TELL: Log Level Suggestions via Modeling Multi-level Code Block Information

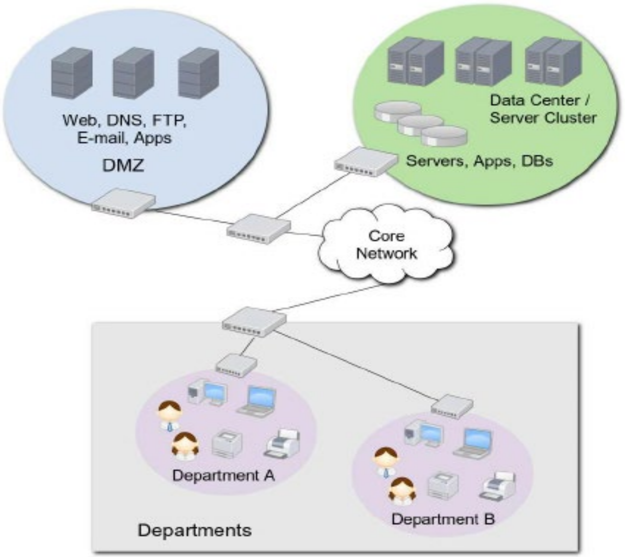
Jiahao Liu, Jun Zeng, Xiang Wang, Kaihang Ji, Zhenkai Liang

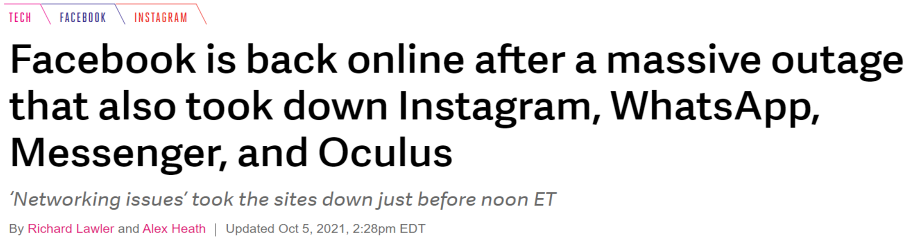
ISSTA, July 2022

NUS National University of Singapore

中国科学技术大学 1958 University of Science and Technology of China

Incidents in Complex Software Systems

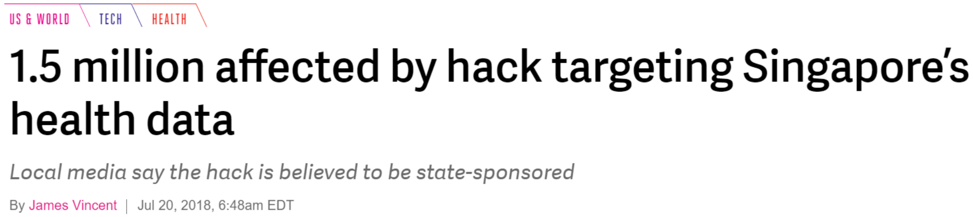



TECH FACEBOOK INSTAGRAM

Facebook is back online after a massive outage that also took down Instagram, WhatsApp, Messenger, and Oculus

'Networking issues' took the sites down just before noon ET

By Richard Lawler and Alex Heath | Updated Oct 5, 2021, 2:28pm EDT


US & WORLD TECH HEALTH

1.5 million affected by hack targeting Singapore's health data

Local media say the hack is believed to be state-sponsored

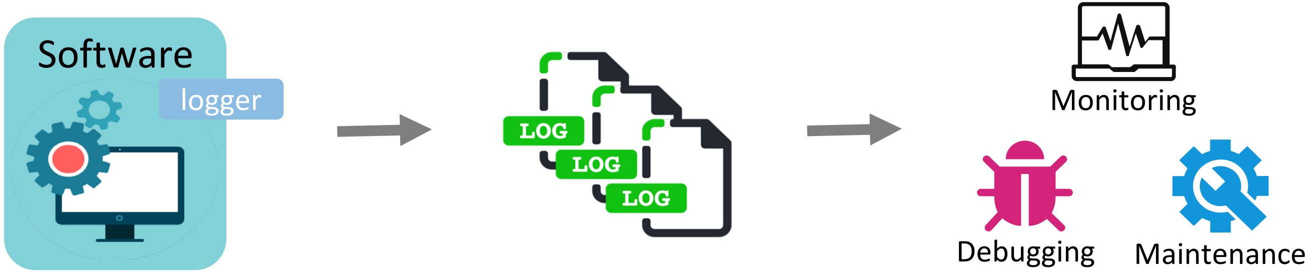
By James Vincent | Jul 20, 2018, 6:48am EDT

How to manage the reliability and security of software systems?

Observe System Status

Application Log is a common source that records software running information:

- Provide important indicators for software runtime status
- Monitor important information to recover software behaviors



Application logs facilitate system monitoring, debugging, and maintenance

How to Generate High-quality Logs?

```
LOG.warn("Couldn't find the leader with id = " + id)
```

- *Log level:* distinguish code contexts of different importance
- *Static text:* describe the purposes of logging statements
- *Dynamic variable:* record run-time execution information

```
2015-07-29 19:17:57,939 - INFO  [/10.10.34.11:3888] - Received connection request /10.10.34.13:58035
2015-07-29 19:17:57,955 - WARN  [Worker:188978561024] - Interrupting SendWorker
2015-07-29 19:18:04,430 - WARN  [Worker:188978561024] - Interrupted while waiting for message on queue
2015-07-29 19:18:07,748 - WARN  [Worker:188978561024] - Interrupting SendWorker
2015-07-29 19:18:07,756 - WARN  [Worker:188978561024] - Interrupting SendWorker
…
2015-07-29 19:58:05,166 - WARN  [Worker:188978561024] - Couldn’t find the leader with id = 67324
```

Suitable log levels are necessary to help application log analysis

Log Level Suggestions

Developers set levels as a filter to decide in which levels logs should be recorded

Trace Debug Info Warn Error

Log Level

Low levels for critical events will miss some important system signals

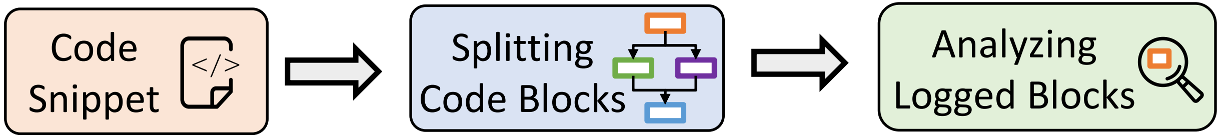
High levels for trivial events will generate lots of alarms

Research Question: Given a logging statement in source code, can we suggest a suitable log level for it?

Previous Work for Log Level Suggestions

Investigate Information of Logged Blocks:

- Which log level should developers choose for a new logging statement? [ESE'17]
- DeepLV: Suggesting log levels using ordinal based neural networks [ICSE'21]



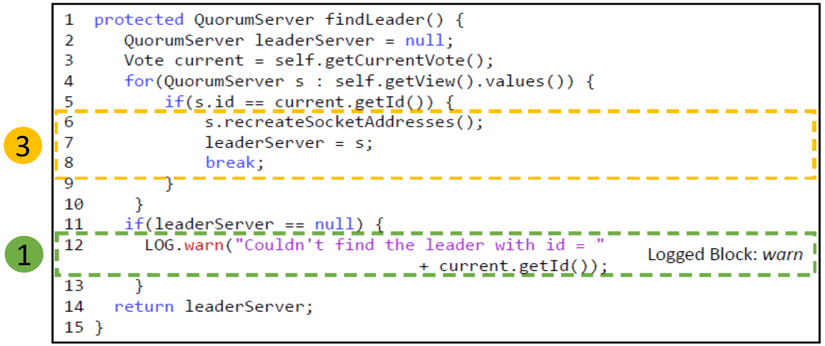
Existing work focuses on features inside one code block (i.e., intra-block information), ignoring features from its surrounding code blocks (i.e., inter-block information)

What is the key information that benefits log level suggestions?

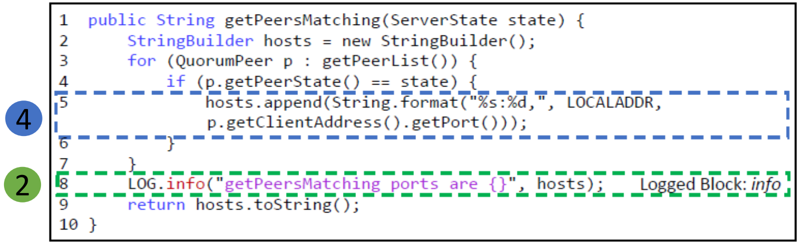
Our Insight

Key Insight: Logging statements describe program behaviors by recording both intra-block and inter-block information.

- Aggregating multi-level (i.e., intra- & inter-block) information gives better suggestions



```
protected QuorumServer findLeader() {
    QuorumServer leaderServer = null;
    Vote current = self.getCurrentVote();
    for(QuorumServer s : self.getView().values()) {
        if(s.id == current.getId()) {
            s.recreateSocketAddresses();
            leaderServer = s;
            break;
        }
    }
    if(leaderServer == null) {
      LOG.warn("Couldn't find the leader with id = "
                          + current.getId());
    }
  return leaderServer;
}
```



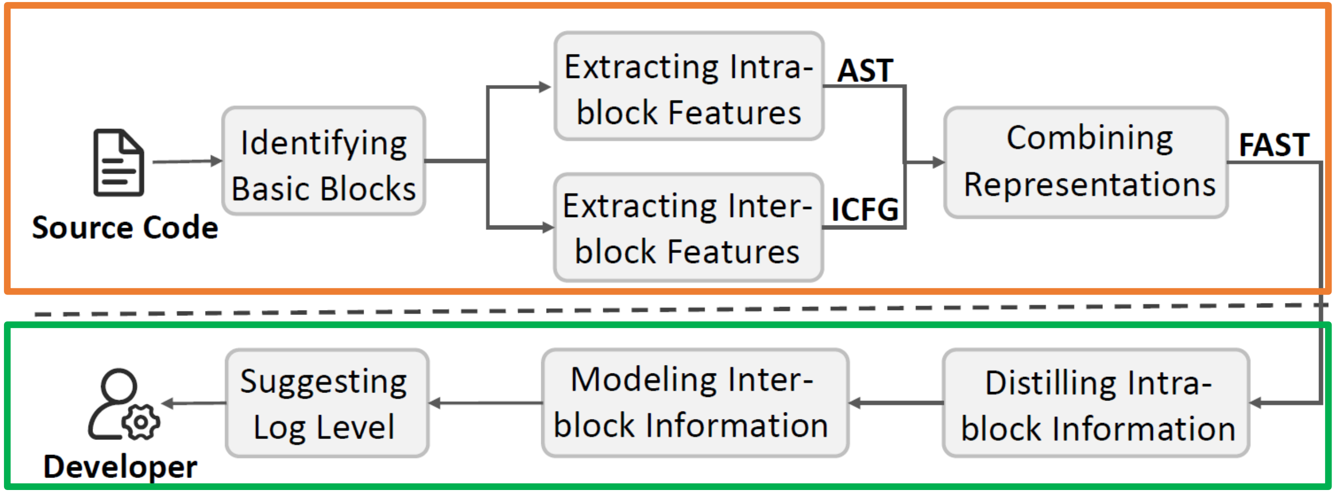
```
public String getPeersMatching(ServerState state) {
    StringBuilder hosts = new StringBuilder();
    for (QuorumPeer p : getPeerList()) {
        if (p.getPeerState() == state) {
            hosts.append(String.format("%s:%d,", LOCALADDR,
            p.getClientAddress().getPort()));
        }
    }
    LOG.info("getPeersMatching ports are {}", hosts);
    return hosts.toString();
}
```

1 2 *Intra-block* 3 *inter-block of block 1*

4 *inter-block of block 2*

1. Encode multi-level code block information into one representation?
2. Distill meaningful information for log level suggestions?

TELL (TeLL Log Levels)


Source Code
Identifying Basic Blocks
Extracting Intra-block Features
Extracting Inter-block Features
AST
ICFG
Combining Representations
FAST
Distilling Intra-block Information
Modeling Inter-block Information
Suggesting Log Level
Developer

Flow of AST (FAST) Construction: Encode multi-level block information

Log Level Suggestions: Extract meaningful information to suggest levels

How to Represent Multi-level Information?

Design choices:

- Log level suggestion levels: basic block level or function level?

Observations:

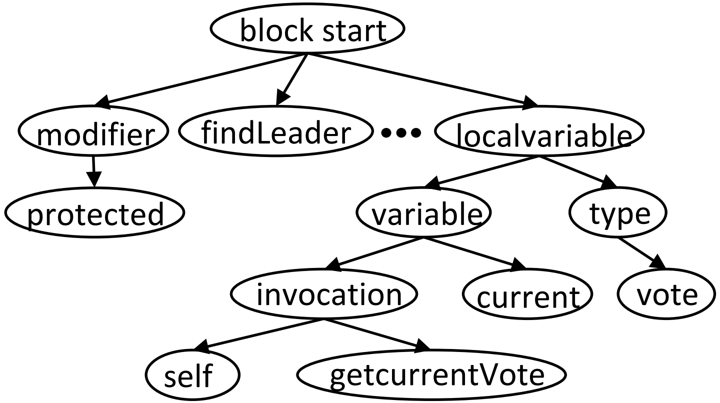
- A functions reflects complicated program behaviors, resulting in the constituent **logging statements at different levels**
- Most logged basic blocks include **one logging statement**
- The logic of basic block is **simple** & **easy** to represent

TELL chooses to analyze basic blocks to suggest log levels:

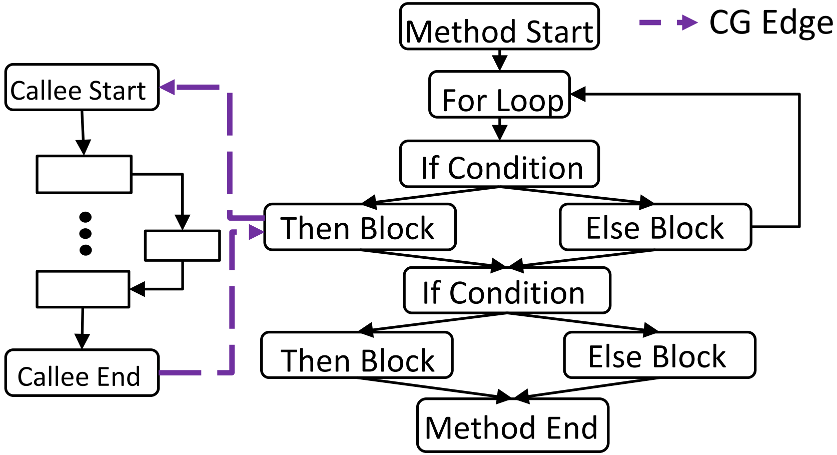
- Use basic blocks to model intra-block information
- Analyze block dependencies to extract inter-block information

Extracting Multi-level Information

- Abstract Syntax Tree (AST) of basic block preserves semantics and syntax
- Control Flow Graph (CFG) and Call Graph (CG) of a program can provide comprehensive semantics in the context of code blocks



Abstract Syntax Tree (AST)

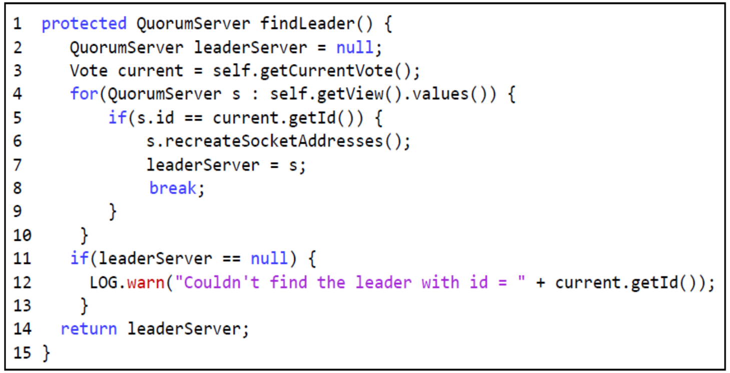


Inter-procedural Control Flow Graph (ICFG)

Combining Code Representations

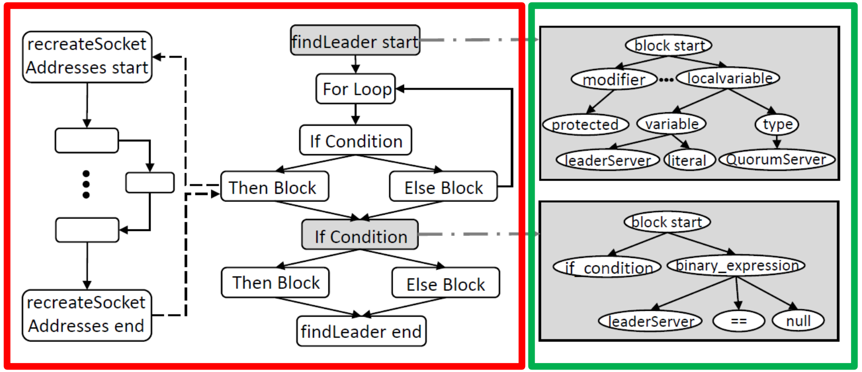
Integrate multi-level block information as Flow of AST (FAST) representation

- *Intra-block information* – Syntactical and semantic features in the AST
- *Inter-block information* – Contextual features in the ICFG



```
protected QuorumServer findLeader() {
    QuorumServer leaderServer = null;
    Vote current = self.getCurrentVote();
    for(QuorumServer s : self.getView().values()) {
        if(s.id == current.getId()) {
            s.recreateSocketAddresses();
            leaderServer = s;
             break;
        }
    }
    if(leaderServer == null) {
      LOG.warn("Couldn't find the leader with id = " + current.getId());
    }
   return leaderServer;
}
```

Code fragment



Inter-block Information (ICFG)

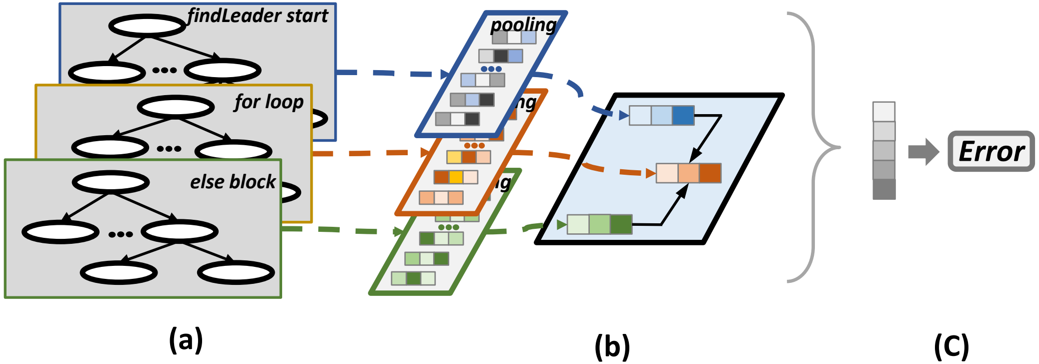
Intra-bock Information (AST)

FAST

Hierarchical Block Graph Network (HBGN)

Key Idea: design a hierarchical block graph network (HBGN) to propagate and aggregate information over the FAST to learn multi-level block features

- Graph neural networks (GNNs) excel at modeling graph-structured data



a) Distill intra-block information to parameterize each code block as a vector

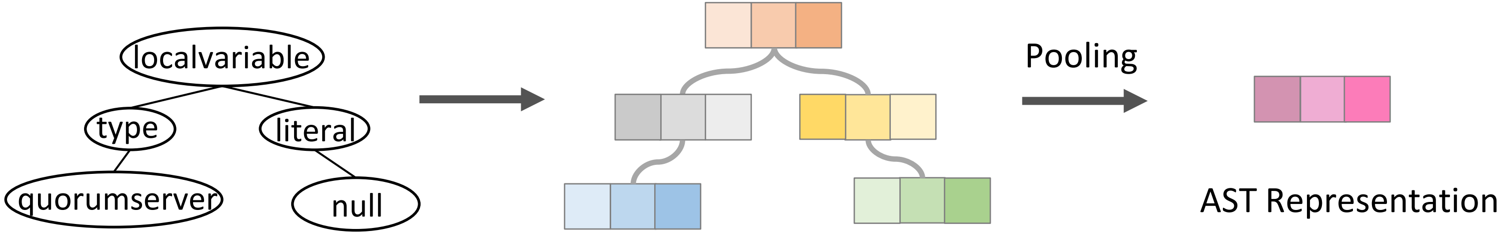
b) Model inter-block information by propagating information from neighbors

c) Suggest log levels with high-quality representations

Distilling Intra-block Information

Distill Intra-block information in an AST

- Symbols and graph structures of AST encode semantics and syntax
- Utilize GNN’s propagation and aggregation mechanism to extract intra-block features
- Apply average pooling to obtain AST’s embedding representation

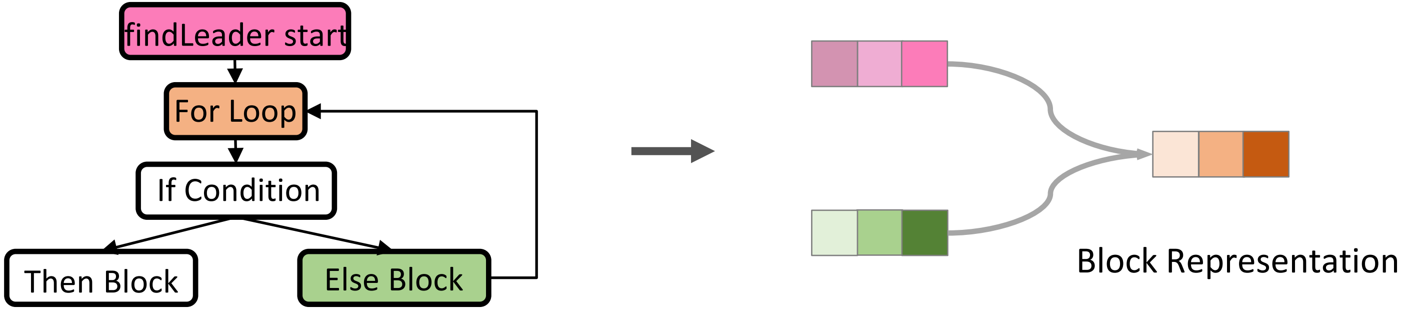


Extract intra-block information in AST with the first GNN

Modeling Inter-block Information

Model Inter-block information in an ICFG

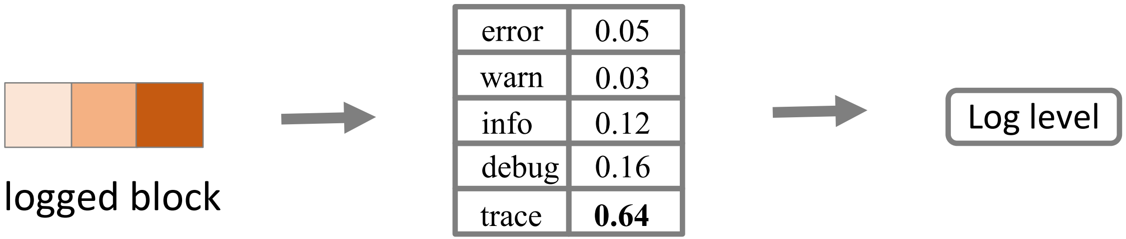
- Inter-block information complements AST’s intra-block information and provides additional clues to log level prediction
- Adopt the second GNN to aggregate neighbor information for ego code blocks and generate vector representations for logged blocks



`For Loop` obtains information from `findLeader start` and `Else Block`

Suggesting Log Levels

Given the vector representation of logged blocks with multi-level code block information, we convert log level suggestions into a classification problem.



error	0.05
warn	0.03
info	0.12
debug	0.16
trace	**0.64**

Suggesting levels with logged block representations

Evaluation

Experiment setup:

- Nine **large-scale open-source systems** from various domains

APACHE JMeter™ elastic APACHE WICKET APACHE HBASE kafka

Cassandra Flink Karaf APACHE ZooKeeper™

- Well-maintained logging statements in these systems as **ground truth**
- Training, Validation, and Testing sets – proportions: **60%, 20%, 20%**

Evaluation aspects:

- How **effective** is TELL on log level suggestions vs. state-of-the-art?
- To what extent do intra- and inter-block information **benefits** suggestions?
- How well does TELL suggest log levels **across systems**?

Effectiveness on Log Level Suggestions

Compare TELL with DeepLV in the **Accuracy**, **AUC** *(Area Under the Curve)* and **AOD** *(Average Ordinal Distance Score)* on nine large-scale open-source systems

Systems	**Accuracy**		**AUC**		**AOD**	
	DeepLV	**TeLL**	**DeepLV**	**TeLL**	**DeepLV**	**TeLL**
Cassandra	0.606	0.635	0.842	0.884	0.805	0.812
Elasticsearch	0.577	0.703	0.813	0.905	0.802	0.841
Flink	0.652	0.729	0.851	0.925	0.838	0.863
HBase	0.603	0.707	0.842	0.921	0.817	0.873
JMeter	0.623	0.737	0.839	0.921	0.809	0.872
Kafka	0.518	0.642	0.795	0.888	0.775	0.812
Karaf	0.672	0.750	0.856	0.908	0.816	0.867
Wicket	0.638	0.744	0.850	0.899	0.793	0.856
Zookeeper	0.609	0.746	0.848	0.924	0.820	0.887
Average	0.611	**0.710**	0.837	**0.908**	0.808	**0.854**

TELL achieves higher Accuracy, AUC and AOD than DeepLV in all systems

Study of Multi-level Block Information

Investigate the contribution of intra-block and inter-block information for log level suggestions

- Intra-block information → None/ASTNN/GNN
- Inter-block information → None/GNN

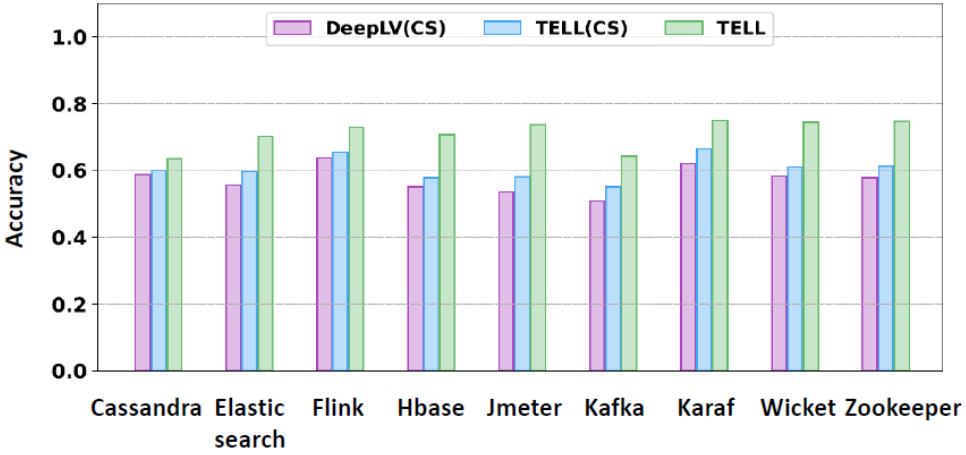
Model	Accuracy	AUC	AOD	Time(s)
GNN-NONE	0.517	0.840	0.733	3,459
NONE-GNN	0.602	0.875	0.796	1,780
ASTNN-GNN	0.618	0.878	0.802	8,908
GNN-GNN (HBGN)	**0.678**	**0.893**	**0.833**	**4,713**

Both Intra- and inter-block information contributes to log level suggestions

Cross System Suggestions

Evaluate TELL's effectiveness on cross-system suggestions

- Train HBGN using eight arbitrary systems of the nine systems
- Test HBGN on the remaining system



TELL has the potential to predict log levels for new systems

Conclusion

We propose TELL to facilitate log level suggestions:

- Encode intra-block and inter-block information into a joint representation – Flow of AST (FAST)
- Utilize Hierarchical Block Graph Network (HBGN) to extract information from FAST to suggest log levels
- Outperform the state-of-the-art in terms of accuracy
- Release source code at: https://github.com/ljiahao/TeLL

Key insights:

- Multi-level code block information benefits log level suggestions
- GNNs excel at extracting useful information from programs

Thank you!