《Detecting Missing Information in Bug Descriptions》	2017	ESEC/FSE
《Predicting Node Failure in Cloud Service Systems》	2018	ESEC/FSE
《Deep Just-in-Time Defect Prediction: How Far Are We? 》	2021	ISSTA

Detecting Missing Information in Bug Descriptions

2017 ESEC/FSE

Pattern code: P_SR_LABELED_LIST
Description: paragraph containing a non-empty labeled list of sentences that indicate actions. The label is optional and indicates S2R terms. The "action sentences" may be simple or continuous present/past sentences or imperative sentences. The list may contain OB and EB sentences in no particular order.
Rule: ([S2R label])

Pattern code: S_OB_NEG_AUX_VERB
Description: negative sentence with auxiliary verbs
Rule: ([subject]) [negative aux. verb] [verb] [complement]
Definitions:
[negative aux. verb] ∈ {are not, can not, does not, did not, *etc.* }
Example: [*The icon*] **[*did not*]** [*change*] [*to an hourglass...*] (from Eclipse 150)

Figure 1: Most common OB discourse pattern.

Observed

Reproduce

[2.] [*C-t to open a new tab.*] [*The second tab is now displayed.*]
[3.] [*Type 'hello'.*] [*This text appears in the location bar.*]
[4.] [*Click on the header for the first tab to switch to that tab.*]
[5.] [*Click on the header for the second tab...*]

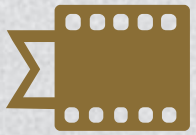
Figure 3: Most common S2R discourse pattern.

Detecting Missing Information in Bug Descriptions

2017 ESEC/FSE

Regular Expressions-based DeMIBuD

The regular expressions rely on frequently used words found in our EB and S2R discourse patterns



Heuristics-based DeMIBuD

DeMIBuD-H uses part-of-speech (POS) tagging and heuristics to match sentences and paragraphs to discourse patterns.

Machine Learning-based DeMIBuD

DeMIBuD-ML is based on state-of-the-art approaches in automated discourse analysis and text classification

Predicting Node Failure in Cloud Service Systems

2018 ESEC/FSE

Cloud Service Systems

Cloud computing has emerged as a new paradigm for delivery of computing as services via the Internet. A typical cloud service system contains a large number of physical servers, or "nodes".

Challenge

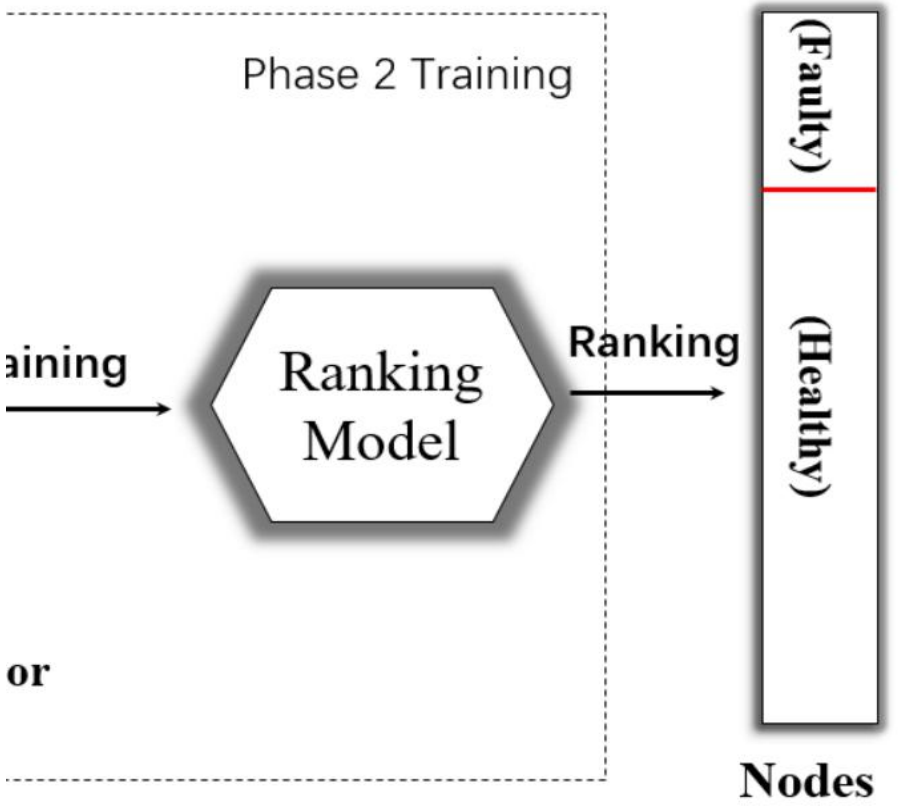
- Complicated failure causes
- Complex failure-indicating signals
- Highly imbalanced data

Predicting Node Failure in Cloud Service Systems

2018 ESEC/FSE

Table 1: Some examples of features

Feature	Type	Description
UpdateDomain	Spatial	The domain where nodes share same update setting.
MemoryUsage	Temporal	Memory consumption.
DiskSectorError	Temporal	Sector errors in a disk drive.
ServiceError	Temporal	Error counts from a deployed service.
RackLocation	Spatial	The location of the rack the node belongs to.
LoadBalanceGroup	Spatial	The group where nodes' load are balanced.
IOResponse	Temporal	I/O Response time.
OSBuildGroup	Spatial	The group where nodes have the same OS build.



Temporal Data

Spatial Data

which indicate explicit/implicit dependency in global relationships among nodes

aggregated as temporal data from the original sources

Deep Just-in-Time Defect Prediction: How Far Are We?

2021 ISSTA

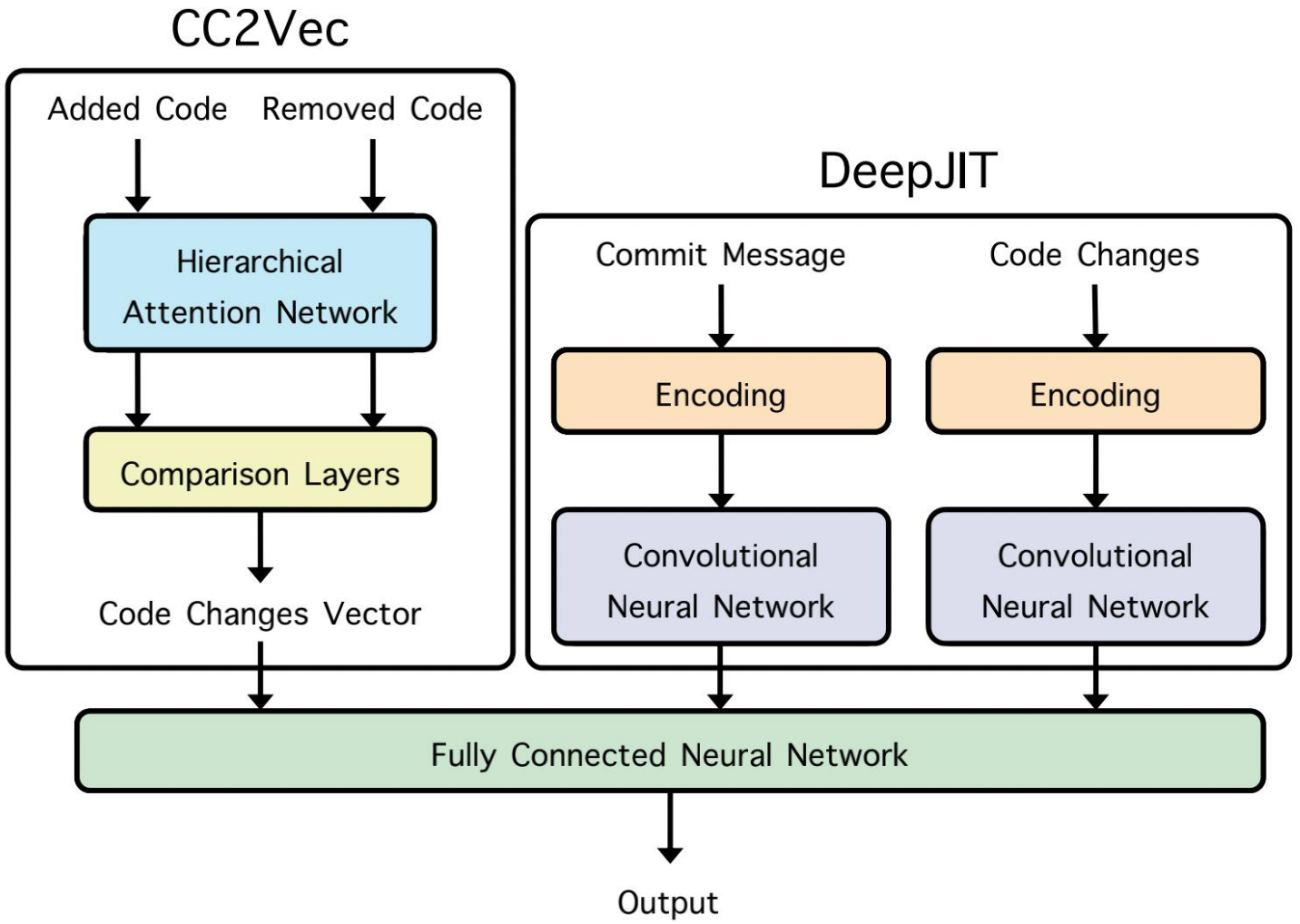


Figure 2: The overall framework of CC2Vec + DeepJIT

Deep Just-in-Time Defect Prediction: How Far Are We?

2021 ISSTA

2 gerrit-server/src/main/java/com/google/gerrit/server/git/VisibleRefFilter.java

```
103 103   if (!deferredTags.isEmpty() && (!result.isEmpty() || filterTagsSeperately)) {
104 104     TagMatcher tags = tagCache.get(projectName).matcher(
105 105         tagCache,
106 106         db,
107     -       filterTagsSeperately ? filter(db.getAllRefs()).values() : result.values());
    107 +       filterTagsSeperately ? filter(refs).values() : result.values());
108 108     for (Ref tag : deferredTags) {
109 109       if (tags.isReachable(tag)) {
110 110         result.put(tag.getName(), tag);
111 111       }
112 112     }
113 113   }
```

Fixing Commit: 6db280663f836096c30a9626e7170f4a36d8cc1f

2 gerrit-server/src/main/java/com/google/gerrit/server/git/VisibleRefFilter.java

```
112 112   if (!deferredTags.isEmpty() && (!result.isEmpty() || filterTagsSeperately)) {
113 113     TagMatcher tags = tagCache.get(projectName).matcher(
114 114         tagCache,
115 115         db,
116     -       filterTagsSeperately ? filter(refs).values() : result.values());
    116 +       filterTagsSeperately ? filter(db.getAllRefs()).values() : result.values());
117 117     for (Ref tag : deferredTags) {
118 118       if (tags.isReachable(tag)) {
119 119         result.put(tag.getName(), tag);
120 120       }
121 121     }
122 122   }
```

Figure 1: An illustrative example

Deep Just-in-Time Defect Prediction: How Far Are We?

2021 ISSTA

Finding : The DeepJIT commit-message/code-change vector representations contribute more for deep JIT defect prediction than the CC2Vec code-change vector representations.

Finding: In general, CC2Vec cannot clearly outperform Deep_x0002_JIT in the extended dataset, indicating that the vector representation of code changes extracted by CC2Vec do not contribute much in advancing JIT defect prediction.

Finding : DeepJIT and CC2Vec cannot always outperform traditional approaches on all the studied projects, and the finding holds for multiple widely-used metrics.

Finding : Our simplistic LApredict can substantially outperform the advanced deep-learning-based approaches in JIT defect prediction under both within-project and cross-project scenarios in terms of both effectiveness and efficiency.

THANK YOU FOR YOUR LISTENING.

谢谢您的聆听