Linking Source Code to Untangled Change Intents

Xiaoyu Liu[1] LiGuo Huang[1] Chuanyi Li[1,2] Vincent Ng[3]

[1] Dept. of Computer Science and Engineering, Southern Methodist University

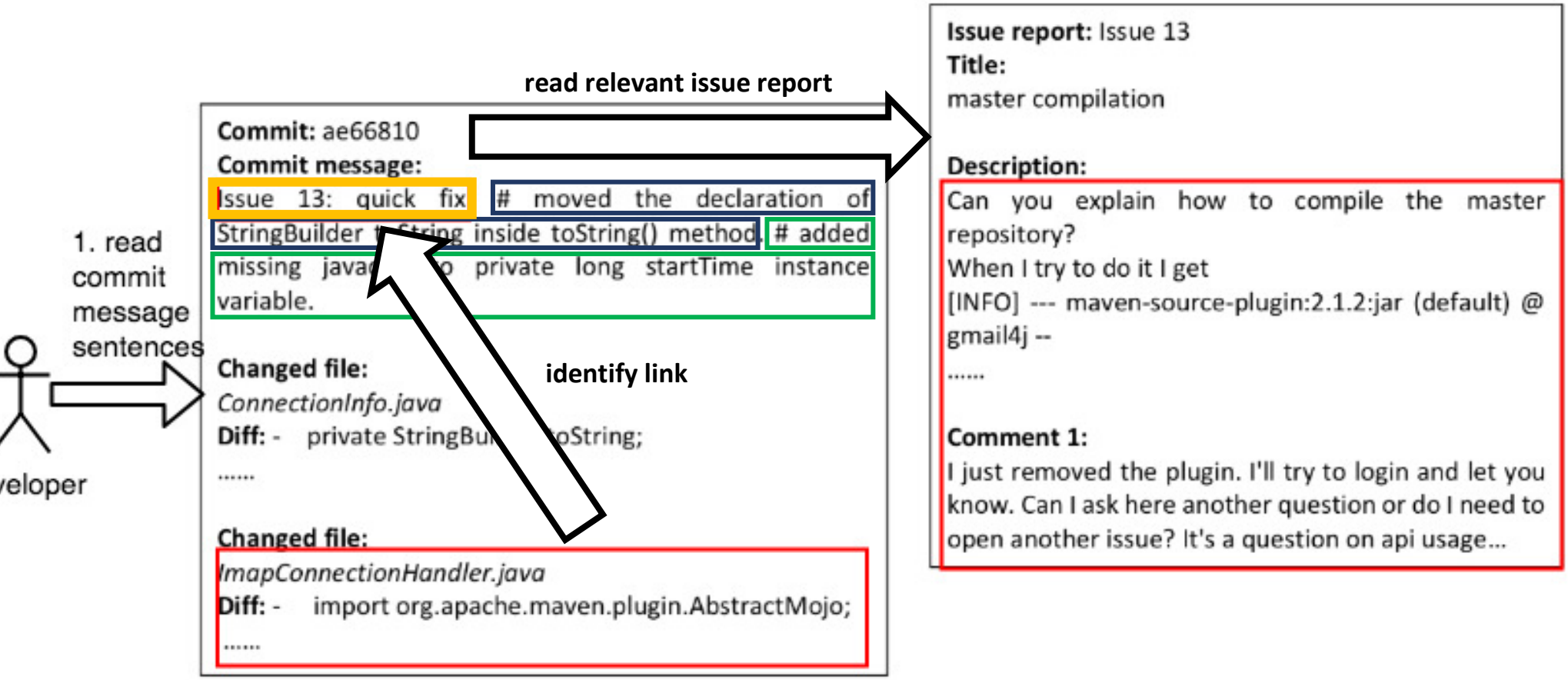
[2] State Key Laboratory for Novel Software Technology, Nanjing University

[3] Human Language Technology Research Institute, University of Texas at Dallas

34th IEEE International Conference on Software Maintenance and Evolution

Madrid, Spain
September 23-29, 2018

Motivating Example



Key Challenges

Change intents are tangled together in commit message

Manually link changed source code to change intents take lots of human efforts

Key Challenges

Change intents are tangled together in commit message

- **Solution**: Untangle change intents

Manually link changed source code to change intents take lots of human efforts

- **Solution**: An automated approach

xisting Software Artifacts Linking Approaches

Manual approaches

- **Problem:** Time-consuming, labor-intensive, and requires a great deal of experience

Information Retrieval (IR)-based approaches

- Use a IR-based model: measures similarity between changed source code and untangled change intent
- **Problem**: Changed entities extracted from source code could be very ***different*** from what is described in commit messages and other related software documents

Goal

Propose approaches to address the task of linking changed source code to untangled change intent **at the sentence level**

Our Approaches

AutoCILink-P

- Automatically identify links by using **manually defined patterns**

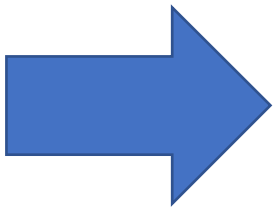
AutoCILink-ML

- Automatically identify links by applying **supervised learning**

attern-based Link Identification System (AutoCILink-P)

ntangling change intents

Issue 13: quick fix. # moved the declaration of StringBuilder inside toString() method. # added missing javadoc to provide long startTime instance variable



- *Issue 13: quick fix.*
- *# moved the declaration of StringBuilder inside toString() method.*
- *# added missing javadoc to provide long startTime instance variable*

attern-based Link Identification System (AutoCILink-P)

Enriched untangled change intents

Commit: ae66810
Commit message:
ssue 13: quick fix. # moved the declaration of StringBuilder
toString inside toString() method. # added missing javadoc
to private long startTime instance variable.

Issue report: Issue 13
Title:
master compilation

Description:
Can you explain how to compile the master reposit
When I try to do it I get
...

attern-based Link Identification System (AutoCILink-P)

Extracting terms from enriched change intents and changed entities om changed source code

"# moved the declaration of StringBuilder inside toString() method."

Terms: "declaration", "string", "builders", ...

Changed file:
SimulationRun.java
Diff: .....
+ eventHandlerMap.put(SimulationEvent.-TYPE_ACQUIRE_JOB_NOTIFICATION_EVENT, new AcquireJobNotificationEventHandler(job-Executor));
.....

Changed entities: "event", "handler", "map", "simulation", "type", "job",...

attern-based Link Identification System (AutoCILink-P)

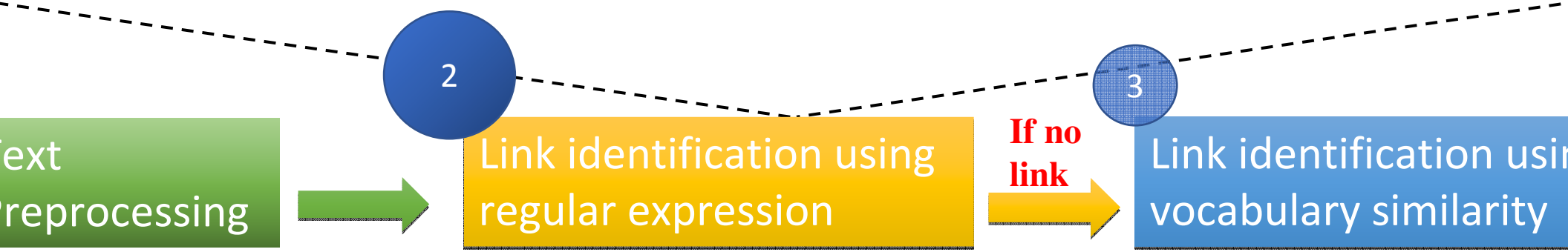
nerating regular expressions

```
^.*\s*(<verb>)(.*?)<entity>(.*)

^.*\s*<entity>(.*?)(<verb>)(.*)
```

<verb>: collected from change types (e.g., *move*)

<entity>: noun words in changed sou code identifiers, comments or string literals (e.g., *builder*)



attern-based Link Identification System (AutoCILink-P)

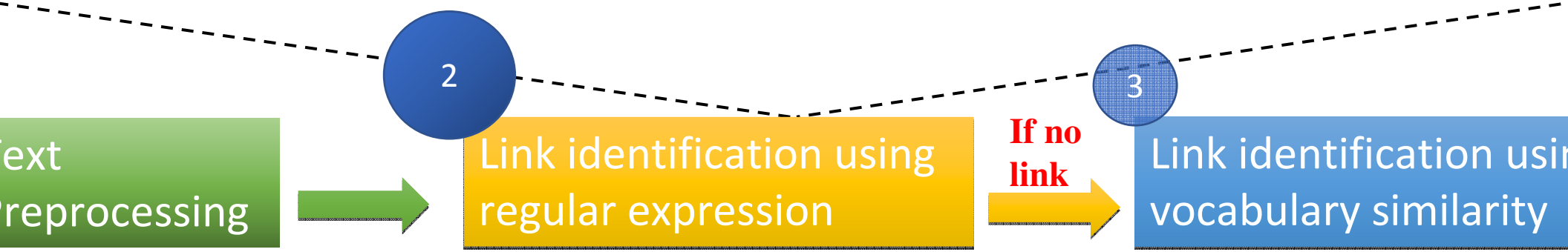
nerating regular expressions

```
^.*\s*(<verb>)(.*?)<entity>(.*)

^.*\s*<entity>(.*?)(<verb>)(.*)
```

^.*\s***move**.*?**builder**.*

"*moved the declaration of stringbuila inside toString() method*"



attern-based Link Identification System (AutoCILink-P)

ocabulary similarity between enriched untangled change intent terms d changed source code entities

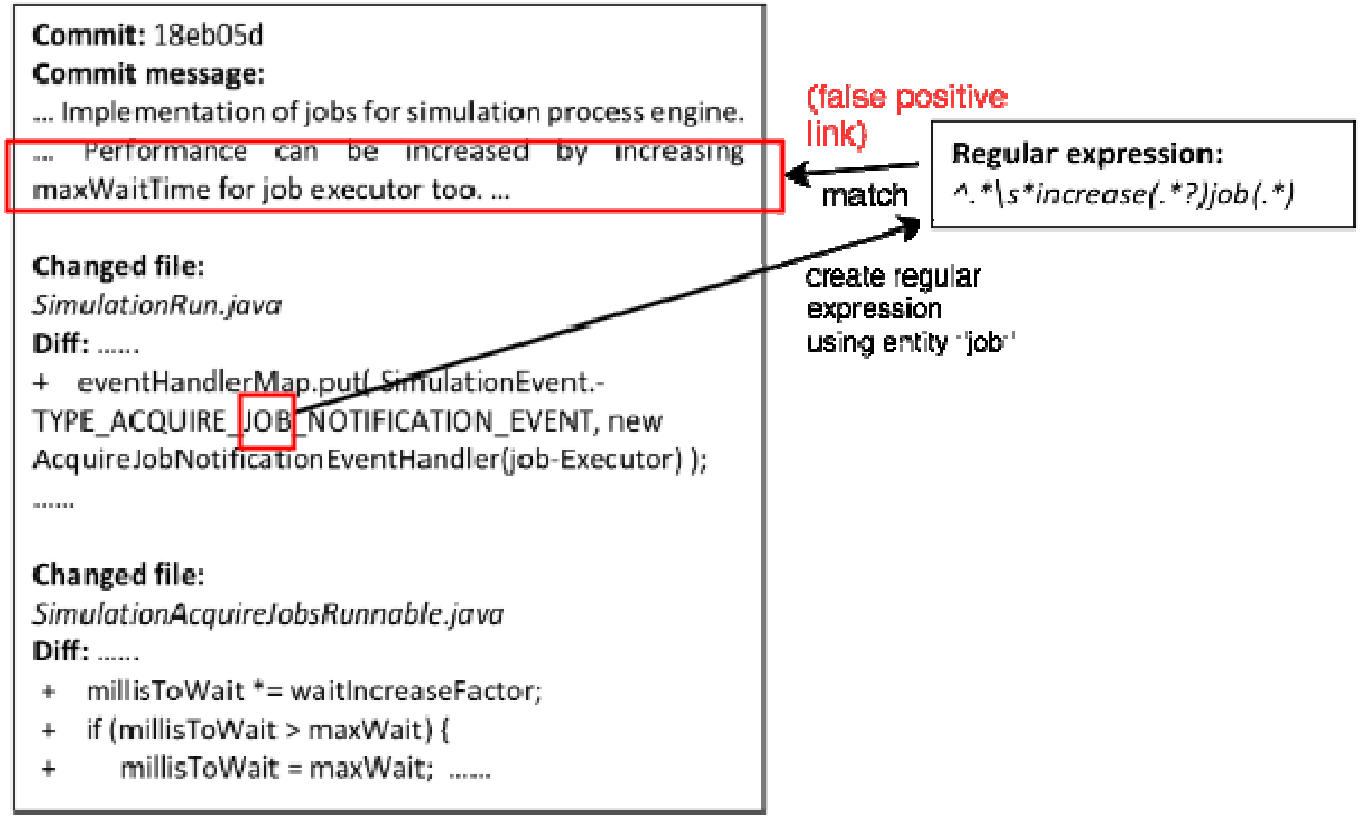
$$im(i,c) = \frac{\sum_{t \in V, e \in V} \omega(t,i,V) \times \omega(e,c,V)}{\sqrt{\sum_{t \in V} \omega(t,i,V)^2} \times \sqrt{\sum_{e \in V} \omega(e,c,V)^2}}$$

$$m(d,c) = \frac{\sum_{t \in V, e \in V} \omega(t,d,V) \times \omega(e,c,V)}{\sqrt{\sum_{t \in V} \omega(t,d,V)^2} \times \sqrt{\sum_{e \in V} \omega(e,c,V)^2}}$$

$$sim(Link_{(i,c)}) = max(sim(i,c), \bigcup_{x=1}^{n} sim(d_x, c$$

AutoCILink-P's Key Weakness

Regular expressions are not always precise

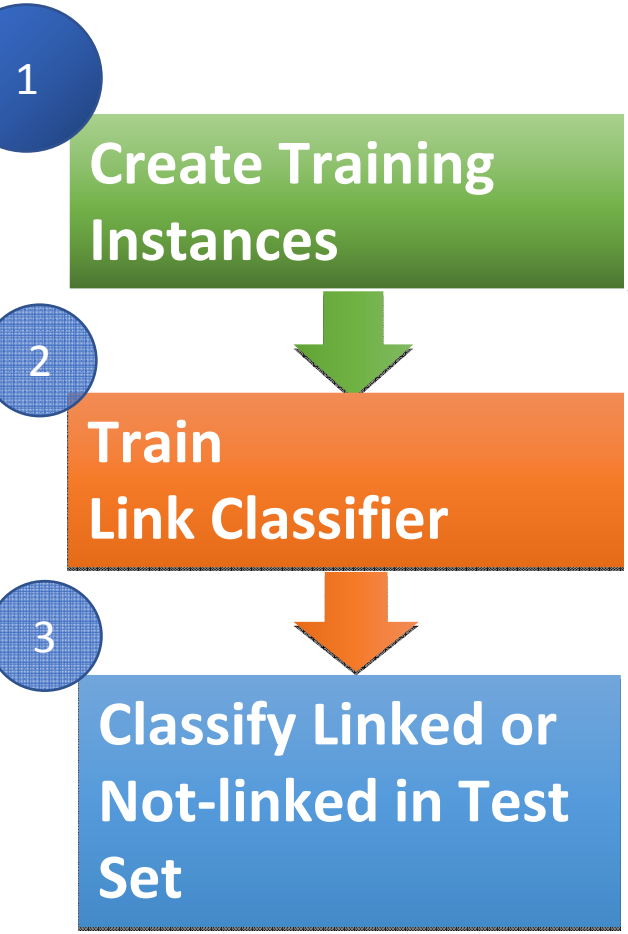


AutoCILink-P's Key Weakness

Impreciseness of regular expressions

- **Solution**: employs **a learning-based link classification system** to weigh the importance of matched regular expressions

upervised learning-based link classification system (AutoCILink-M



- Create training instances for each changed source code and enriched untangled change intent pair in the training dataset.
- Each instance annotated as ***linked*** or ***not link***depending on whether there is a link
- Each instance is represented using 6 features

Features

Type 1: Regular Expression Features encode the ***presence*** or ***absence*** of th
regular expression in the training set

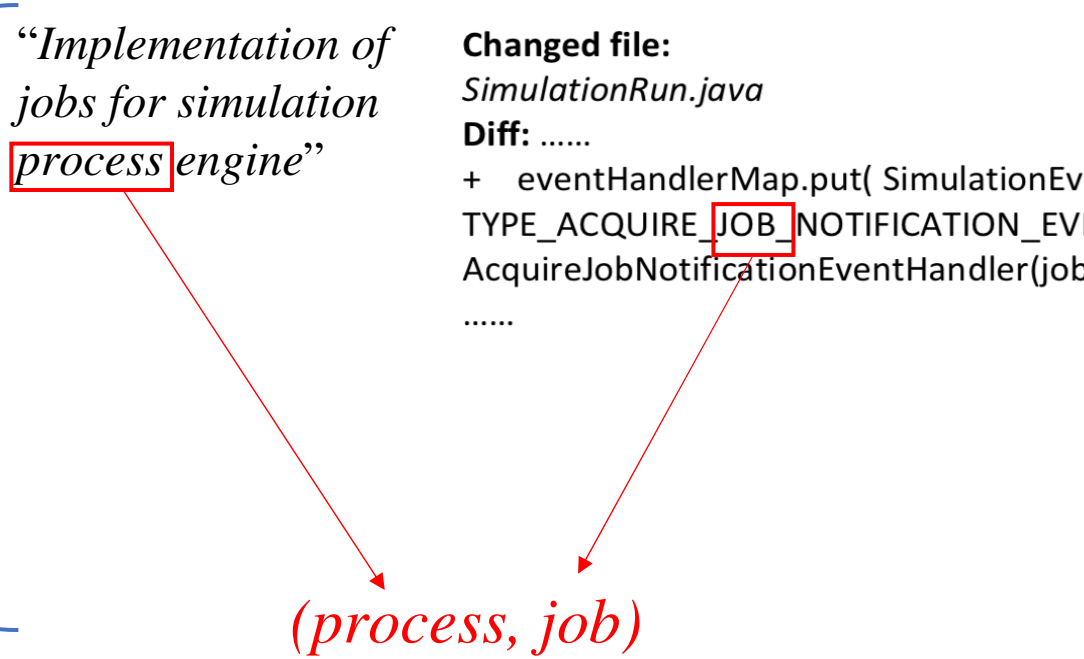
Type 2-4: Three types of **Vocabulary Features**

ree Types of Vocabulary Features

pe 2: Vocabulary Pair Features

pe 3: Vocabulary Similarity atures

pe 4: Term Unmatched Features



ree Types of Vocabulary Features

pe 2: Vocabulary Pair Features

pe 3: Vocabulary Similarity
tures

pe 4: Term Unmatched Features

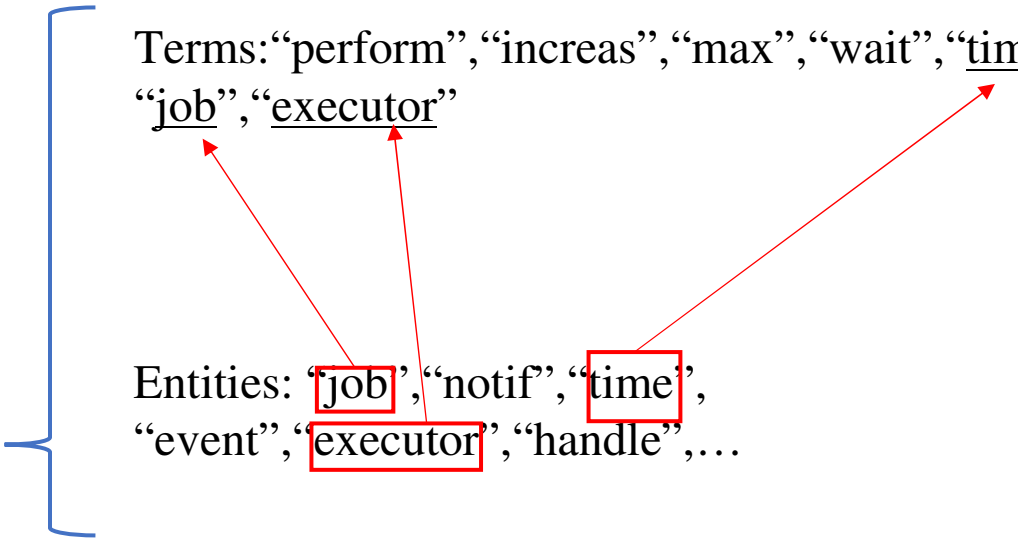
Encode the vocabulary simila
scores between an enriched
untangled change intent and
changed source code

ree Types of Vocabulary Features

pe 2: Vocabulary Pair Features

pe 3: Vocabulary Similarity atures

pe 4: Term Unmatched Features



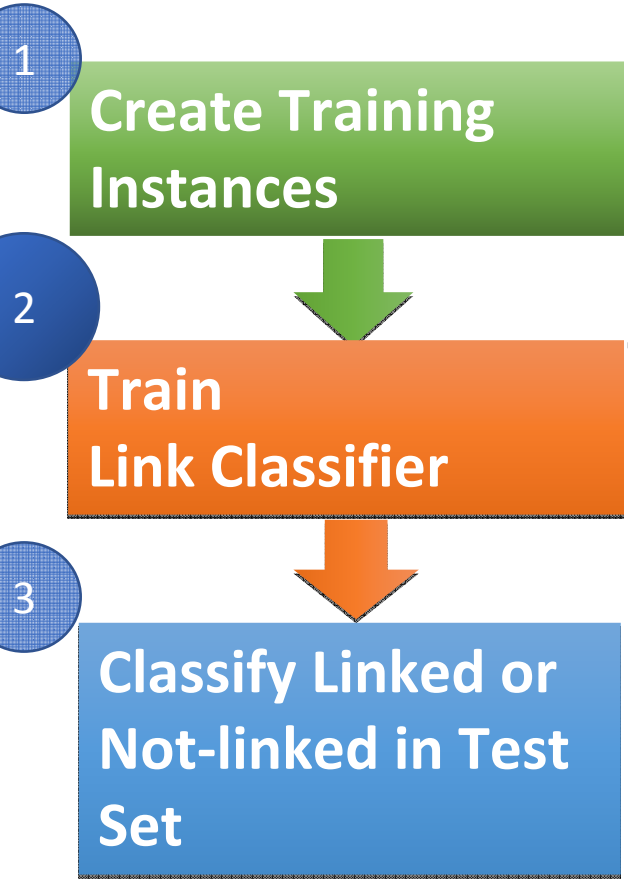
3 out of 7 (42.9%) matched, so the percentage of n terms unmatched in changed entities (1-42.9%=57 falls in the range of ***[50%,60%)***

Features

Type 5: Code Import Features encode the percentage of terms unmatched each imported code module changed entities

Type 6: Untangled Change Intent Count Features encode the number of untangled change intents in corresponding commit message

ervised learning-based link classification system (AutoCILink-ML

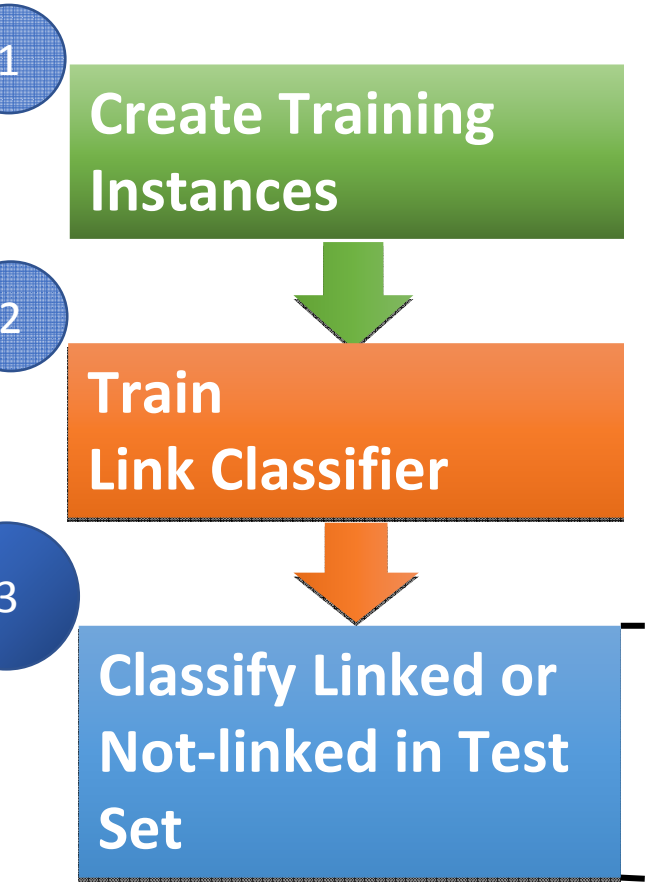


AutoCILink-ML uses the SVM learning algorithm to train a link classifier.

SMU

UT D

pervised learning-based link classification system (AutoCILink-M



AutoCILink-ML determines whether the given untangled change intent an changed source code file ar "*linked*" or "*not linked*".

MU

mpirical Evaluation

asets: Open Source Java Projects from GitHub

19 Projects from GitHub, 572 untangled change intents, 2739 changed source code fil
025 “linked” code-intent pairs (70.1%), and 1288 “not linked” code-intent pairs (29.9

Five-Fold Cross Validation

rics

Recall, Precision, F1-score

Accuracy: percentage of code-intent pairs correctly classified

eline Systems

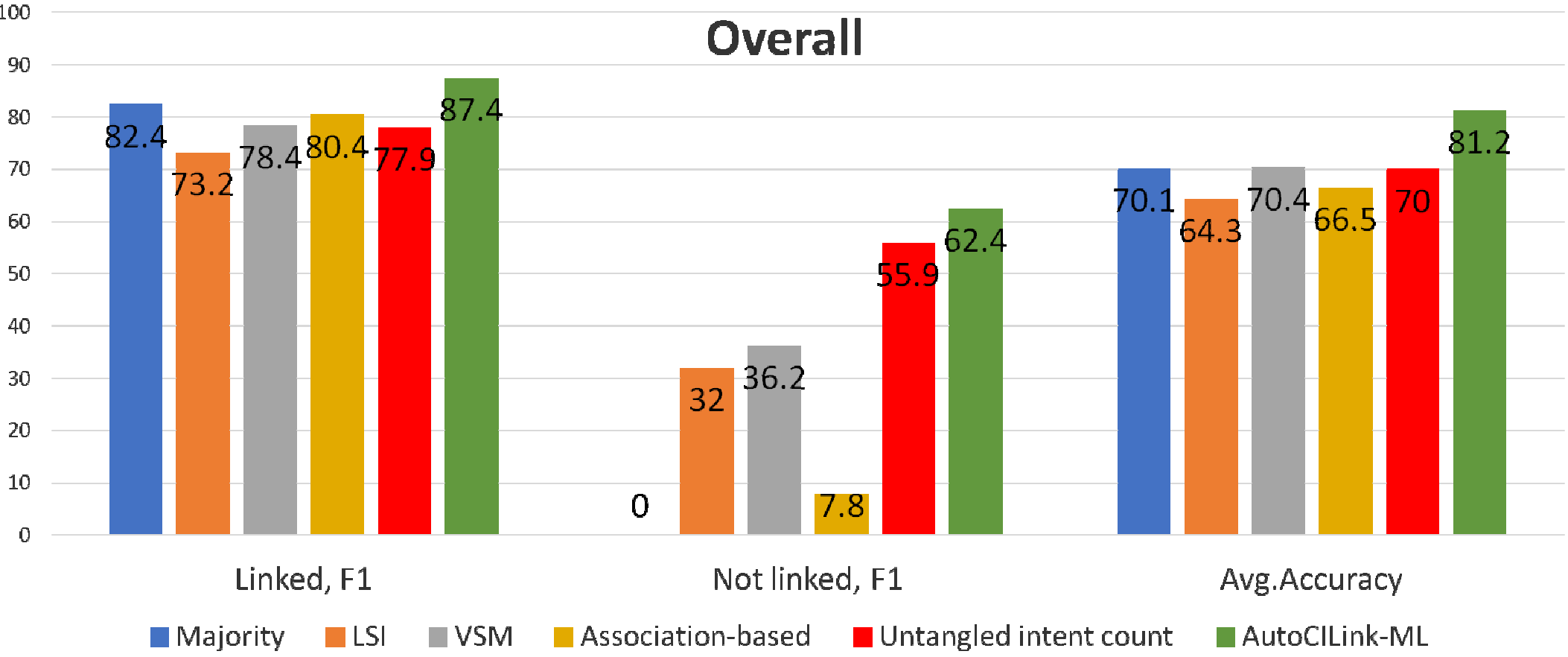
***IR-based Systems*:** LSI, VSM, Association-based

***Majority Classifier*:** a greedy approach simply classify into the majority class (i.e., “*link*

Untangled Intent Count Classifier: classify using threshold based on untangled ch
ntents count

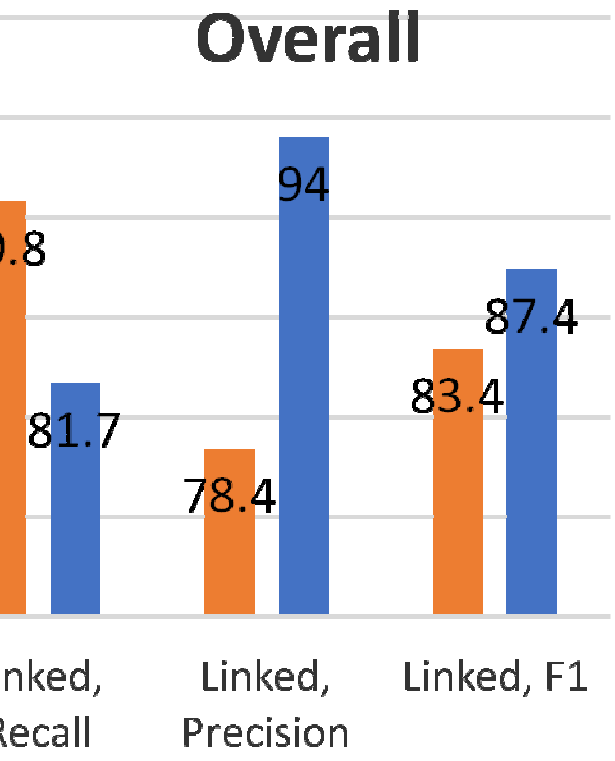
verall Performance

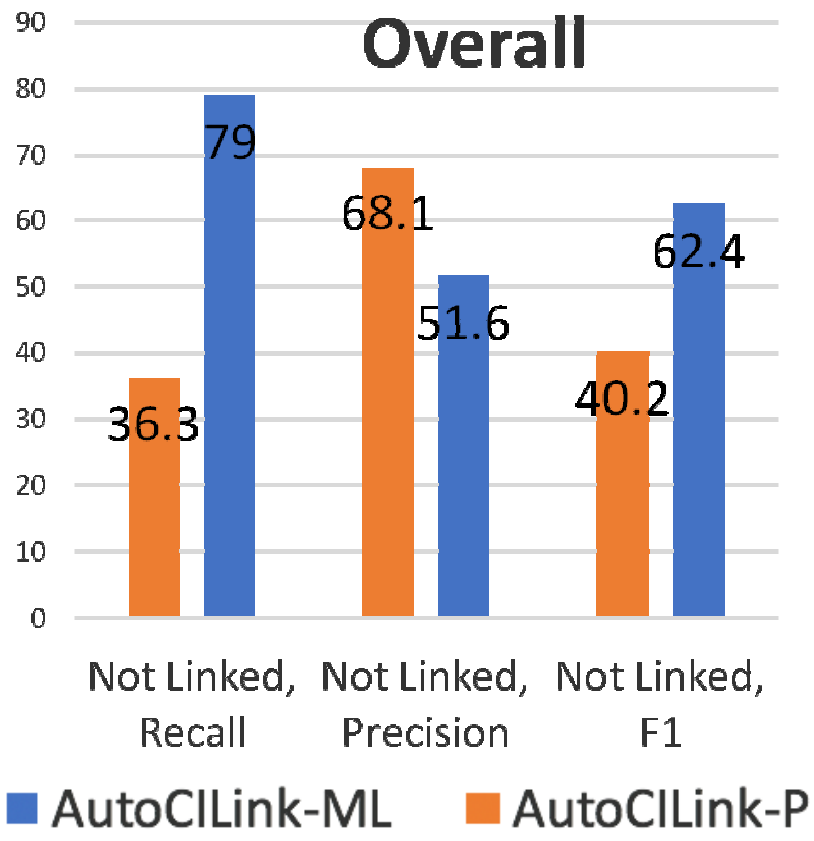
***1.** How effective is AutoCILink in linking changed code to untangled change intents*

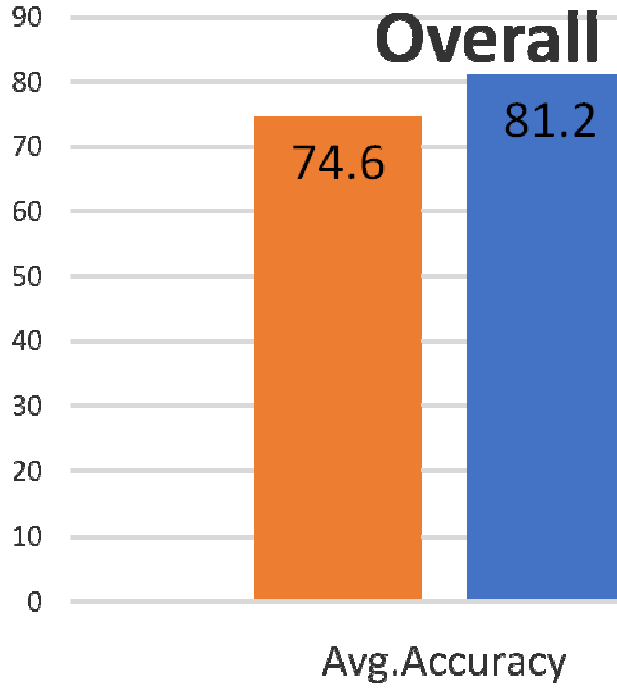


utoCILink-ML vs. AutoCILink-P

***2.** Which system is more accurate in linking changed code to untangled change ents, AutoCILink-ML or AutoCILink-P*







ature Ablation

. Which feature types have the largest impact on the performance of oCILink-ML?

The most important features are *Untangled Change Intent Count Features* and *Term Unmatched Features*, as their removal results in a 18.1% and 12.2% drop in accuracy, respectively.