	Update Metrics					Detection Metrics			
	xMatch	**METEOR**	**BLEU-4**	**SARI**	**GLEU**	**P**	**R**	**F1**	**Acc**
Never Update	50.0	67.4	72.1	24.9	68.2	0.0	0.0	0.0	50.0
? (**?**)	25.9	60.0	68.7	42.0*	67.4	54.0	**95.6**	69.0	57.1
Update w/ implicit detection	58.0	72.0	74.7	31.5	72.7	**100.0**	23.3	37.7	61.7
Pretrained update + detection									
SEQ(C, M_{edit}) + features	**62.3**†	75.6*	77.0*	42.0*	76.2	91.3*	82.0§	86.4*	87.1§¶
GRAPH(C, T_{edit}) + features	59.4	74.9§	76.6†	**42.5**$^{\|}$	75.8*†	85.8	87.1	86.4*	86.3†
HYBRID(C, M_{edit}, T_{edit}) + features	**62.3**†	75.8$^{\dagger\|}$	**77.2**	42.3†	**76.4**	92.3	82.4§	87.1†	87.8$^{*\|}$
Jointly trained update + detection									
SEQ(C, M_{edit}) + features	61.4*	**75.9**$^{\|}$	76.6†	42.4$^{\dagger\|}$	75.6†	88.3†	86.2	87.2†	87.3$^{\S\|}$
GRAPH(C, T_{edit}) + features	60.8	75.1§	76.6†	41.8*	75.8*	88.3†	84.7*	86.4*	86.7†¶
HYBRID(C, M_{edit}, T_{edit}) + features	61.6*	75.6*†	76.9*	42.3†	75.9*	90.9*	84.9*	**87.8**	**88.2***

Table 4: Results on joint inconsistency detection and update on the cleaned test sample. Scores for which the difference in performance is *not* statistically significant are shown with identical symbols.

`@return` and `@param` comments, conform to particular format. We instead capture deeper code/comment relationships by learning their syntactic and semantic structures. **?** propose a siamese network for correlating comment/code representations. In contrast, we aim to correlate comments and code through an attention mechanism.

Just-In-Time Inconsistency Detection: **?** (**?**) detect inconsistencies in a block/line comment upon changes to the corresponding code snippet using a random forest classifier with hand-engineered features. Our approach does not require such extensive feature engineering. Although their task is slightly different, we consider their approach as a baseline. **?** concurrently present a preliminary study of an approach which maps a comment to the AST nodes of the method signature (before the code change) using BOW-based similarity metrics. This mapping is used to determine whether the code changes have triggered a comment inconsistency. Our model instead leverages the full method context and also learns to map the comment directly to the code changes. **?** predict whether a comment will be updated using a random forest classifier utilizing surface features that capture aspects of the method that is changed, the change itself, and ownership. They do not consider the existing comment since their focus is not inconsistency detection; instead, they aim to understand the rationale behind comment updating practices by analyzing useful features. **?** develops at approach which locates inconsistent identifiers upon code changes through lexical matching rules. While we find such a rule-based approach (represented by our OVERLAP(C, deleted) baseline) to be effective, a learned model performs significantly better. **?** builds a system to mitigate the damage of inconsistent comments by prompting developers to validate a comment upon code changes. Comments that are not validated are identified, indicating that they may be out of date and unreliable. **?** present a framework for maintaining consistency between code and todo comments by performing actions described in such comments when code changes trigger the specified conditions to be satisfied.

9 Conclusion

We developed a deep learning approach for just-in-time inconsistency detection between code and comments by learning to relate comments and code changes. Based on evaluation on a large corpus consisting of multiple types of comments, we showed that our model substantially outperforms various baselines as well as post hoc models that do not consider code changes. We further conducted an extrinsic evaluation in which we demonstrated that our approach can be used to build a comprehensive comment maintenance system that can detect and update inconsistent comments.

Acknowledgments

This work was supported by the Bloomberg Data Science Fellowship and a Google Faculty Research Award.

Ethics Statement

Through this work, we aim to reduce time-consuming confusion and vulnerability to software bugs by keeping developers informed with up-to-date-documentation, in order to consequently help improve developers productivity and software quality. Buggy software and incorrect API usage can result in significant malfunctions in many everyday operations. Maintaining comment/code consistency can help prevent such negative-impact events.

However, over-reliance on such a system could result in developers giving up identifying and resolving inconsistent comments themselves. By presuming that the system detects all inconsistencies and all of these are properly addressed, developers may also take the available comments for granted, without carefully analyzing their validity. Because the system may not catch all types of inconsistencies, this could potentially exacerbate rather than resolve the problem of inconsistent comments. Our system is not intended to serve as an infallible safety net for poor software engineering practices but rather as a tool that complements good ones, working alongside developers to help deliver reliable, well-documented software in a timely manner.

Possimus reiciendis illo ex quibusdam consequatur pariatur perspiciatis, nobis voluptatem impedit officia aspernatur, ipsam soluta pariatur voluptate inventore vitae ducimus.Fuga minus pariatur

nulla asperiores maxime eos earum quos placeat eligendi, atque est iure facilis dolorem sunt consectetur earum?Et alias voluptatem placeat, adipisci nulla dignissimos inventore tenetur