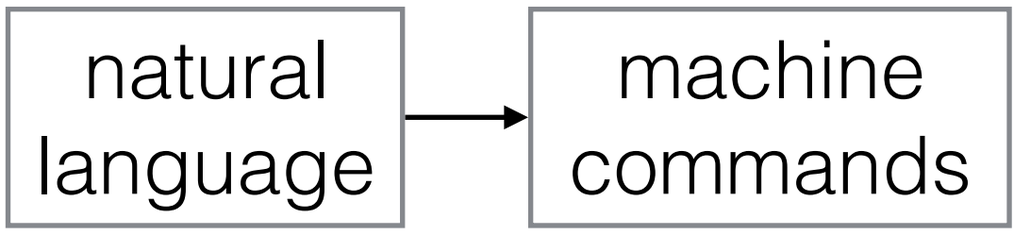
NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System

Victoria Lin Chenglong Wang Luke Zettlemoyer
Michael D. Ernst
{xilin,clwang,lsz,mernst}@cs.washington.edu

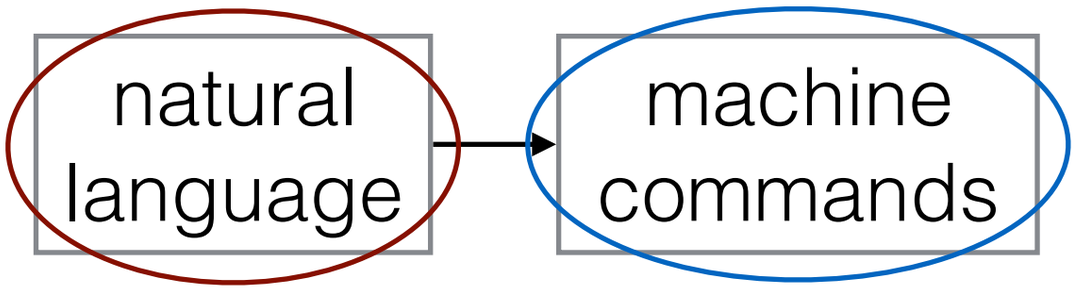
OUTLINE

Problem Definition



OUTLINE

Problem Definition



convenient

???

powerful

OUTLINE

Problem Definition

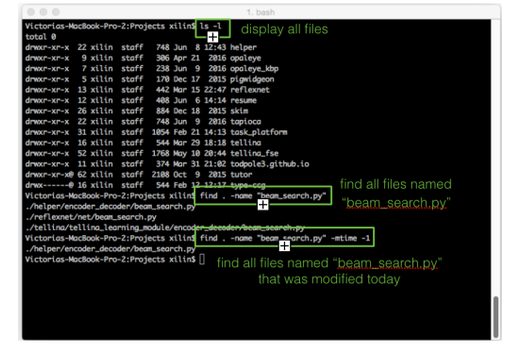
natural language → machine commands

convenient


???

powerful

Domain



OUTLINE

Problem Definition

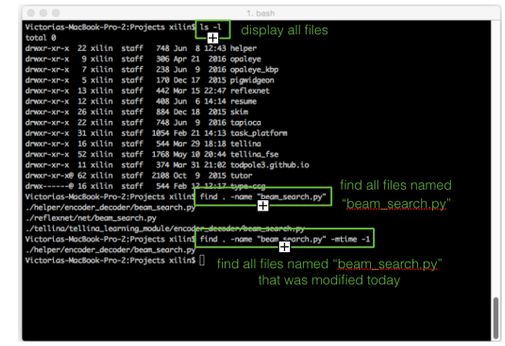
natural language → machine commands

convenient

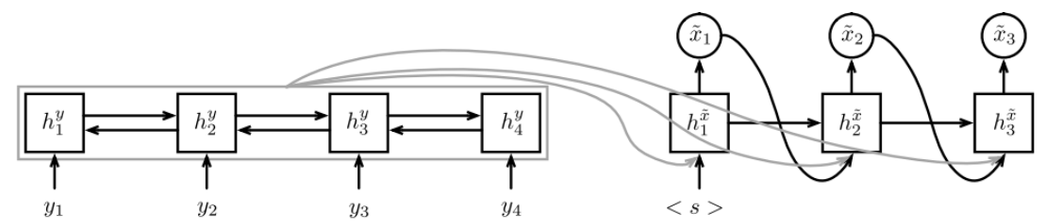
???

powerful

Domain



Data-Driven Approaches



Adaptions from state-of-the-art neural machine translation models

OUTLINE

Problem Definition

natural language → machine commands

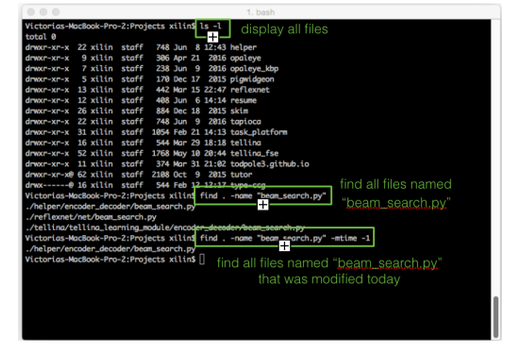
convenient

???

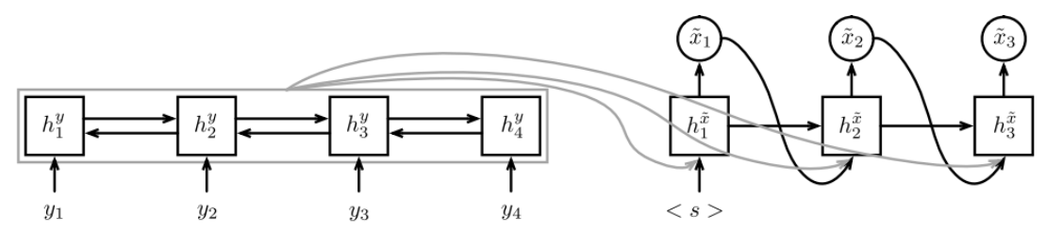
powerful

Corpus Construction

Domain



Data-Driven Approaches



Adaptions from state-of-the-art neural machine translation models

OUTLINE

Problem Definition

natural language → machine commands

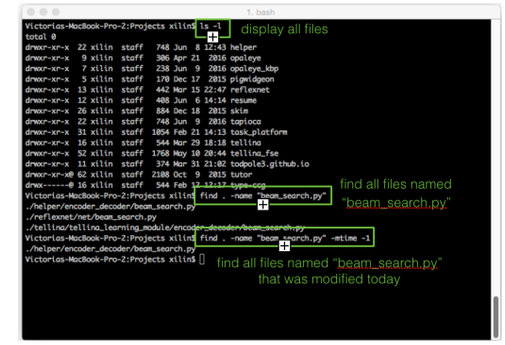
convenient

???

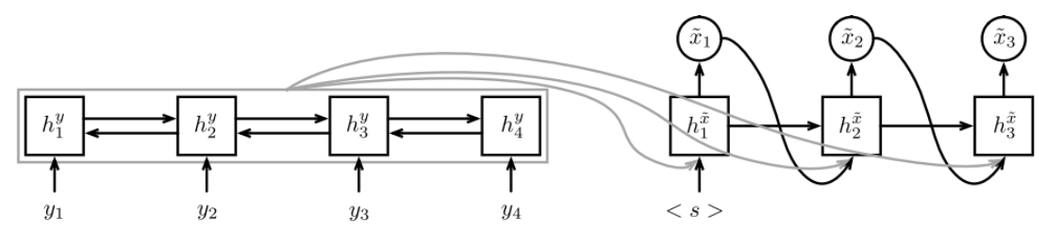
powerful

Corpus Construction

Domain



Data-Driven Approaches



Adaptions from state-of-the-art neural machine translation models

System Performance

Qualitative Analysis

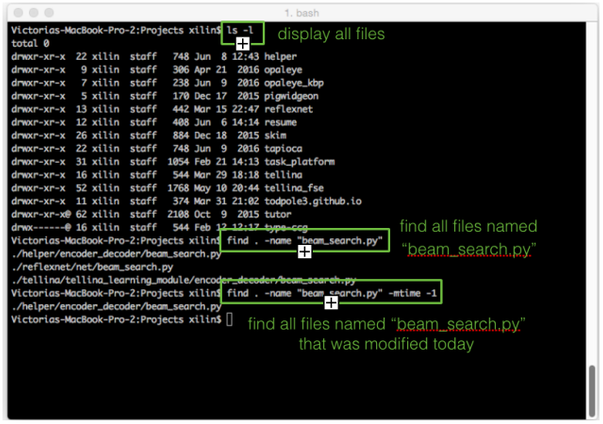
Live Demo

PROBLEM DEFINITION

- Natural Language → Command Translation
  - Generating ***one-liners***
    - In most command languages complex semantics can be represented in short syntactic forms
    - Other work: code block generation (Polosukhin and Skidanov ’18)
  - ***Single-turn interaction*** between the user & the system (building block for multi-turn system)
    - Other work: conversational natural language programming assistant (Pandita et. al. ’18)
    - Semantic parsing can be a building block conversational programming assistant

DOMAIN - BASH

- Potentially Wide User Base
  - Most Linux users know bash, but not mastering it
- Command Interface Language
- Generalizable to other command languages



BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep
"Salesforce" {}
```

BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep
"Salesforce" {}
```

Head command

BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}
```

Flag

BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

```
find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}
```

Argument

BASH EXAMPLE

- find all '*.c' files under $HOME directory which contain the string “Salesforce"

find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "Salesforce" {}

Compound Commands

RELATED WORK

- Neural Networks: Natural Language → Formal Languages
  - ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
  - ✓ NL → Regular expression (Locascio et. al. ’16)
  - ✓ NL → Logical forms (Li & Lapata ’16)
  - ✓ NL → Python (Wang et. al. ’16)
  - ✓ NL → Python (Yin & Neubig ’17, Rabinovich et. al. ’17)

Rule-Based Systems	Statistical Models over Discrete Structures

RELATED WORK

- Neural Networks: Natural Language → Formal Languages
  - ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
  - ✓ NL → Regular expression (Locascio et. al. ’16)
  - ✓ NL → Logical forms (Li & Lapata ’16)
  - ✓ NL → Python (Wang et. al. ’16)
  - ✓ NL → Python (Yin & Neubig ’17, Rabinovich et. al. ’17)

Adapted from NMT methods for natural language translation

RELATED WORK

- Neural Networks: Natural Language → Formal Languages

Seq2Seq

- ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
- ✓ NL → Regular expression (Locascio et. al. ’16)
- ✓ NL → Logical forms (Li & Lapata ’16)
- ✓ NL → Python (Wang et. al. ’16)

Seq2Tree

- ✓ NL → Python (Yin & Neubig ’17, Rabinovich et. al. ’17)

Expressive —> Simplest Data Representation

RELATED WORK

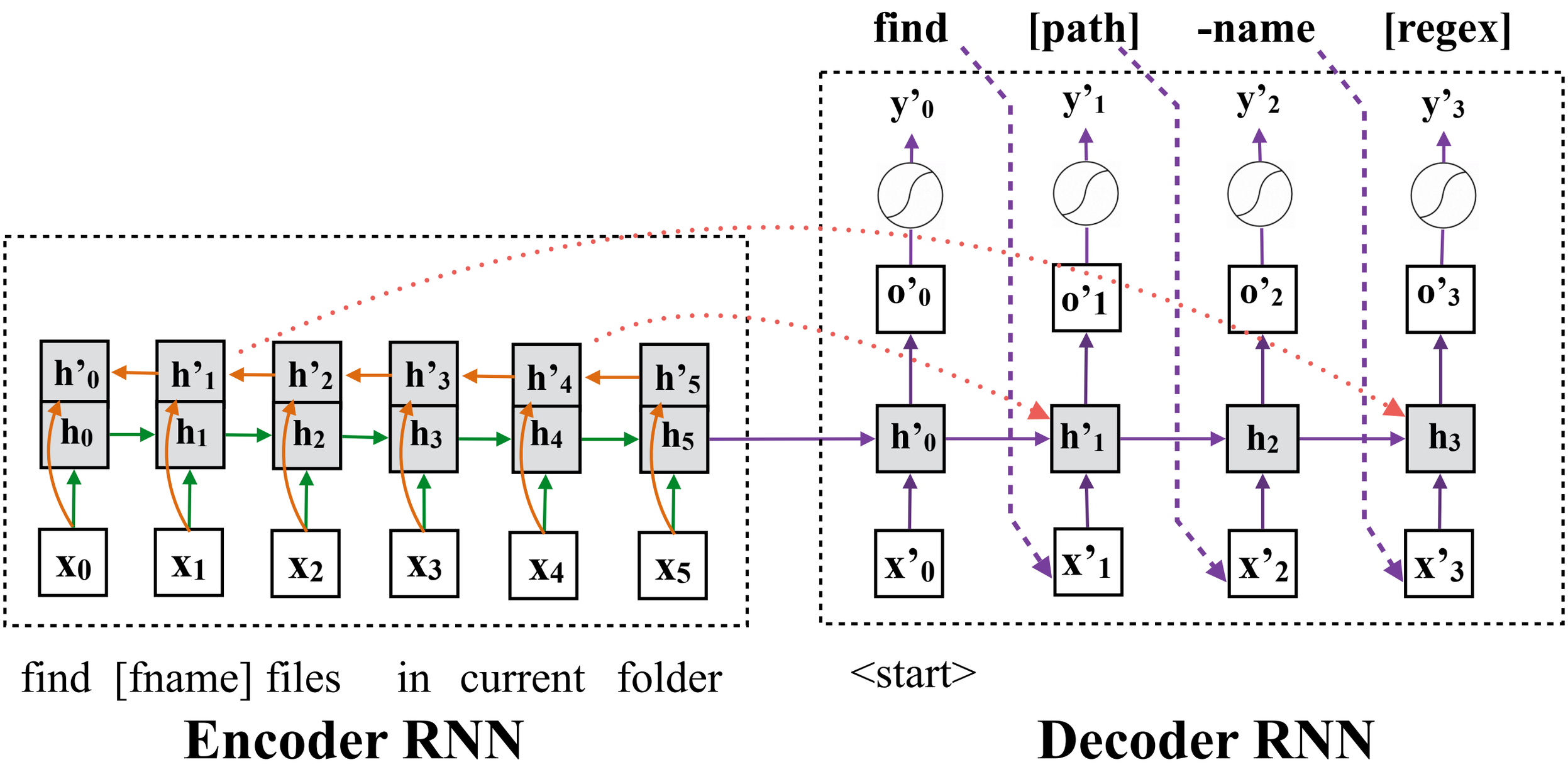
- Neural Networks: Natural Language → Formal Languages

Seq2Seq

- ✓ NL → Syntactic parse trees (Vinyals et. al. ’14)
- ✓ NL → Regular expression (Locascio et. al. ’16)
- ✓ NL → Logical forms (Li & Lapata ’16)
- ✓ NL → Python (Wang et. al. ’16)

Target Domain: Shallow Syntax Structure (No Formal Grammar)

SEQUENCE-TO-SEQUENCE NEURAL NETWORK



SEQ2SEQ + COPYING

- find all **'*.c'** files under **$HOME** directory whose content has the string **“salesforce"**

  find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}

  ✗ Large number of out-of-vocabulary words (arguments)

SEQ2SEQ + COPYING

- find all **'*.c'** files under **$HOME** directory whose content has the string **“salesforce"**

  find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}

  ✗ Large number of out-of-vocabulary words

CopyNet

Incorporating Copying Mechanism in Sequence-to-Sequence Learning, Gu et. al. EMNLP 2016

SEQ2SEQ + COPYING

- find all **'*.c'** files under **$HOME** directory whose content has the string **“salesforce"**

  find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}

  ✗ Many command arguments are source tokens transformed through atomic string edits

CopyNet

Incorporating Copying Mechanism in Sequence-to-Sequence Learning, Gu et. al. EMNLP 2016

SEQ2SEQ + COPYING

- find all **'*.c'** files under **$HOME** directory whose content has the string **“salesforce"**

  find "$HOME" -name "*.c" -print0 | xargs -0 -I {} grep "salesforce" {}

  ✗ Many command arguments are source tokens transformed through atomic string edits

Character models? Very long sequences…

SUB-TOKEN COPYING

- find all **‘ * . c '** files under **$ HOME** directory whose content has the string **“ salesforce "**

find " $ HOME " -name " * . c " -print0 | xargs -0 -I {} grep " salesforce " {}

Split the constant tokens in both the source and target sequences into a sequence of sub-tokens consists of the following:

1. Consecutive sub-sequences of alphabetical letters
2. Consecutive sub-sequences of digits
3. All other special tokens

Run CopyNet on the sub-tokens

SUB-TOKEN COPYING

- find all **‘ * . c '** files under **$ HOME** directory whose content has the string **“ salesforce "**

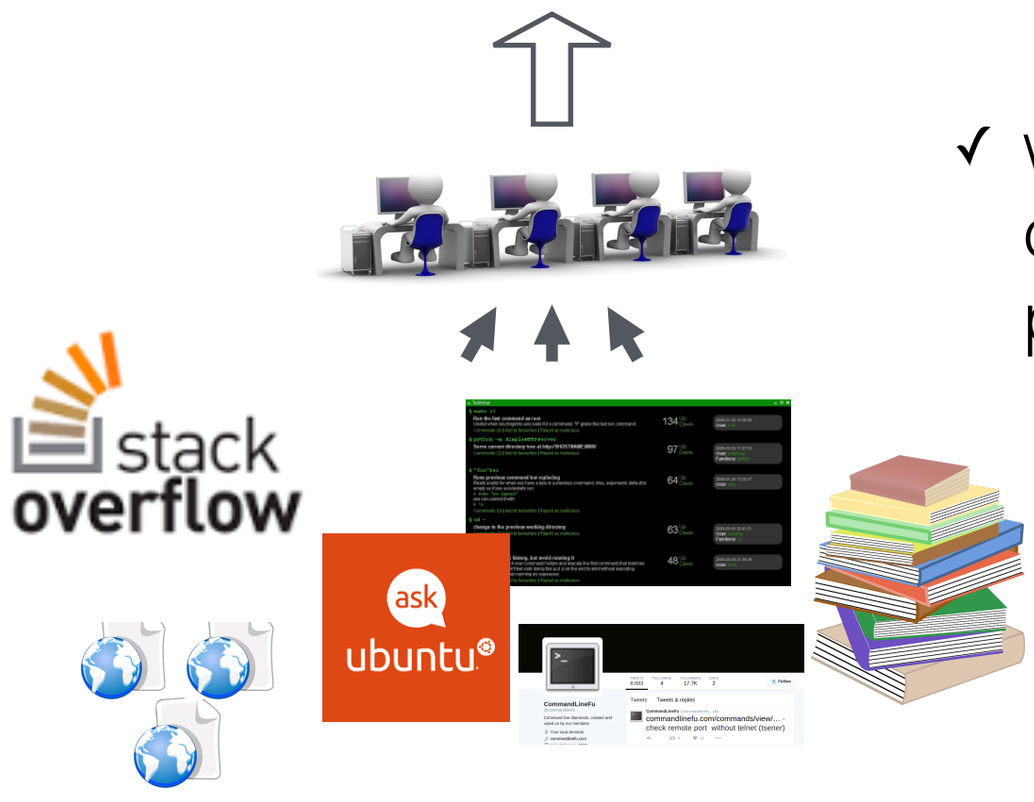
find " $ HOME " -name " * . c " -print0 | xargs -0 -I {} grep " salesforce " {}

Enables learning of

1. Substring addition
2. Substring deletion
3. Substring replacement
4. Semantics of the special characters such as “$”, quotation marks, “*”, etc.

DATA COLLECTION

- Bash programmers hired upwork™
- Collect bash commands and their natural language descriptions from the web



✓ web interface to control the collection process

BASH COMMAND FILTERING

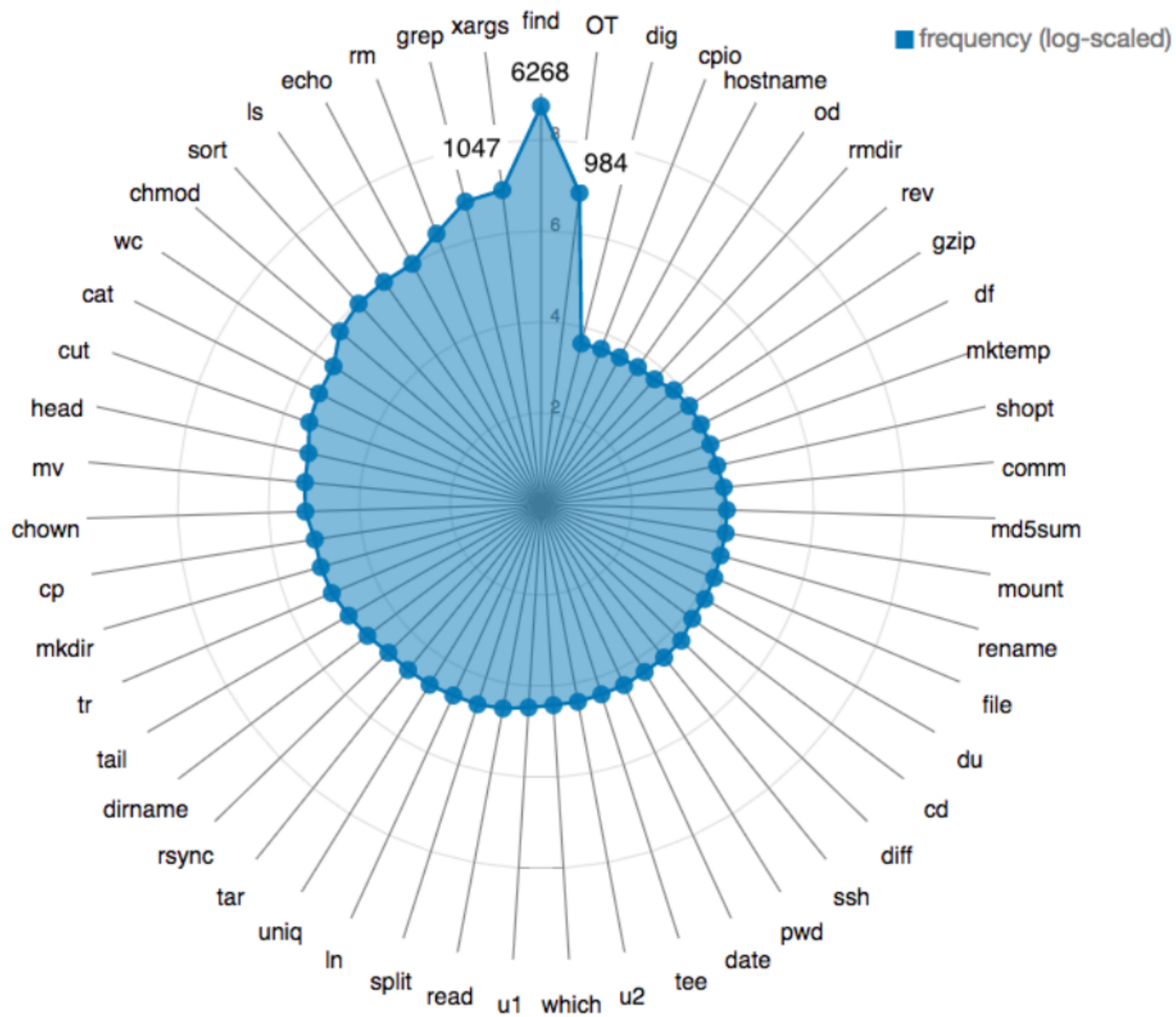
- Bash Command

In-scope	Single command	
	Logical connectives	&&, ||, ()
	Nested command	pipeline |
		command substitution $()
		process substitution <()
Out-of-scope	I/O redirection	<, <<
	Variable assignment	=
	Parameters	e.g. $1, $HOME
	Multi-statement	if, for, while, until, etc.
	Regex structure	e.g. x*y*
	Non-bash programs	triggered by awk, java, etc.

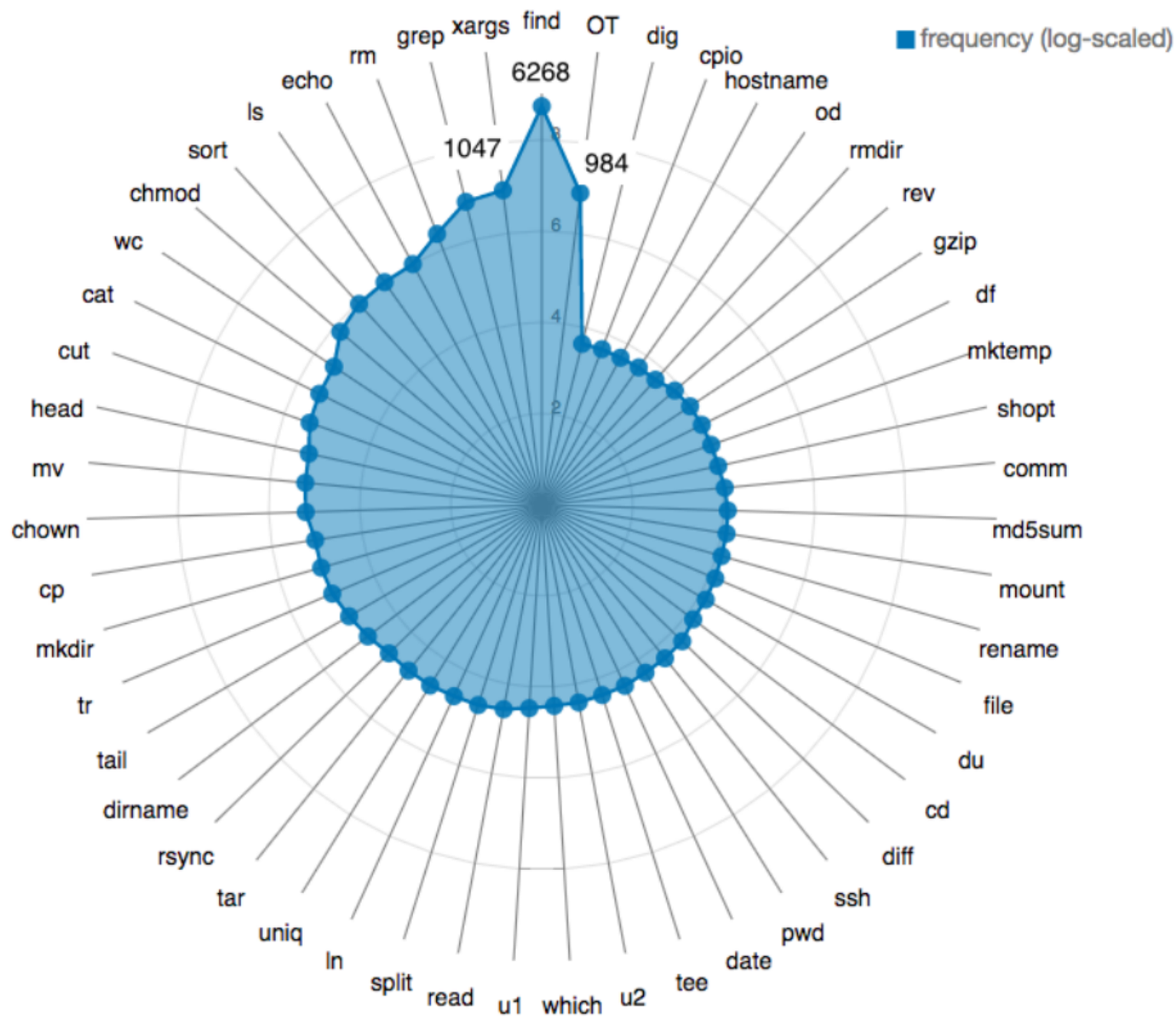
DATA STATISTICS

- 12,609 pairs —> 9,301 pairs after filtering
- 8,090 train, 609 dev, 606 test
- 100+ unique bash commands, 537 unique flags

TOP-50 COMMAND HISTOGRAM



TOP-50 COMMAND HISTOGRAM



The rest combined: 984

EVALUATION METHODOLOGY

- Manual Evaluation (Multiple Correct Solutions)
  - 3 bash programmers (hired via upwork™) judged the top-3 suggestions of each test example
    - Full command correctness
    - Command template correctness

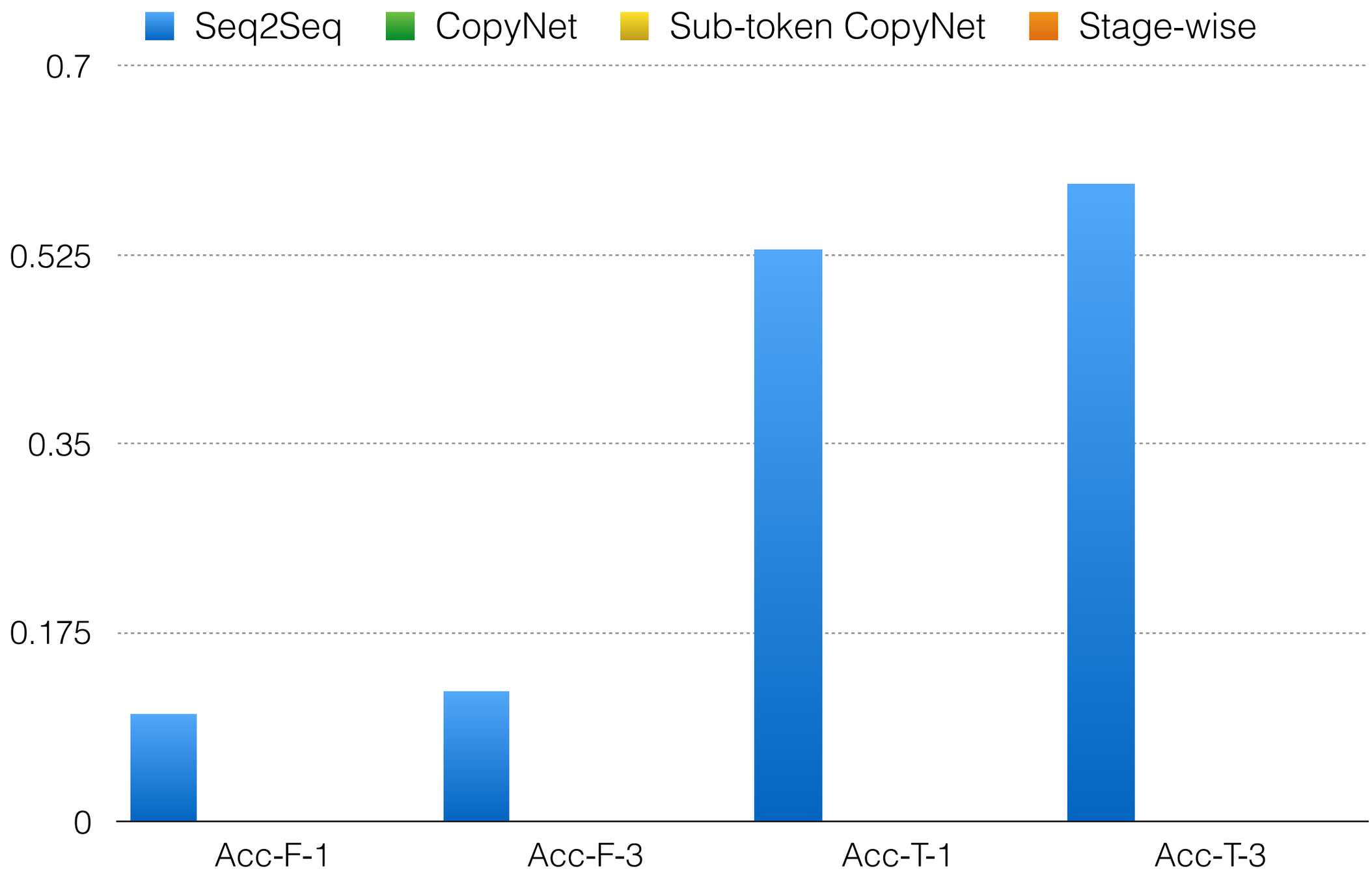
find [path] -name [regex] -print0 | xargs -0 -I {} grep [regex] {}

  - Final judgement: majority vote
  - Inter-annotator agreement: 0.89, 0.83, 0.80

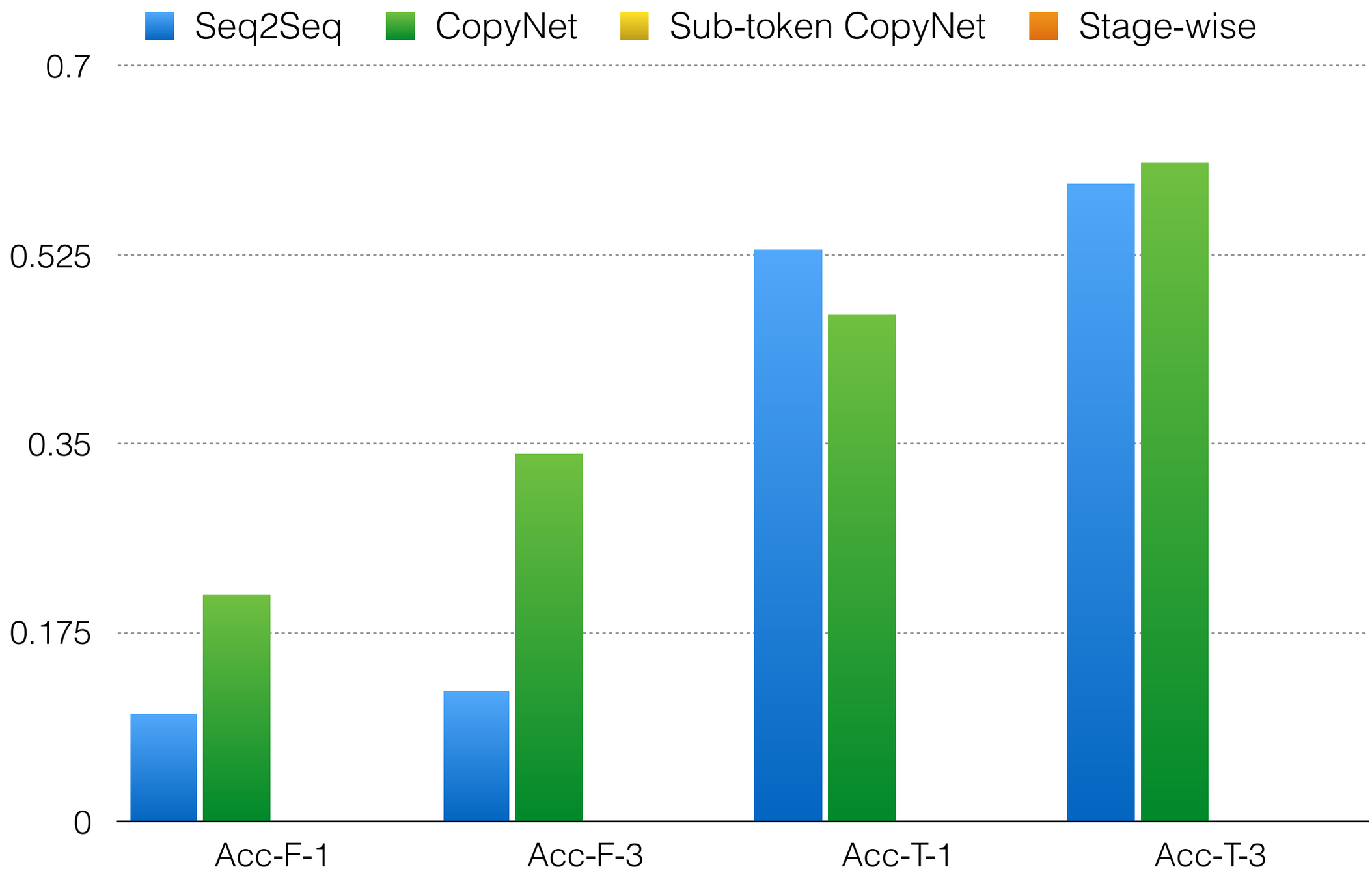
BASELINES

- Vanilla Seq2Seq (Sutskever et. al. ‘14)
- CopyNet (Gu et. al. ‘17)
- Three-stage translation model (Lin et. al. ‘17)
    1. Convert both NL and bash command to templates
    2. Apply Seq2Seq translation on the templates
    3. Fill arguments using heuristics

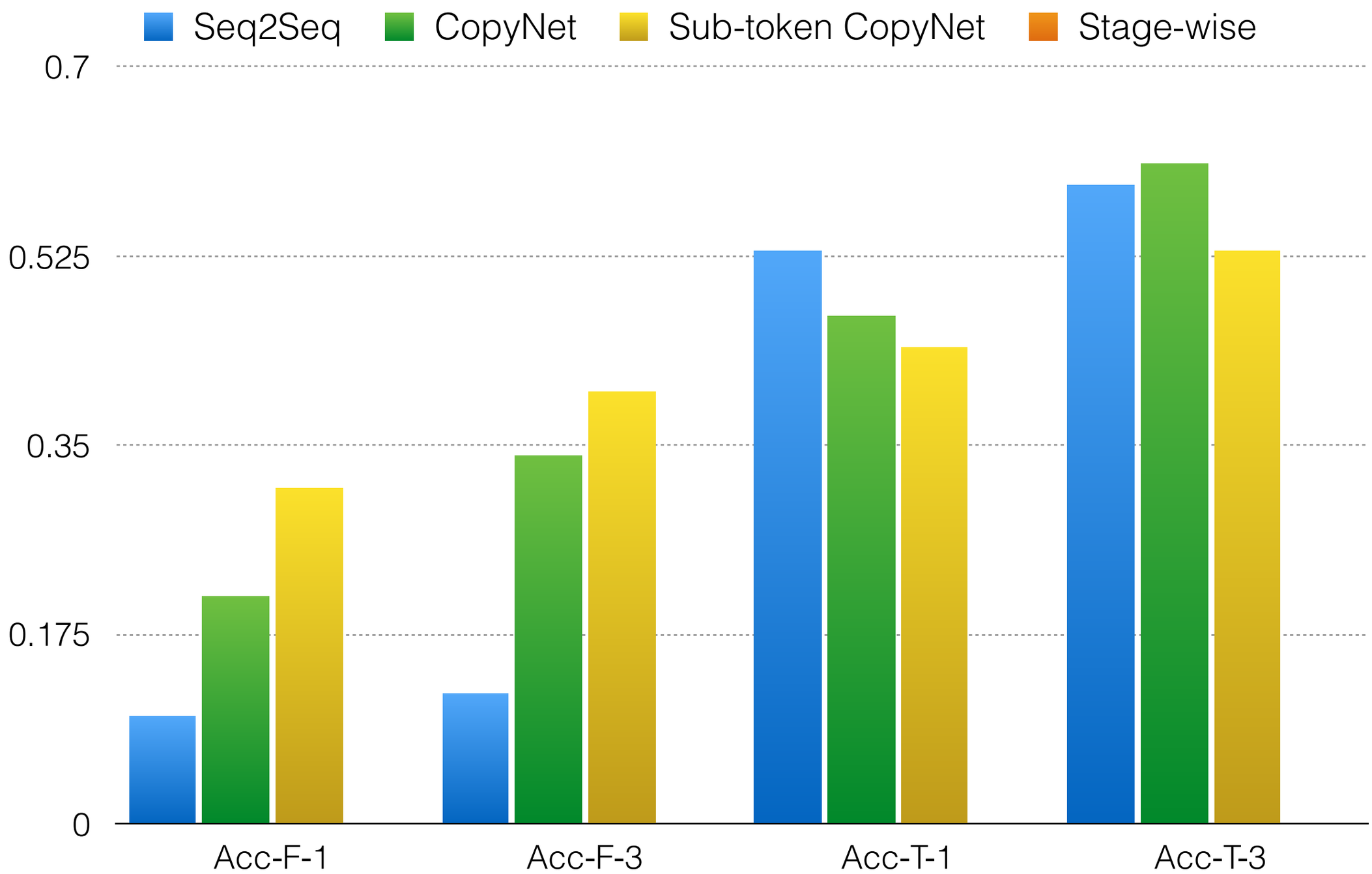
SYSTEM PERFORMANCE (Dev Set)



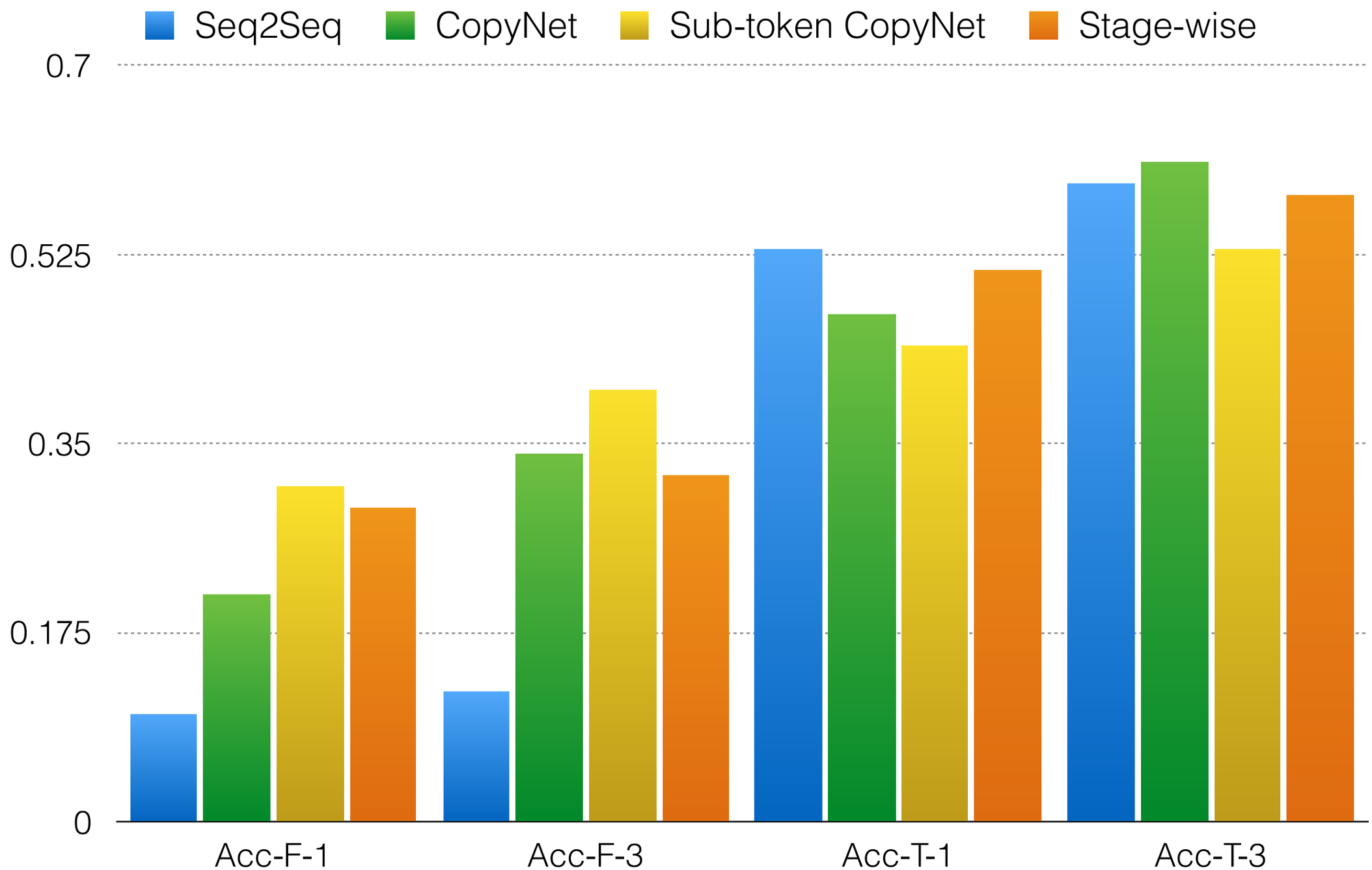
SYSTEM PERFORMANCE (Dev Set)



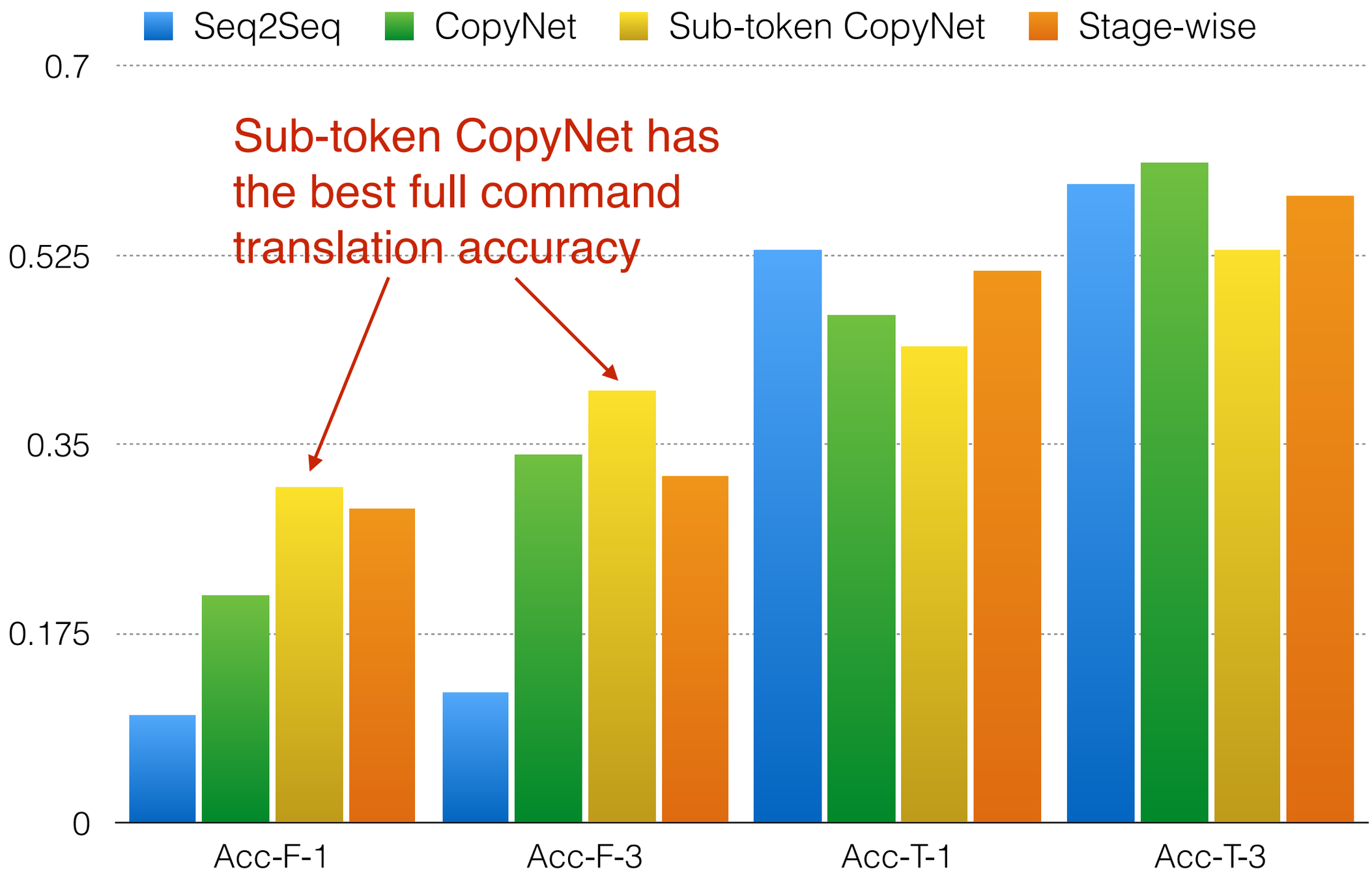
SYSTEM PERFORMANCE (Dev Set)



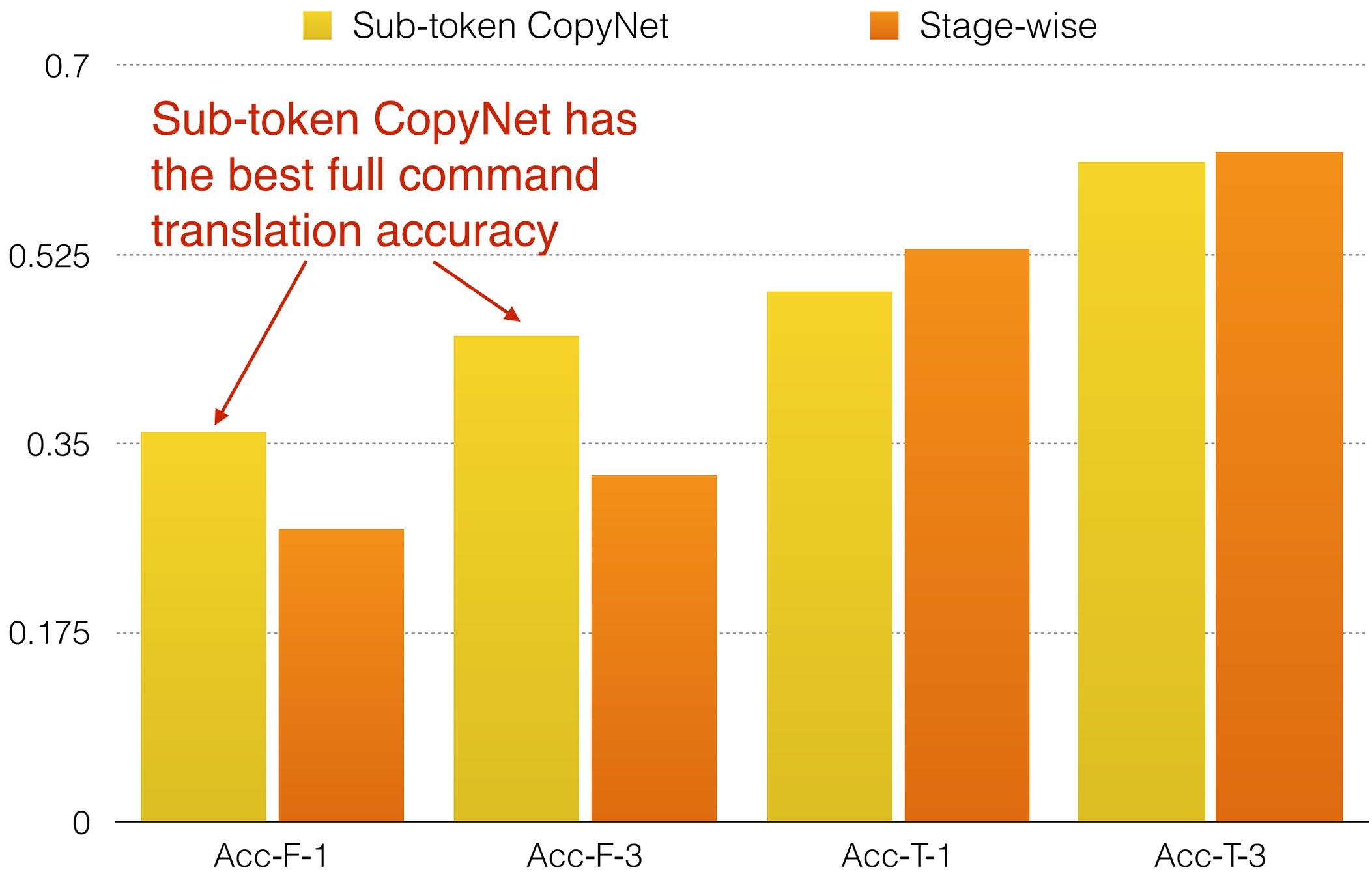
SYSTEM PERFORMANCE (Dev Set)



SYSTEM PERFORMANCE (Dev Set)



SYSTEM PERFORMANCE (Test Set)



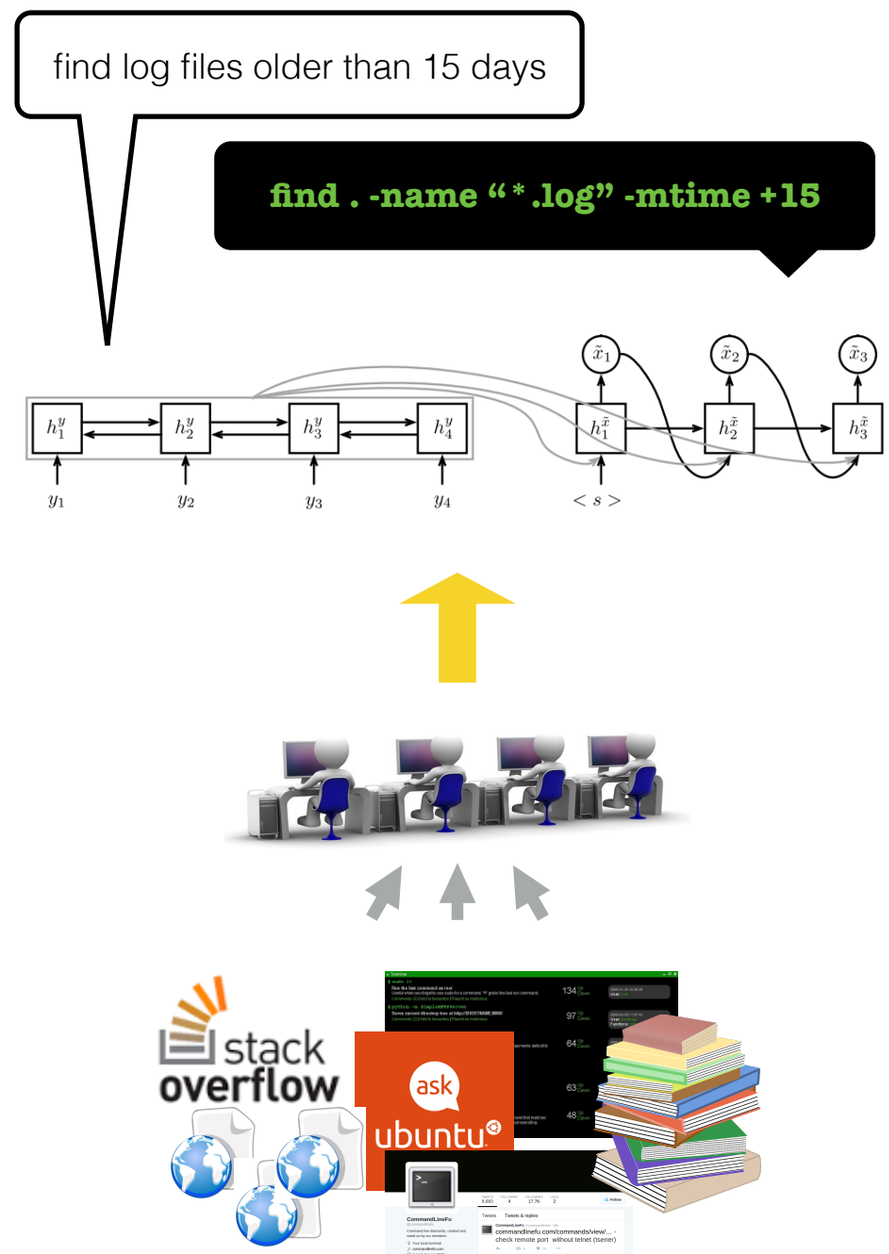
QUALITATIVE ANALYSIS

- Live Demo: http://tellina.rocks
- Split '/usr/bin/gcc' into 10 files of about equal size
- Which files in the computer were modified more than 30 days ago and larger than 500M
- Find all *company* (case-insensitive) files/directories under /basedir with null character as the delimiter

Github: https://github.com/TellinaTool

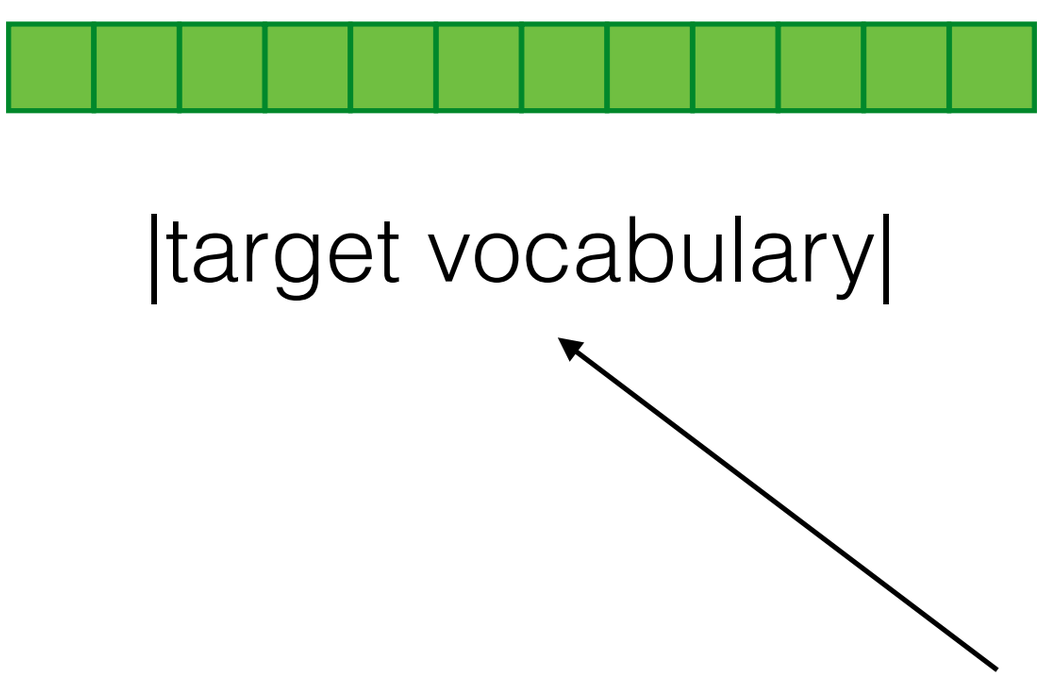
Demo: http://tellina.rocks

- ***Corpus:*** 10k real-world bash commands, paired with human-written English descriptions
- ***Data-driven baselines:*** motivated by SOTA neural machine translation approaches *copying, sub-token modeling*
- ***Huge space for improvements***
- To appear in LREC 2018 conference proceedings
- Contact: xilin@salesforce.com

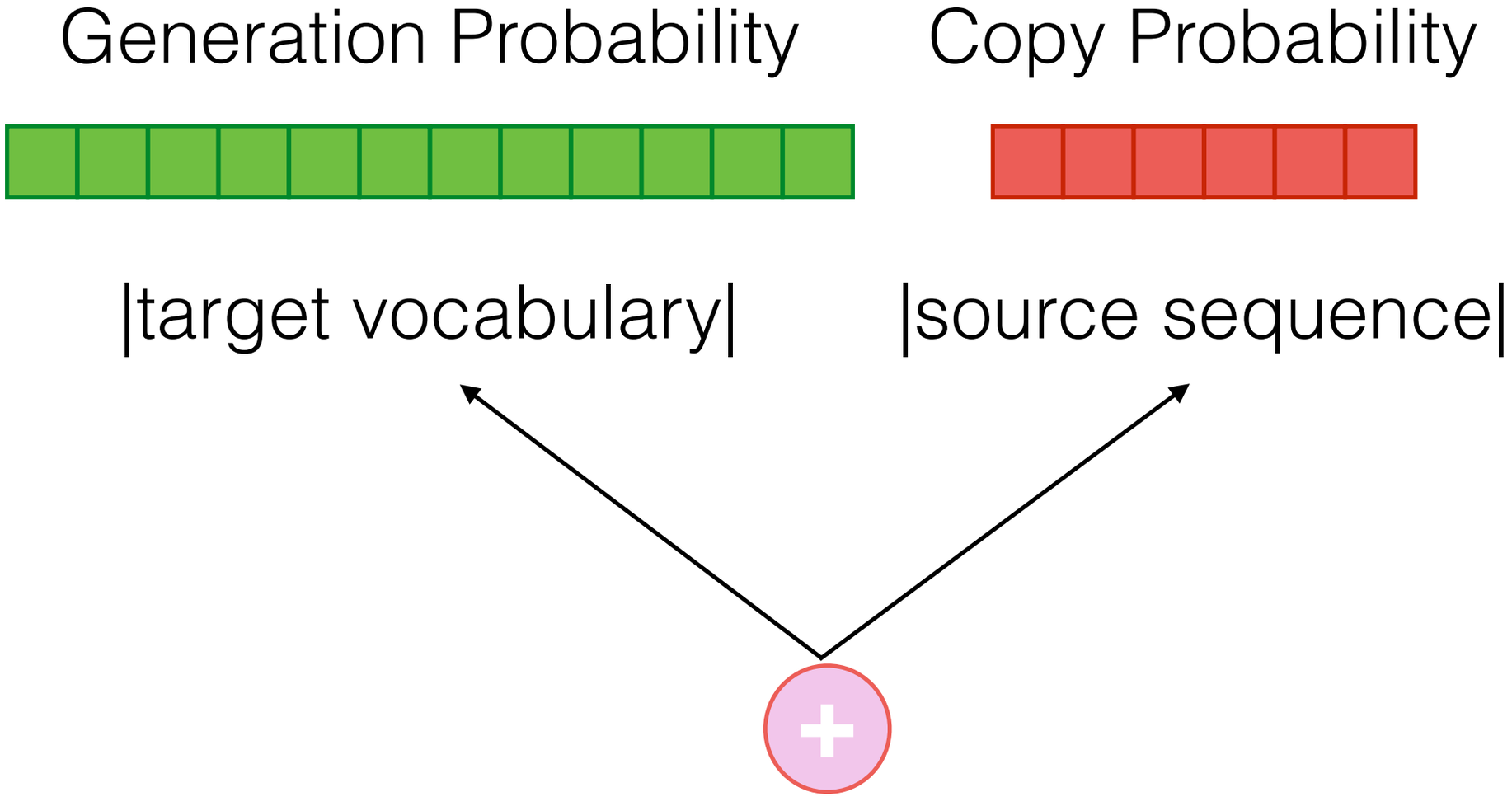


BKI - SEQ2SEQ OUTPUT PROBABILITY

Generation Probability



BKII - COPYNET OUTPUT PROBABILITY



BKIII - COPYNET (Gu et. al. 2016)

Generation Probability

|target vocabulary|

Copy Probability

|source sequence|

$$p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) = p(y_t, \mathsf{g}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) + p(y_t, \mathsf{c}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M})$$

"hidden state" "copying context"

BKIV- COPYNET (Gu et. al. 2016)

Generation Logit

Copy Logit

softmax ( )

|target vocabulary|

|source sequence|

$$p(y_t, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \bar{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & y_t \notin \mathcal{V} \cup \mathcal{X} \end{cases}$$

$$p(y_t, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:x_j=y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases}$$

BKV - SPEED-UP EXPERT SOURCING

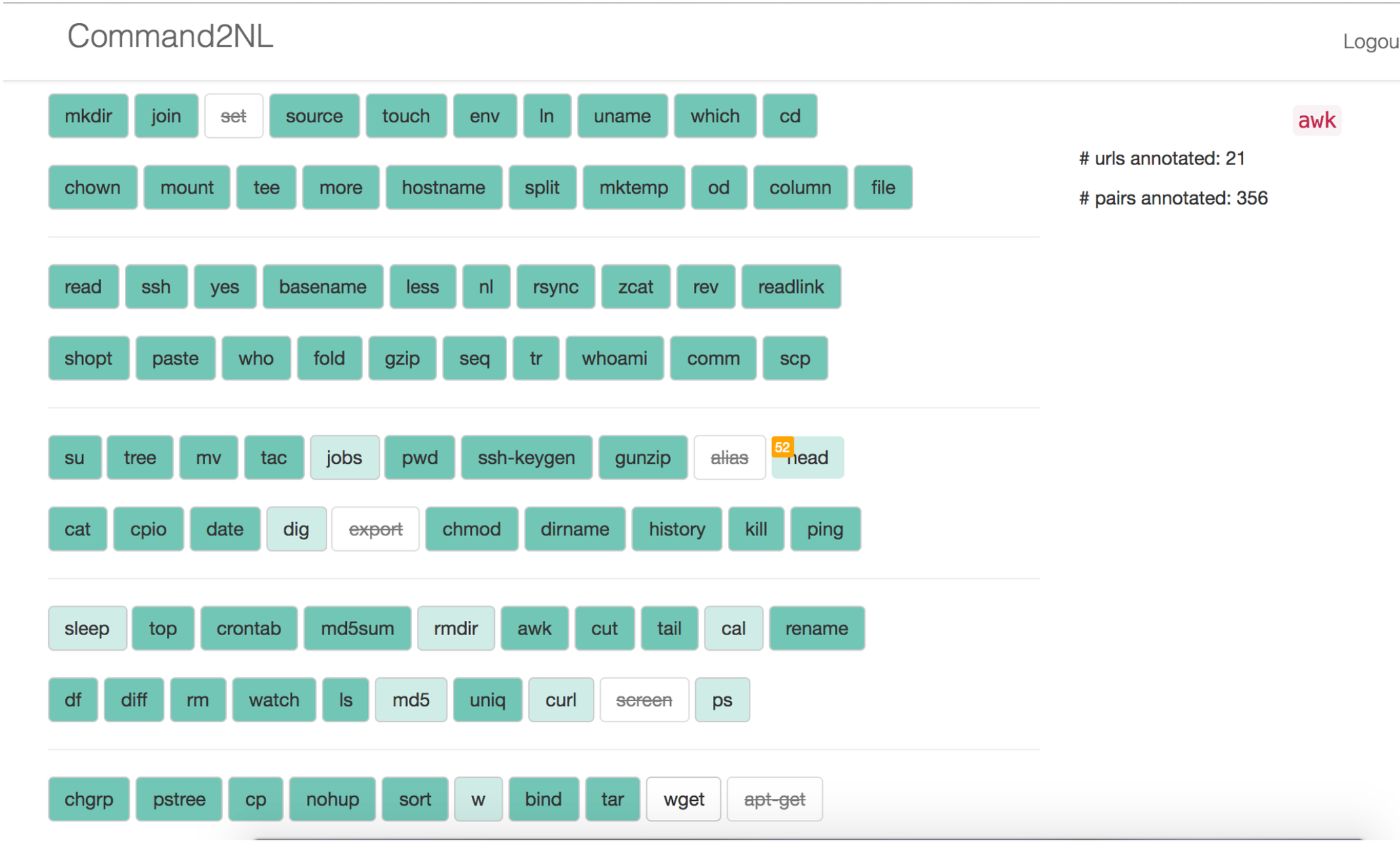


Figure 2. Data Collection Interface Screenshot

BKVI - THREE-STAGE TRANSLATION APPROACH

natural language input:
find all log files older than 15 days

Stage 1: rule-based open-vocabulary entity recognition

entity mentions: {filename: “log”, timespan: “15 days”}

natural language template:
find all [filename] files older than [timespan]

Stage 2: NL template to program template translation

Stage 3: Argument filling and post-processing

synthesized program templates:

find . -name “*.log” -mtime +15d

find . -type f -name “*.log” -mtime +15d

…