Automatically pinpointing original logging functions from log messages for network troubleshooting

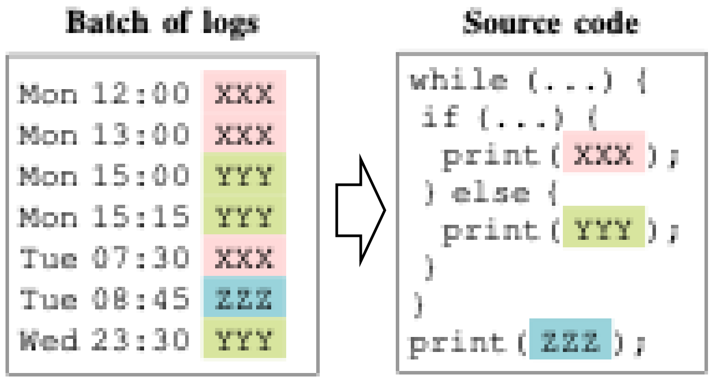
Gaspard Damoiseau-Malraux[1], Satoru Kobayashi[2], Kensuke Fukuda[3]

1: Sorbonne Université, 2: Okayama University, 3: NII/Sokendai

COMPSAC NCIW 2025, July 11, 2025

Introduction

- Network log messages
  - – Huge amount
  - – Often lack details to explain failures
  - ➢ Further analysis is needed
- Source code analysis to find log origins
  - – Identifying <u>original logging functions</u>
  - – Effective to understand system behaviors
    - e.g., What happens in the system when the log message appear?

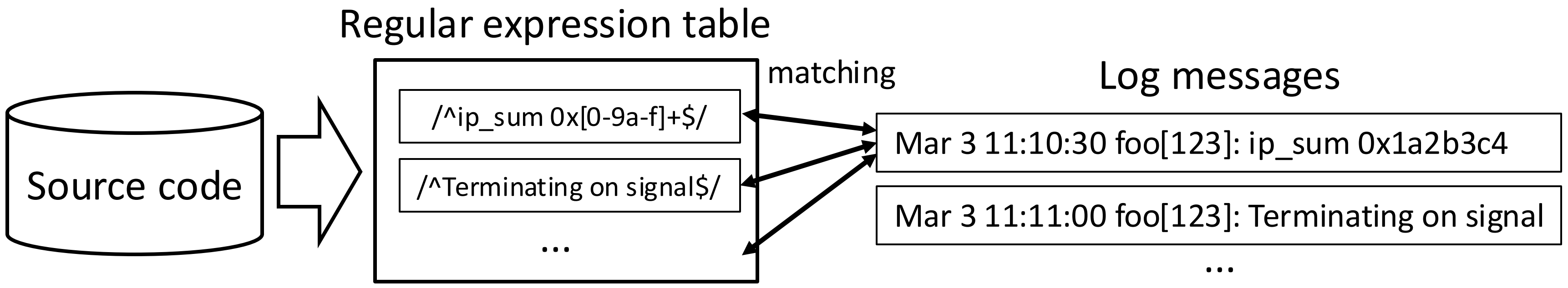


```
Mon 12:00  XXX
Mon 13:00  XXX
Mon 15:00  YYY
Mon 15:15  YYY
Tue 07:30  XXX
Tue 08:45  ZZZ
Wed 23:30  YYY
```

```
while (...) {
 if (...) {
  print( XXX );
 } else {
  print( YYY );
 }
}
print( ZZZ );
```

Existing approach and challenge

- Match input log messages with regular expression (regex) patterns extracted from logging functions in source code [3]
- Challenge: Large processing time due to matching all regex patterns for each log input

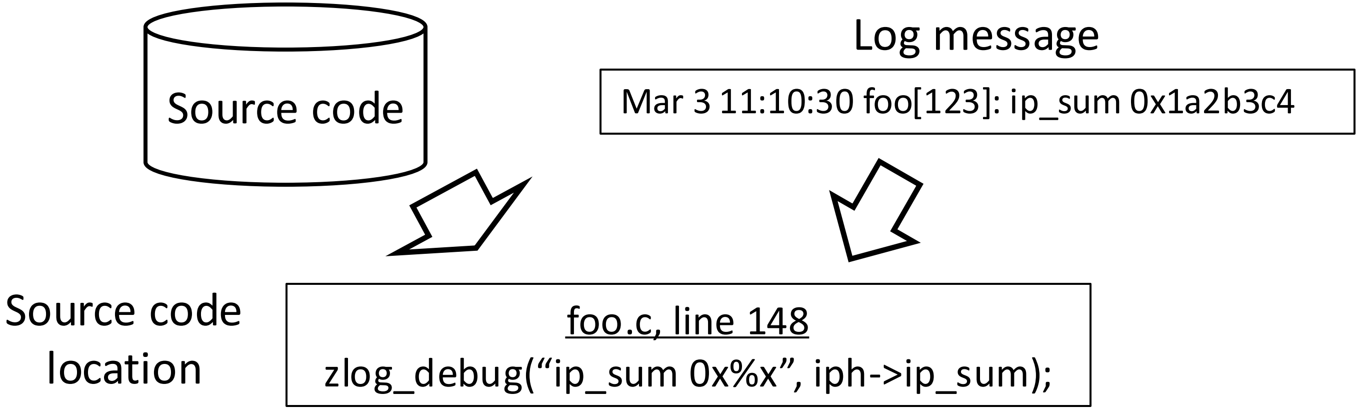


```
Mar 3 11:10:30 foo[123]: ip_sum 0x1a2b3c4
Mar 3 11:11:00 foo[123]: Terminating on signal
```

[3] W. Xu, et al. "Detecting large-scale system problems by mining console logs," Proceedings of SOSP '09, pp.117–132, 2009.

Research goal

- Design and develop a system to automatically identify original logging functions from source code
  - Fast log matching with new hybrid approach
  - Implement SCOLM (Source Code Origins of Log Messages) for C programs



```
Mar 3 11:10:30 foo[123]: ip_sum 0x1a2b3c4
```

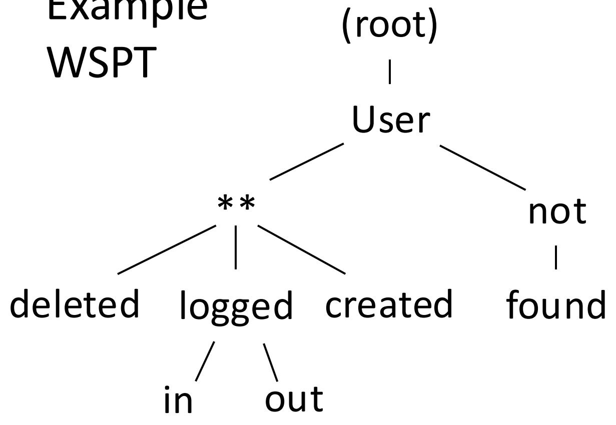
foo.c, line 148

```
zlog_debug("ip_sum 0x%x", iph->ip_sum);
```

Existing work on log matching

- WSPT (Word Segmented Prefix Tree) [4]
  - Prefix tree for words or wildcards
  - A wildcard node corresponds to only one word (i.e., no separators included)
    - WS-templates: satisfying this rule
  - Fast log matching
    - Reduce processing time for log classification by 1/10 compared to regex approach [4]

Example WSPT



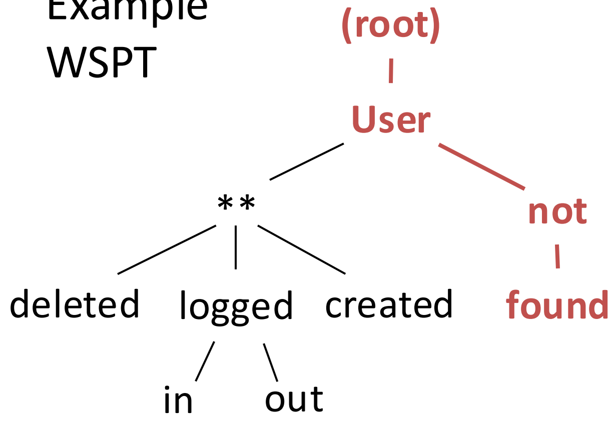
- User not found
- User Alice logged in
- User Bob created
- User Charlie deleted

[4] S. Kobayashi, et al. "amulog: A general log analysis framework for comparison and combination of diverse template generation methods," International Journal of Network Management, vol.32, no.4, p.e2195, 2022.

Existing work on log matching

- WSPT (Word Segmented Prefix Tree) [4]
  - Prefix tree for words or wildcards
  - A wildcard node corresponds to only one word (i.e., no separators included)
    - WS-templates: satisfying this rule
  - Fast log matching
    - Reduce processing time for log classification by 1/10 compared to regex approach [4]

Example WSPT



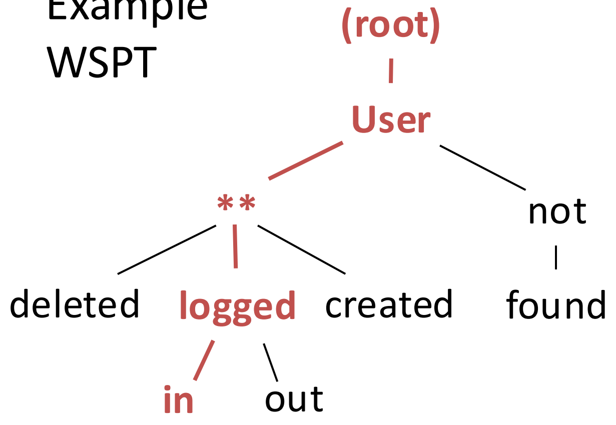
- **User not found**
- User Alice logged in
- User Bob created
- User Charlie deleted

[4] S. Kobayashi, et al. "amulog: A general log analysis framework for comparison and combination of diverse template generation methods," International Journal of Network Management, vol.32, no.4, p.e2195, 2022.

Existing work on log matching

- WSPT (Word Segmented Prefix Tree) [4]
  - Prefix tree for words or wildcards
  - A wildcard node corresponds to only one word (i.e., no separators included)
    - WS-templates: satisfying this rule
  - Fast log matching
    - Reduce processing time for log classification by 1/10 compared to regex approach [4]

Example WSPT



- User not found
- **User Alice logged in**
- User Bob created
- User Charlie deleted

[4] S. Kobayashi, et al. “amulog: A general log analysis framework for comparison and combination of diverse template generation methods,” International Journal of Network Management, vol.32, no.4, p.e2195, 2022.

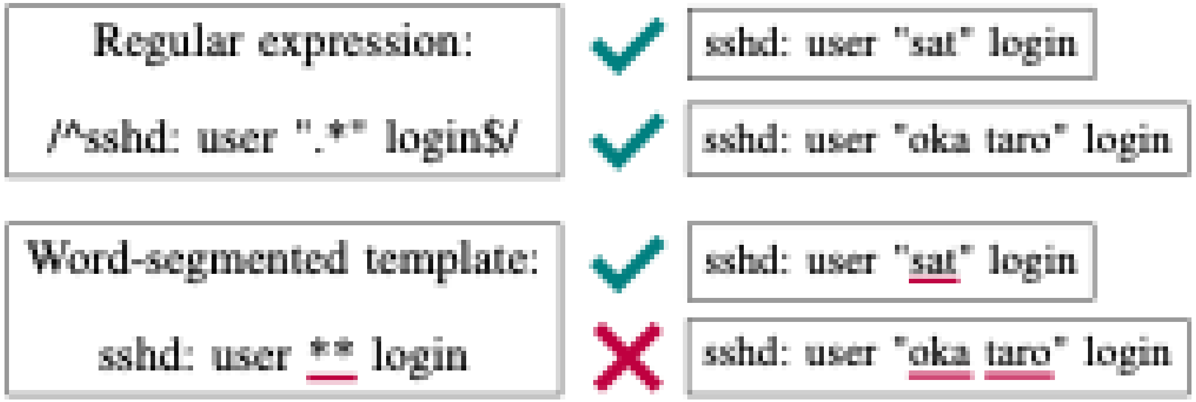
Problem of WSPT

- Log formats from source code are <u>not available as WS-templates</u> for WSPT
  - One format specifier may embed multiple words

  ➢ Not satisfying the assumption "<u>one wildcard corresponds to one word</u>"

Regex approach
Format specifiers can
<u>match multiple words</u>

WSPT approach
Wildcards can <u>match</u>
<u>only one word</u>



Regular expression:

```
/^sshd: user ".*" login$/
```

✓ sshd: user "sat" login

✓ sshd: user "oka taro" login

Word-segmented template:

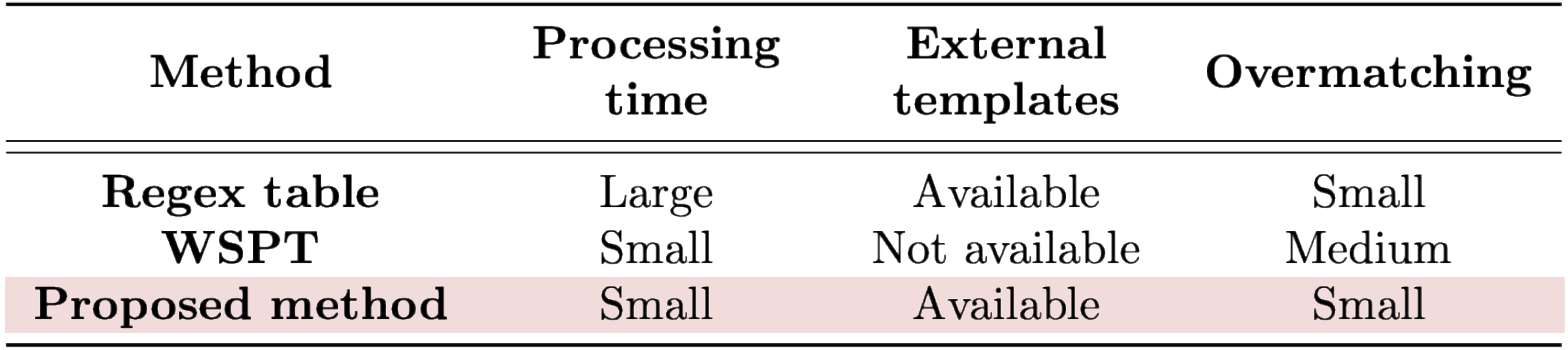
```
sshd: user ** login
```

✓ sshd: user "sat" login

✗ sshd: user "oka taro" login

Proposed approach

- Combining regex tables and WSPT
  - Generate WS-template after matching regex patterns and log messages
  - ➢Number of words is determined for the messages
- Fast processing of logs appearing after the second time

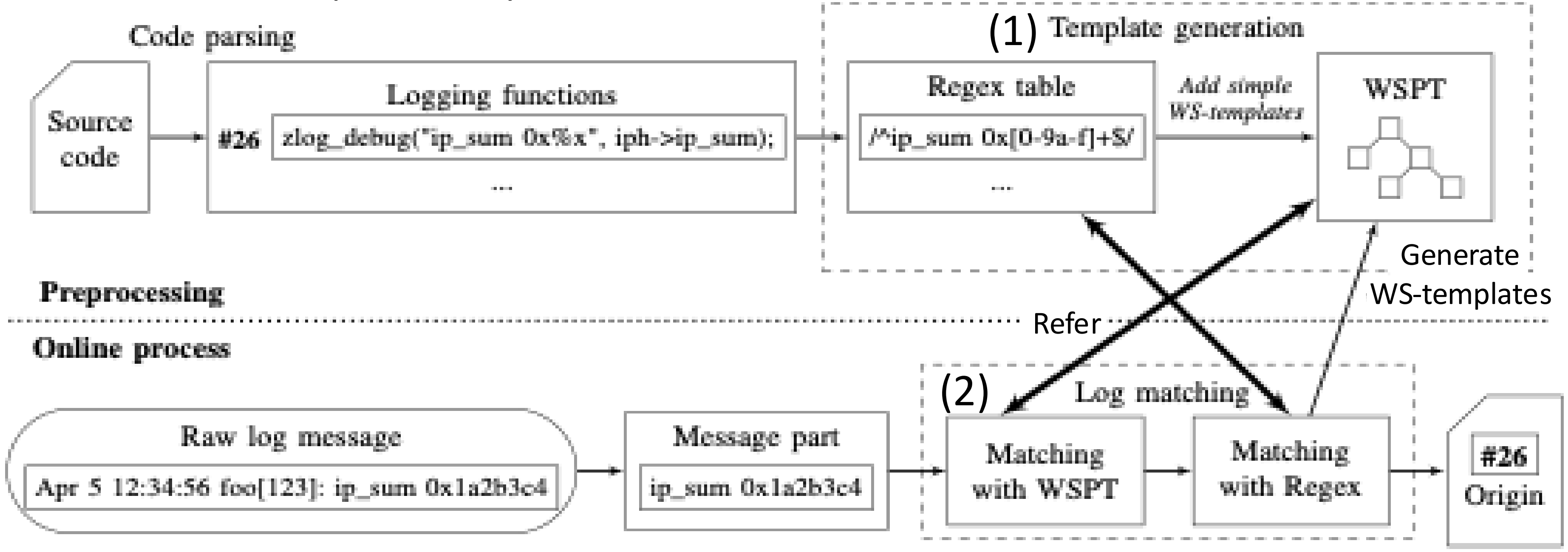


Method	**Processing time**	**External templates**	**Overmatching**
Regex table	Large	Available	Small
WSPT	Small	Not available	Medium
Proposed method	Small	Available	Small

Overview of proposed method

Generate regex patterns
from source code preliminary

Add simple WS-
templates to WSPT
(For fast processing)

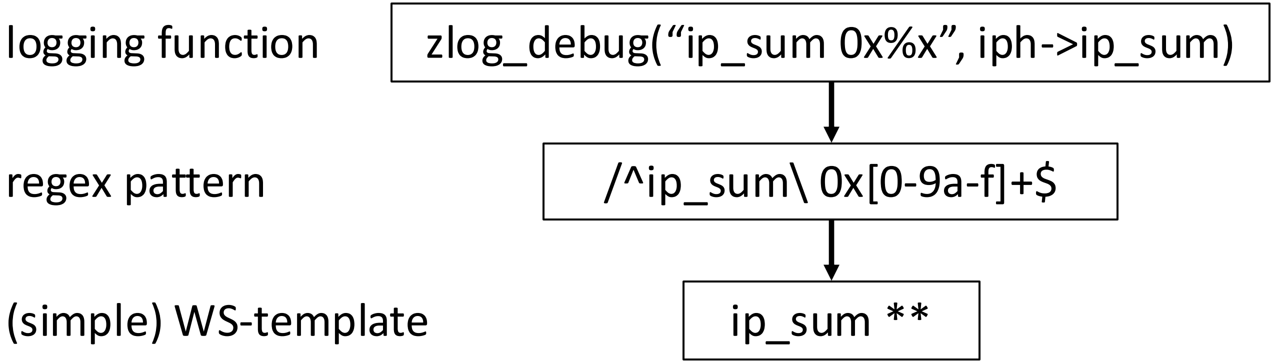


Use message part (no header)
in the input log messages

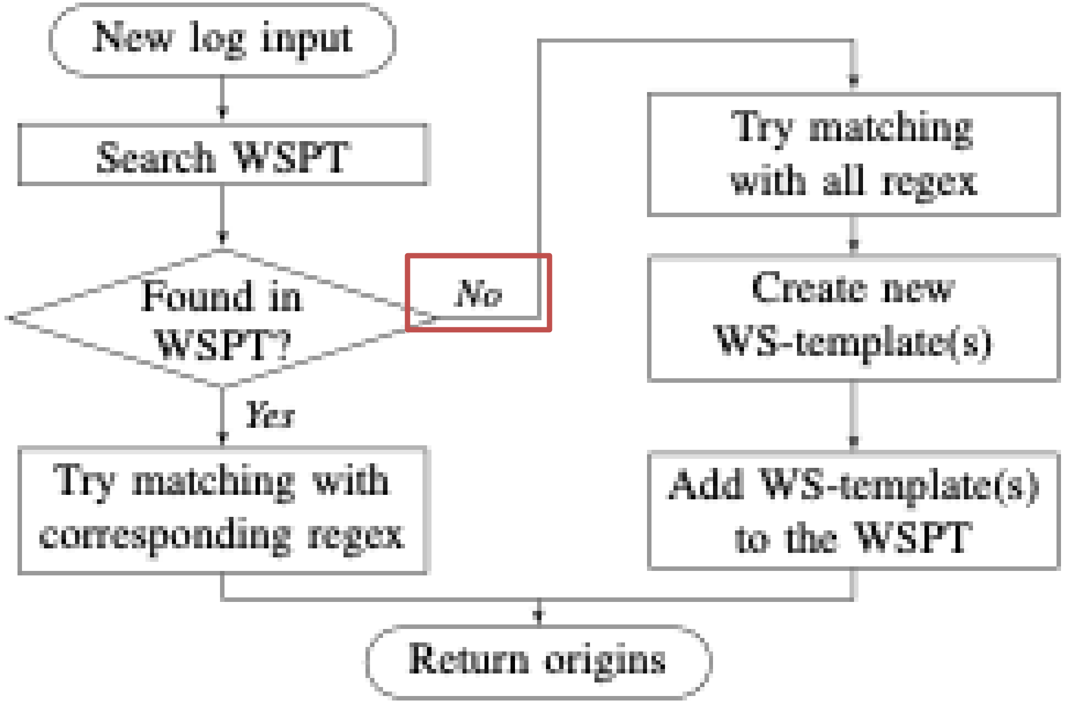
Log matching combining
regex and WSPT

(1) Template generation

1. Extract logging functions from source code
2. Replace format specifiers into regex patterns
3. Generate simple WS-template that has one wildcard for one format specifier



(2) Log matching


New log input
Search WSPT
Found in WSPT?
No
Yes
Try matching with corresponding regex
Try matching with all regex
Create new WS-template(s)
Add WS-template(s) to the WSPT
Return origins

Input

ip_sum 0x1a2b3c4

ip_sum 0x2b3c4d5

…

Regex table

/^ip_sum 0x[0-9a-f]+$/

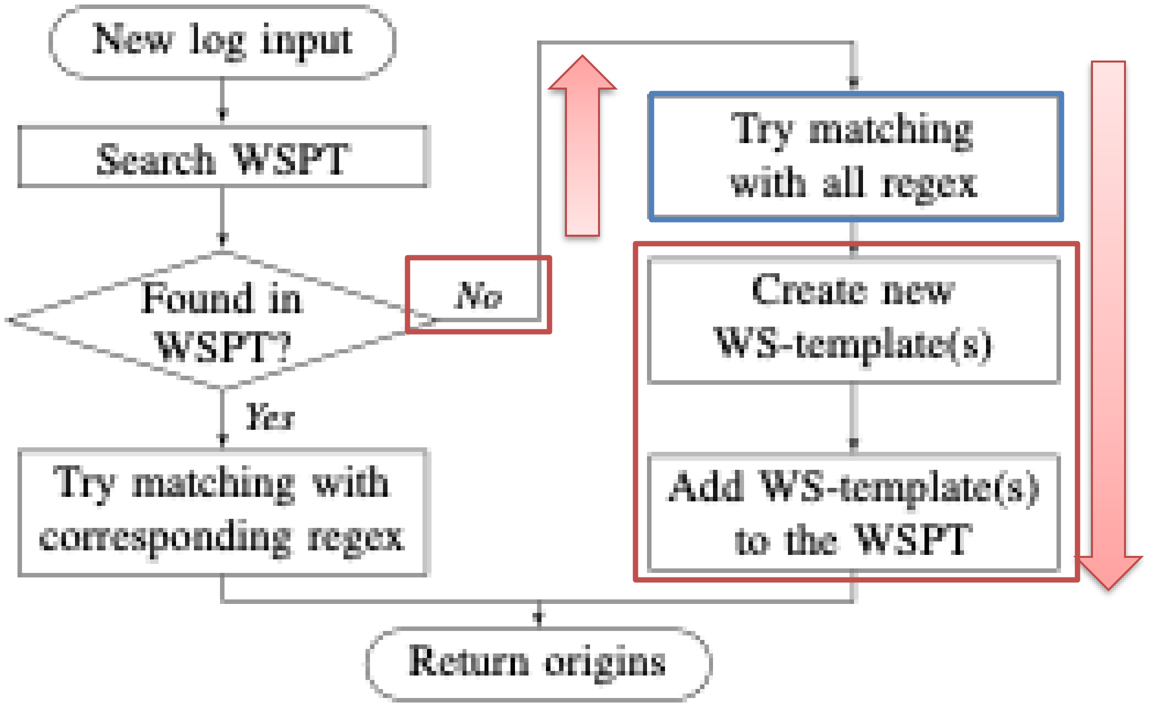
/^Terminating on signal$/

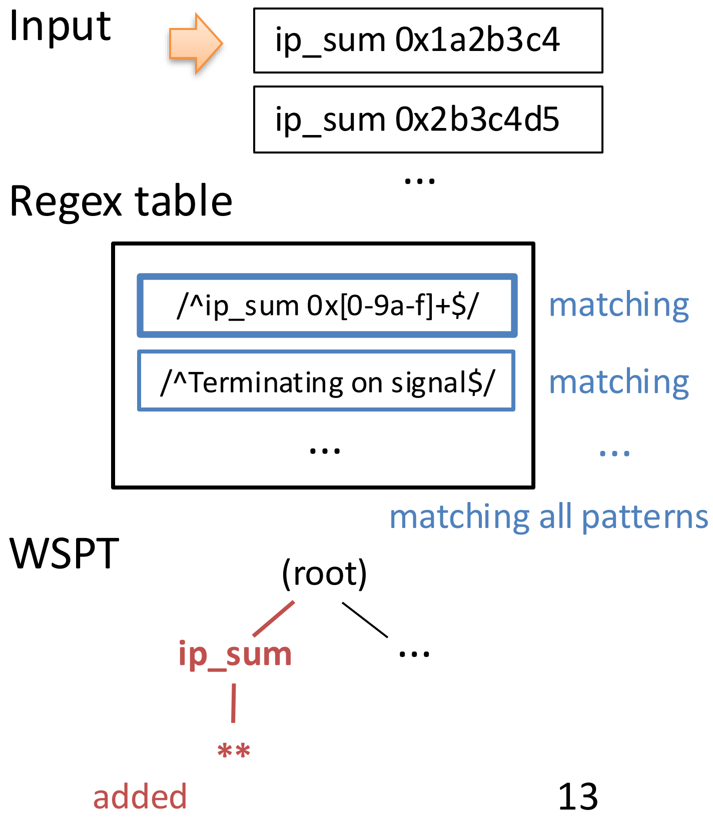
…

WSPT

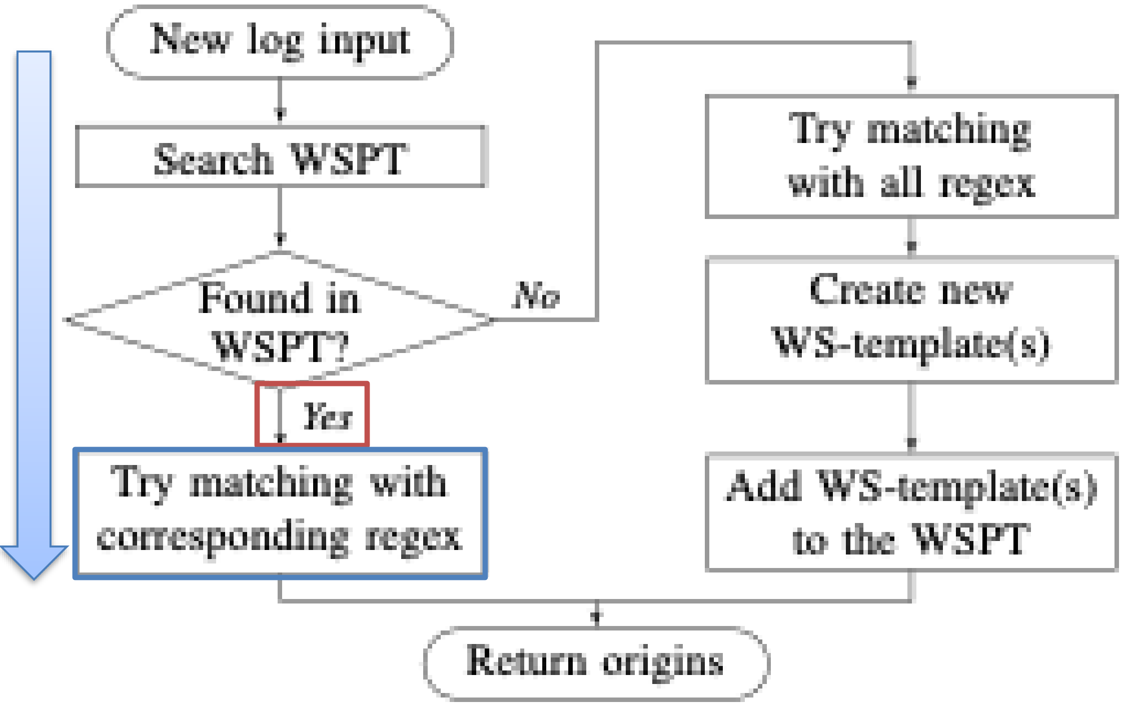
no matching template

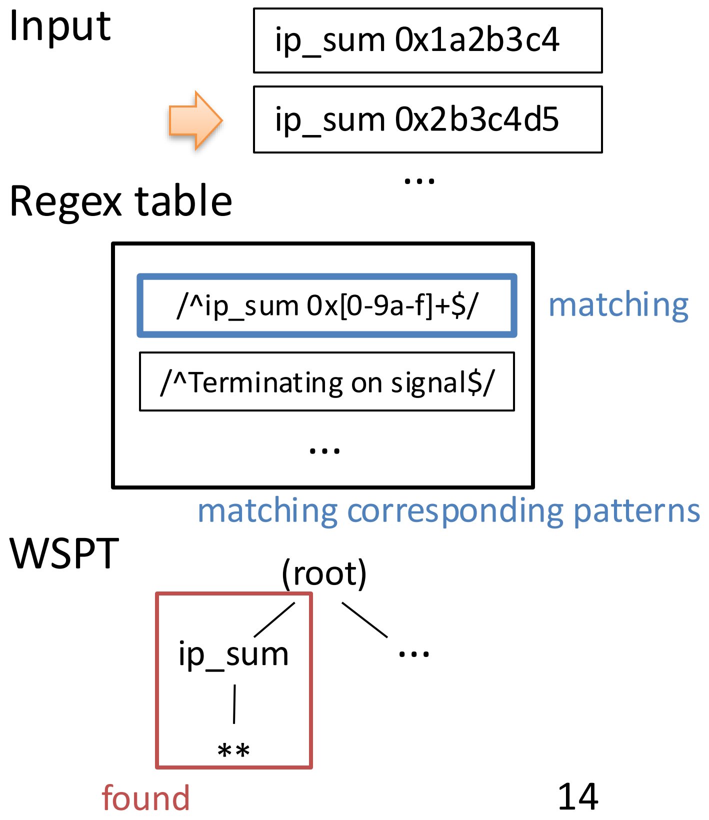
(2) Log matching


New log input
Search WSPT
Found in WSPT?
No
Yes
Try matching with corresponding regex
Try matching with all regex
Create new WS-template(s)
Add WS-template(s) to the WSPT
Return origins



(2) Log matching


New log input
Search WSPT
Found in WSPT?
No
Yes
Try matching with corresponding regex
Try matching with all regex
Create new WS-template(s)
Add WS-template(s) to the WSPT
Return origins



Evaluation setup

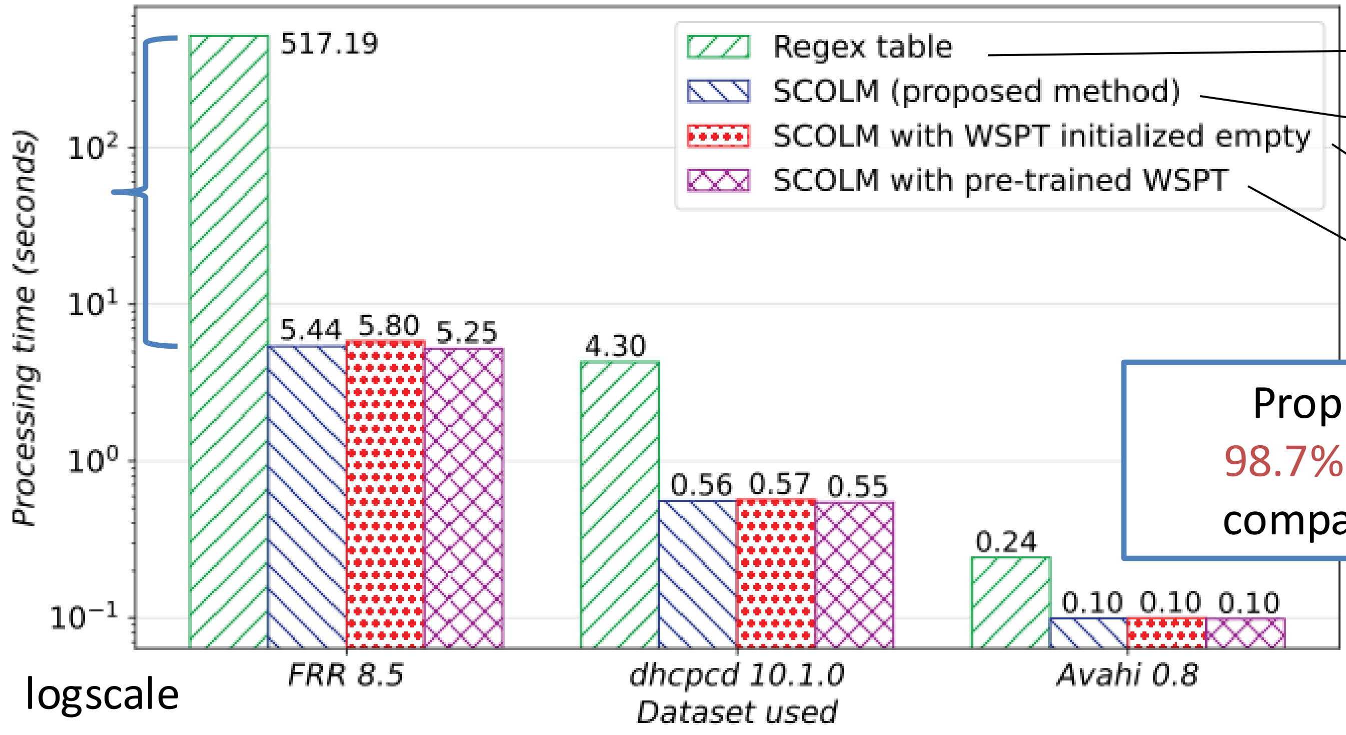
- Implementation
  - SCOLM: Implemented in Python 3.10, available in GitHub

    http://github.com/3atlab/scolm
- Datasets

Software	**# of regex tpl.**	**# of WS tpl.**	**# of logs available**	**How to collect logs**
FRRouting (FRR) 8.5	6,481	6,346	89,856	Network emulation [23]
dhcpcd 10.1.0	586	580	5,509	Home server
Avahi 0.8	367	363	1,971	Lab server

[23] C. Regal-Mezin, et al. "netroub: Towards an emulation platform for network trouble scenarios", in CoNEXT-SW, pp.17-18, 2023.

Comparison on processing time



logscale

Existing method

Proposed method

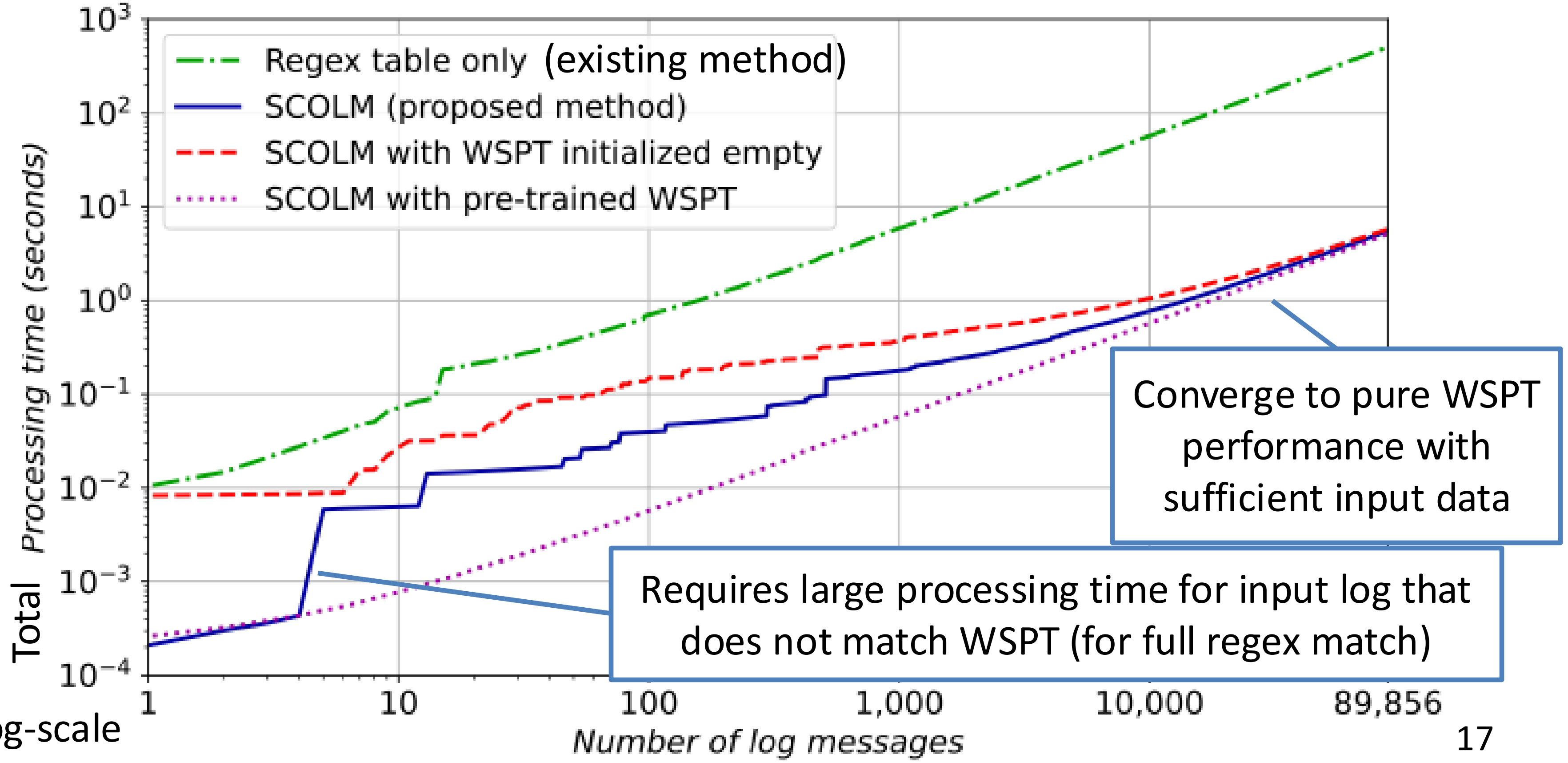
Omitting some steps

Always using WSPT

Proposed method reduces 98.7% processing time in FRR compared to existing method

Measurement in AMD Ryzen 7 7700, Ubuntu Server 22.04 (x86_64), 64GB memory

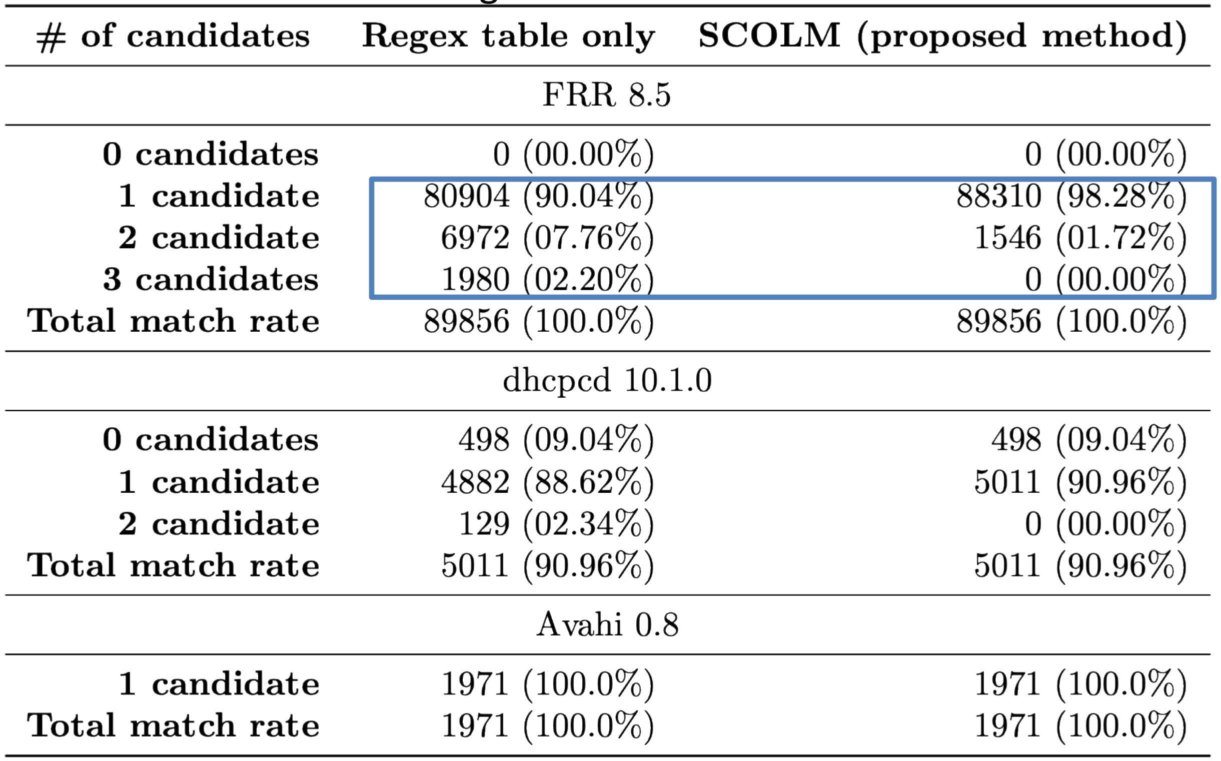
Processing time for each log message



log-log-scale

Number of obtained candidates

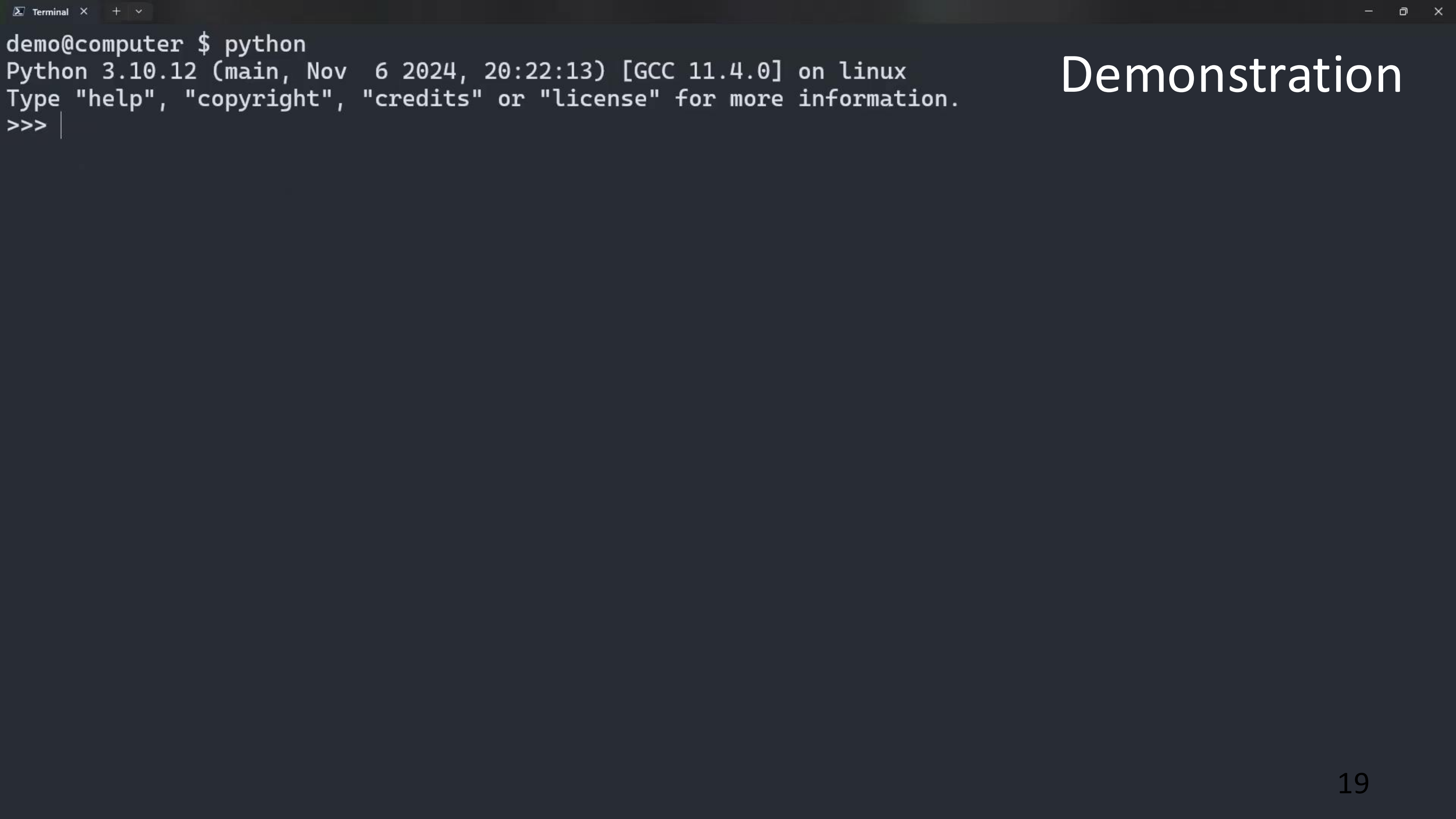
Existing method



# of candidates	Regex table only	SCOLM (proposed method)
	FRR 8.5	
0 candidates	0 (00.00%)	0 (00.00%)
1 candidate	80904 (90.04%)	88310 (98.28%)
2 candidate	6972 (07.76%)	1546 (01.72%)
3 candidates	1980 (02.20%)	0 (00.00%)
Total match rate	89856 (100.0%)	89856 (100.0%)
	dhcpcd 10.1.0	
0 candidates	498 (09.04%)	498 (09.04%)
1 candidate	4882 (88.62%)	5011 (90.96%)
2 candidate	129 (02.34%)	0 (00.00%)
Total match rate	5011 (90.96%)	5011 (90.96%)
	Avahi 0.8	
1 candidate	1971 (100.0%)	1971 (100.0%)
Total match rate	1971 (100.0%)	1971 (100.0%)

WSPT removes candidates with different number of words in variable part

SCOLM reduces False Positives than Regex approach



```
demo@computer $ python
Python 3.10.12 (main, Nov  6 2024, 20:22:13) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Demonstration

Summary

- Design and develop a system to automatically identify original logging functions for helping network troubleshooting
  - Fast log matching combining regex tables and WSPT
- Evaluate processing time and number of obtained candidates using SCOLM with three datasets
  - Reduce 98.7% processing time than regex table approach
  - Reduce false positives of obtained results