CIS600 Presentation

Jingyuan Chen, Yifei Guo, Shu Li, Yanchen Li, Zirui Liu, Wenzhe Ma, Qingyuan Mao

Introduction

XiaoFish - Natural language executes terminal command system

Terminal command line interface (CLI) has become the preferred tool for many developers and system administrators.

1.difficult for most non-technical users

2.spend a lot of time learning and remembering complex terminal commands

Data Cleaning Process

1.cleaned up the ID column. Converting all IDs to lowercase and removing spaces

2.de-duplicated the data, keeping only the first record for each ID

3.remove redundant noise by cleaning the text of commands and descriptions

4.removed the brackets and their contents

5.extracted the keywords describing the text using TF-IDF

Suppose we have the following data:

ID: Example_123

Command: ping 192.168.1.1 -c 4

Description: Send 4 packets to a specific IP address (e.g., 192.168.1.1)

Cleaned up:

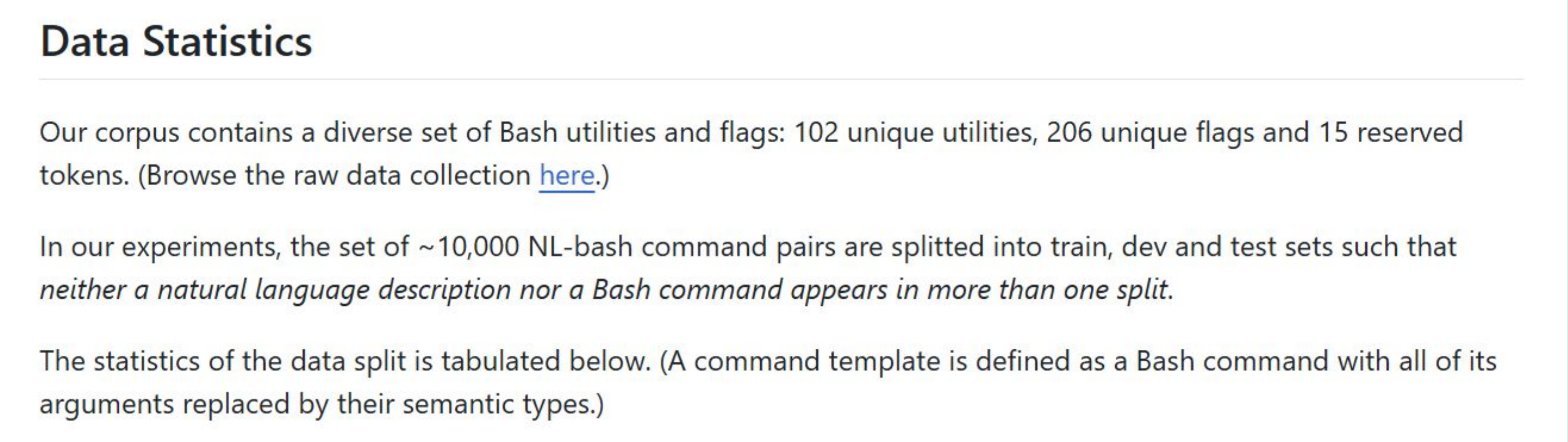
IDs were standardized to lowercase: example_123

Command cleaned up to: ping <IP_ADDRESS> <FLAG> <NUM>

Description cleaned up to: Send 4 packets to a specific IP address

The keyword extracted using TF-IDF is: send packets specific address

Dataset Generation



Data Statistics

Our corpus contains a diverse set of Bash utilities and flags: 102 unique utilities, 206 unique flags and 15 reserved tokens. (Browse the raw data collection here.)

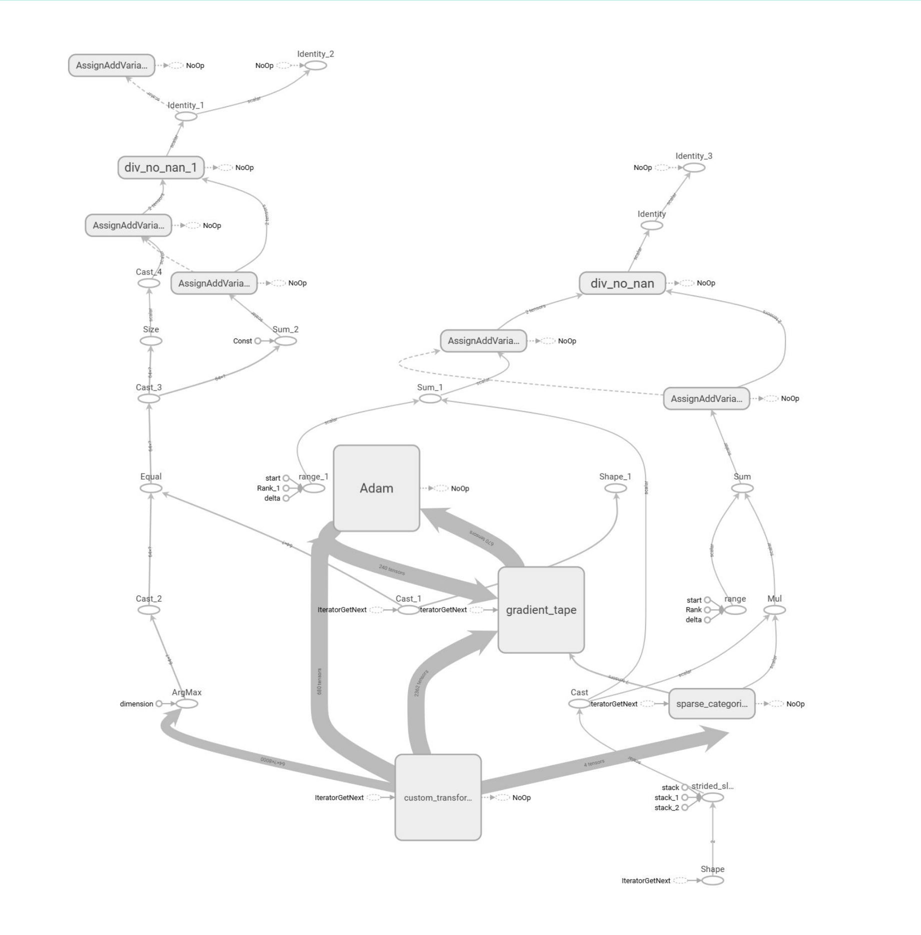
In our experiments, the set of ~10,000 NL-bash command pairs are splitted into train, dev and test sets such that *neither a natural language description nor a Bash command appears in more than one split*.

The statistics of the data split is tabulated below. (A command template is defined as a Bash command with all of its arguments replaced by their semantic types.)

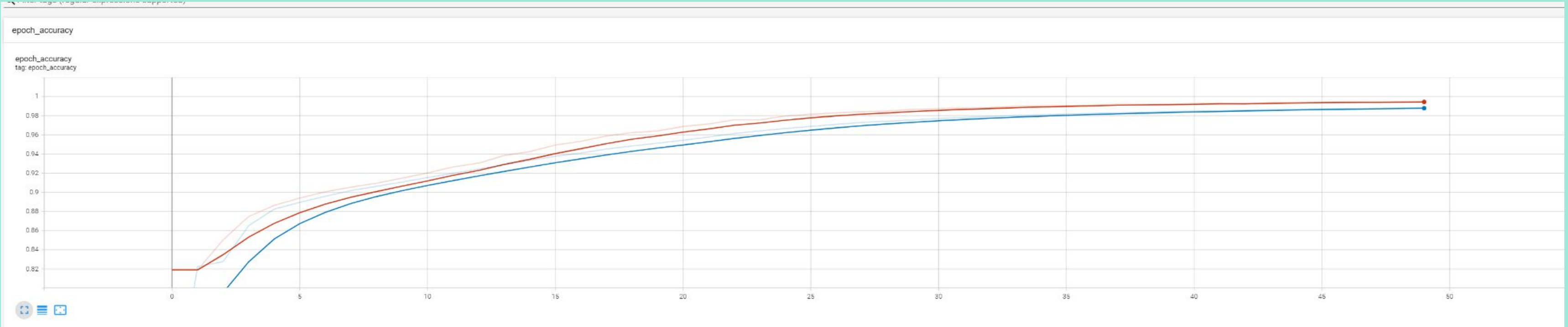
Model Architecture

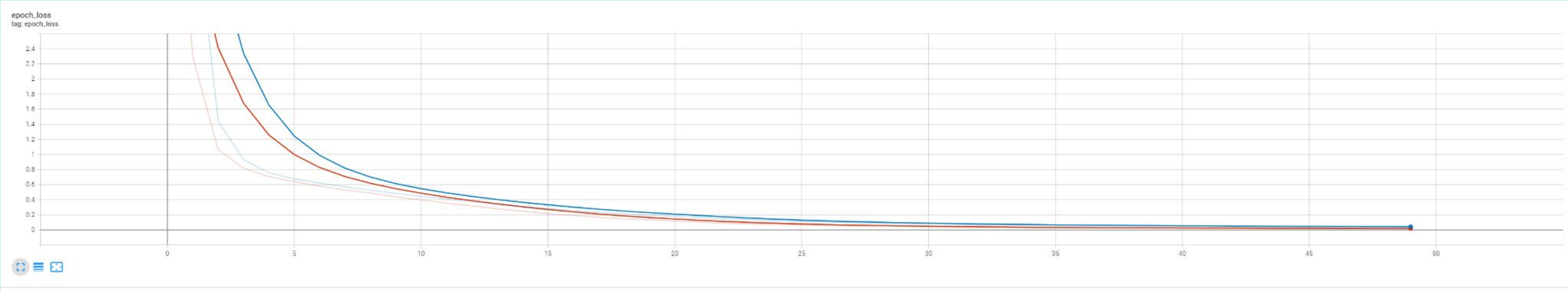
Transformer model: about 5.7M parameter

4 number of layers
8 number of heads
128 dimension of model
512 dimension of feed forward network



Training


epoch_accuracy
epoch_accuracy
tag: epoch_accuracy
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0
5
10
15
20
25
30
35
40
45
50


epoch_loss
tag: epoch_loss
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35
40
45
50

Evaluation and Testing

- Bilingual Evaluation Understudy(BLEU)

- Functional Testing

BLEU

- N-gram Precision

$$Geometric\ Average\ Precision(N) = \exp\left(\sum_{n=1}^{N} W_n \log p_n\right)$$

$$= \prod_{n=1}^{N} {p_n}^{W_n}$$

- Brevity Penalty

$$= (p_1)^{1/4} \cdot (p_2)^{1/4} \cdot (p_3)^{1/4} \cdot (p_4)^{1/4}$$

$$Brevity\ Penalty = \begin{cases} 1, & if\ c > r \\ e^{(1-\frac{r}{c})}, & if\ c \leq r \end{cases}$$

c is *predicted length = number of words in the predicted sentence*

r is *target length = number of words in the target sentence*

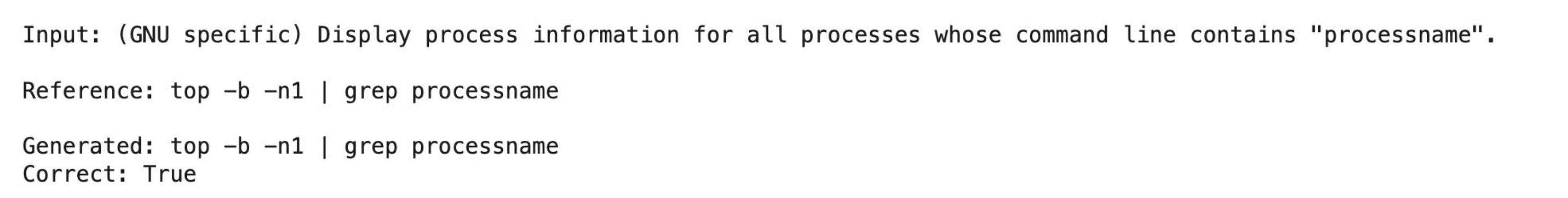
$$Bleu(N) = Brevity\ Penalty\ * Geometric\ Average\ Precision(N)$$

Functional Testing

- Execute both the reference commands and the model-generated
- Capture standard output & the error output
- Compare & Match
- Accuracy

Functional Testing

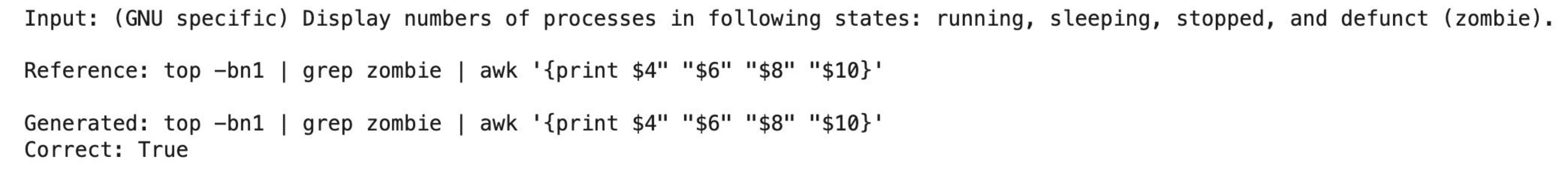
- Sample Output



```
Input: (GNU specific) Display process information for all processes whose command line contains "processname".

Reference: top -b -n1 | grep processname

Generated: top -b -n1 | grep processname
Correct: True
```

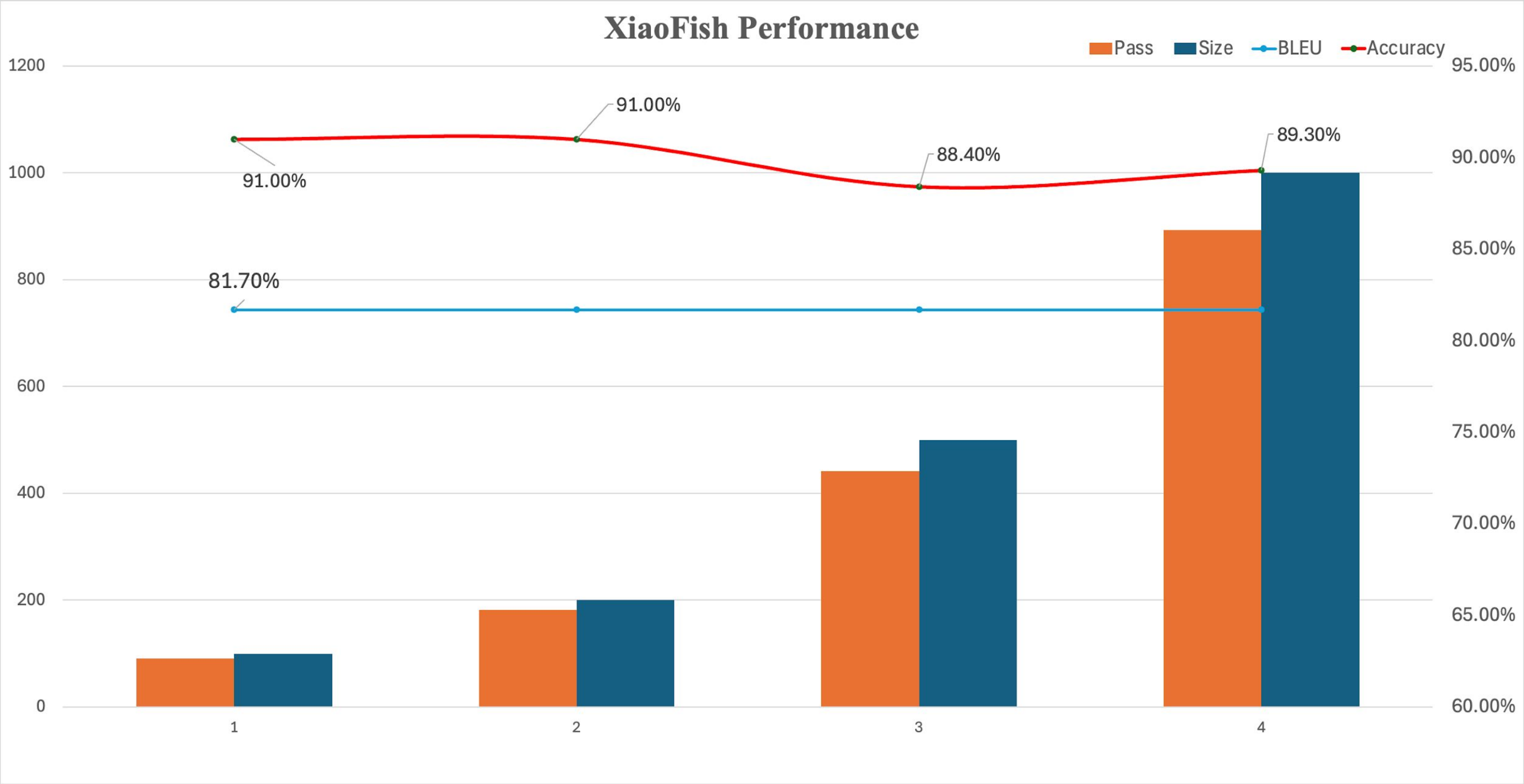


```
Input: (GNU specific) Display numbers of processes in following states: running, sleeping, stopped, and defunct (zombie).

Reference: top -bn1 | grep zombie | awk '{print $4" "$6" "$8" "$10}'

Generated: top -bn1 | grep zombie | awk '{print $4" "$6" "$8" "$10}'
Correct: True
```

Results


XiaoFish Performance
Pass
Size
BLEU
Accuracy
1200
1000
800
600
400
200
0
95.00%
90.00%
85.00%
80.00%
75.00%
70.00%
65.00%
60.00%
91.00%
91.00%
88.40%
89.30%
81.70%
1
2
3
4

Future Directions

Pretrained weights

loss function

Questions?