build a comprehensive comment maintenance system which detects and updates inconsistent comments.

Code and data available: https://github.com/panthap2/deep-jit-inconsistency-detection

Contact: Sheena Panthaplackel <spantha@cs.utexas.edu>