CodeBPE: Investigating Subtokenization Options for Large Language Models Pretrained on Source Code

TLDR: carefully choosing tokenization in LLMs for code is important!

Nadezhda Chirkova
Naver Labs Europe*

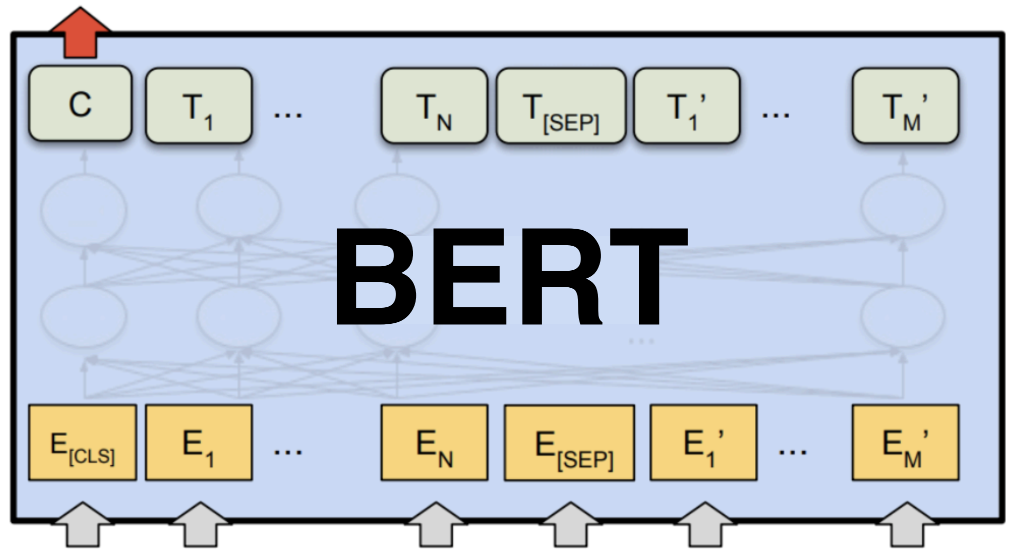
Sergey Troshin
University of Amsterdam*

* Work done at HSE University

Pretrained language models (PLMs)

GitHub

self-supervised pretraining



Final model

supervised finetuning

Task-specific data

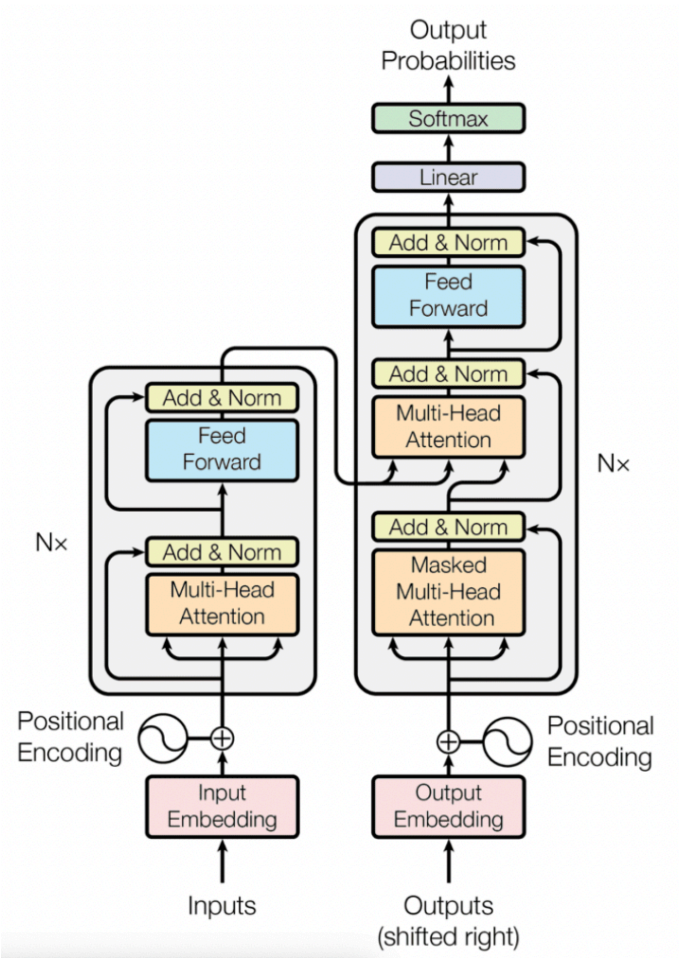
- Learn domain specifics from large code corpora during pretraining
- Often outperform models developed specifically for applied tasks

PLM pipeline

```
FreqLists = [[0, 0] for i in range(vocSz)]
```

subtoken segmentation

```
Freq List s = [ [ 0 , 0] for i in range ( vo c S z ) ]
```



pretraining

Masked Language Modeling objective (or other)

finetuning

Task-specific objective

Research on PLMs for Code

Existing work: investigation of models and pretraining objectives:

- CodeBERT (Feng20), CuBERT (Kanade20)
- CodeGPT (Lu21)
- PLBART (Ahmad21)
- GraphCodeBERT (Guo21) with data flow prediction objective
- CodeT5 (Wang21b) with variable naming and identifier tagging objectives
- DOBF (Roziere21) with variable naming objective

Our work considers another dimension: *subtokenization options*

(e. g. BPE vocabulary size or BPE vs UnigramLM)

Overview

Main goals:

- choose the most effective subtokenization (maximize downstream performance)
- choose the most length-efficient subtokenization without downstream performance drop

Considered options:

- Subtokenization granularity
- UnigramLM vs BPE
- Vocabulary size
- Transferability between programming languages

Downstream tasks:

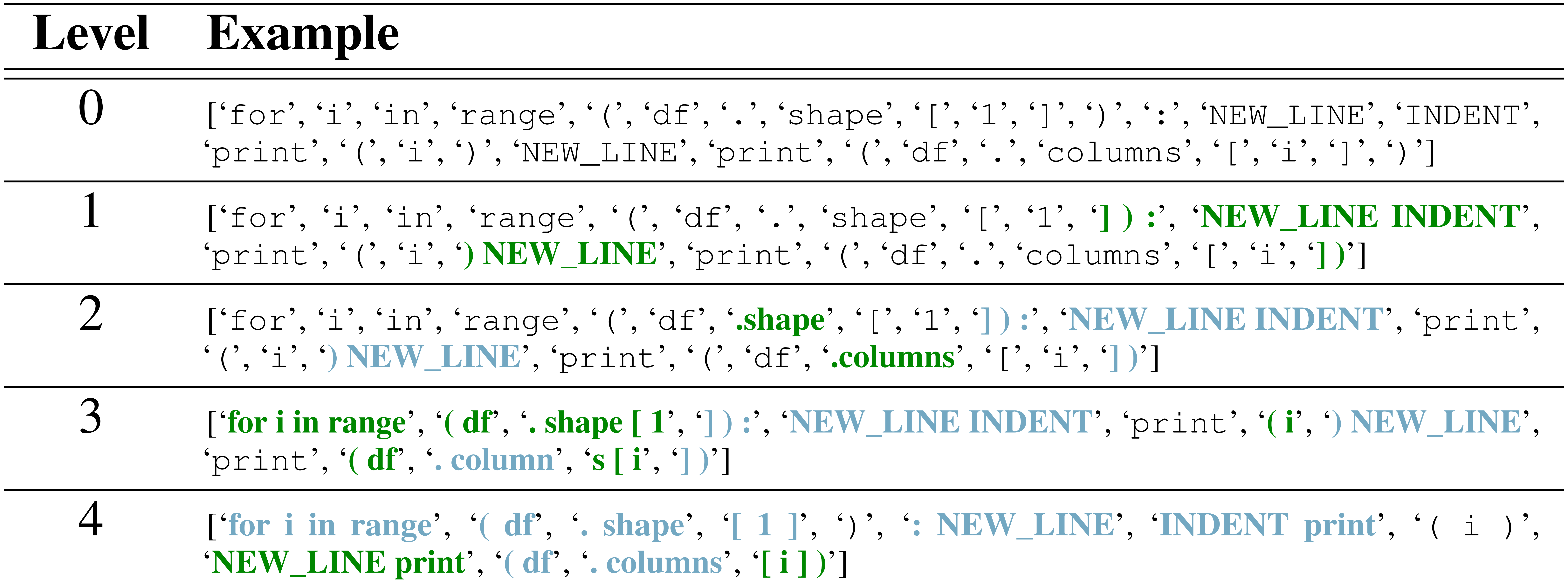
- Code translation
- Code summarization
- Code generation
- Clone detection

Methodology: start from UnigramLM (50k vocab) and add one modification at a time

All experiments with PLBART

Subtokenization granularity

Various levels of including spaces and punctuation in tokens:



Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’]
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘**])**’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘**(i**’, ‘**) NEW_LINE**’, ‘print’, ‘**(df**’, ‘**. column**’, ‘**s [i**’, ‘**])**’]
4	[‘**for i in range**’, ‘**(df**’, ‘**. shape**’, ‘**[1]**’, ‘)’, ‘**: NEW_LINE**’, ‘**INDENT print**’, ‘(i)’, ‘**NEW_LINE print**’, ‘**(df**’, ‘**. columns**’, ‘**[i])**’]

Subtokenization granularity

Various levels of including spaces and punctuation in tokens:

conventionally used

Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’]
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘**])**’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘**(i**’, ‘**) NEW_LINE**’, ‘print’, ‘**(df**’, ‘**. column**’, ‘**s [i**’, ‘**])**’]
4	[‘**for i in range**’, ‘**(df**’, ‘**. shape**’, ‘**[1]**’, ‘)’, ‘**: NEW_LINE**’, ‘**INDENT print**’, ‘(i)’, ‘**NEW_LINE print**’, ‘**(df**’, ‘**. columns**’, ‘**[i])**’]

Subtokenization granularity

Various levels of including spaces and punctuation in tokens:

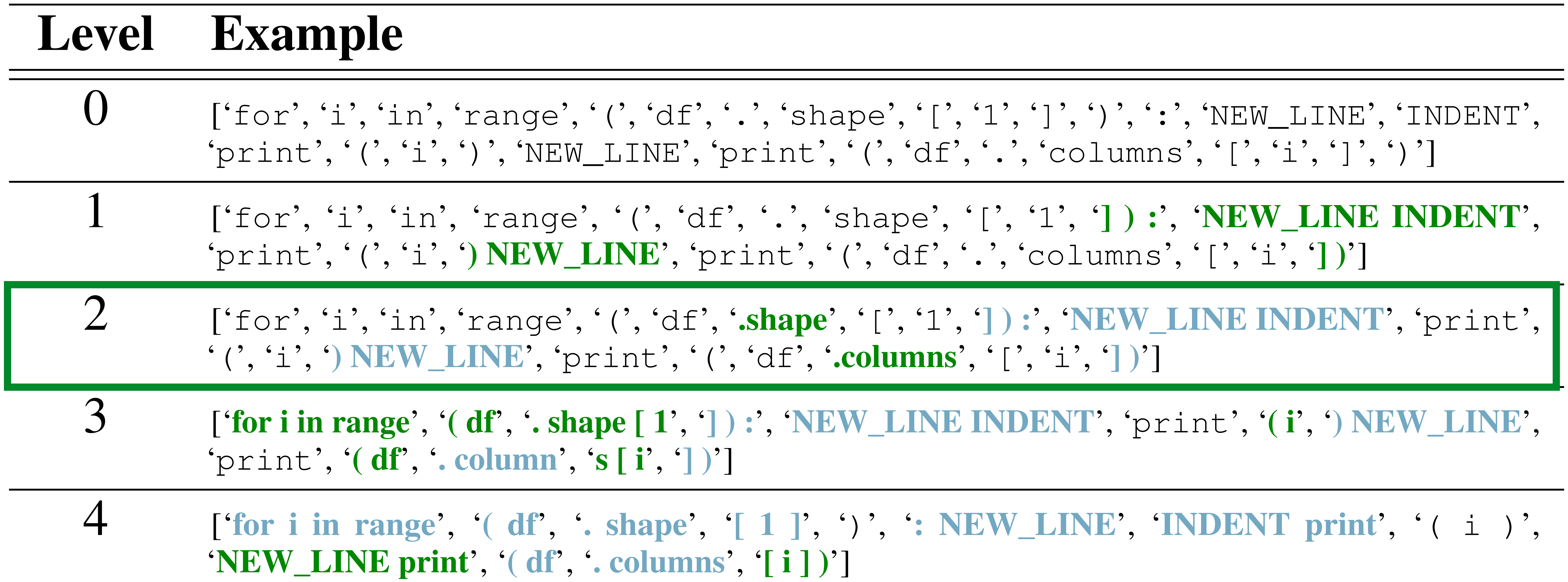
allow merging of punctuation chars

Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’] ←
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘**])**’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘**(i**’, ‘**) NEW_LINE**’, ‘print’, ‘**(df**’, ‘**. column**’, ‘**s [i**’, ‘**])**’]
4	[‘**for i in range**’, ‘**(df**’, ‘**. shape**’, ‘**[1]**’, ‘)’, ‘**: NEW_LINE**’, ‘**INDENT print**’, ‘(i)’, ‘**NEW_LINE print**’, ‘**(df**’, ‘**. columns**’, ‘**[i])**’]

Subtokenization granularity

Various levels of including spaces and punctuation in tokens:

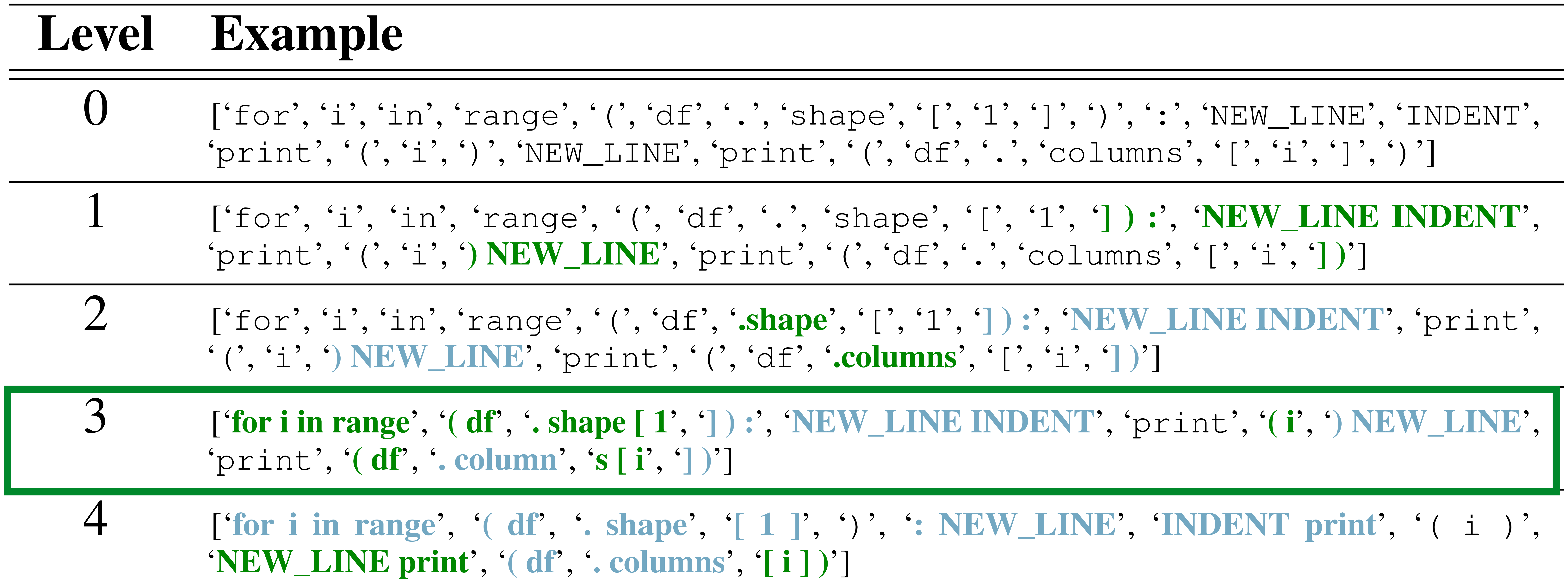
+ allow merging of dots with text



Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’]
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘**])**’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘**(i**’, ‘**) NEW_LINE**’, ‘print’, ‘**(df**’, ‘**. column**’, ‘**s [i**’, ‘**])**’]
4	[‘**for i in range**’, ‘**(df**’, ‘**. shape**’, ‘**[1]**’, ‘)’, ‘**: NEW_LINE**’, ‘**INDENT print**’, ‘(i)’, ‘**NEW_LINE print**’, ‘**(df**’, ‘**. columns**’, ‘**[i])**’]

Subtokenization granularity

Various levels of including spaces and punctuation in tokens:

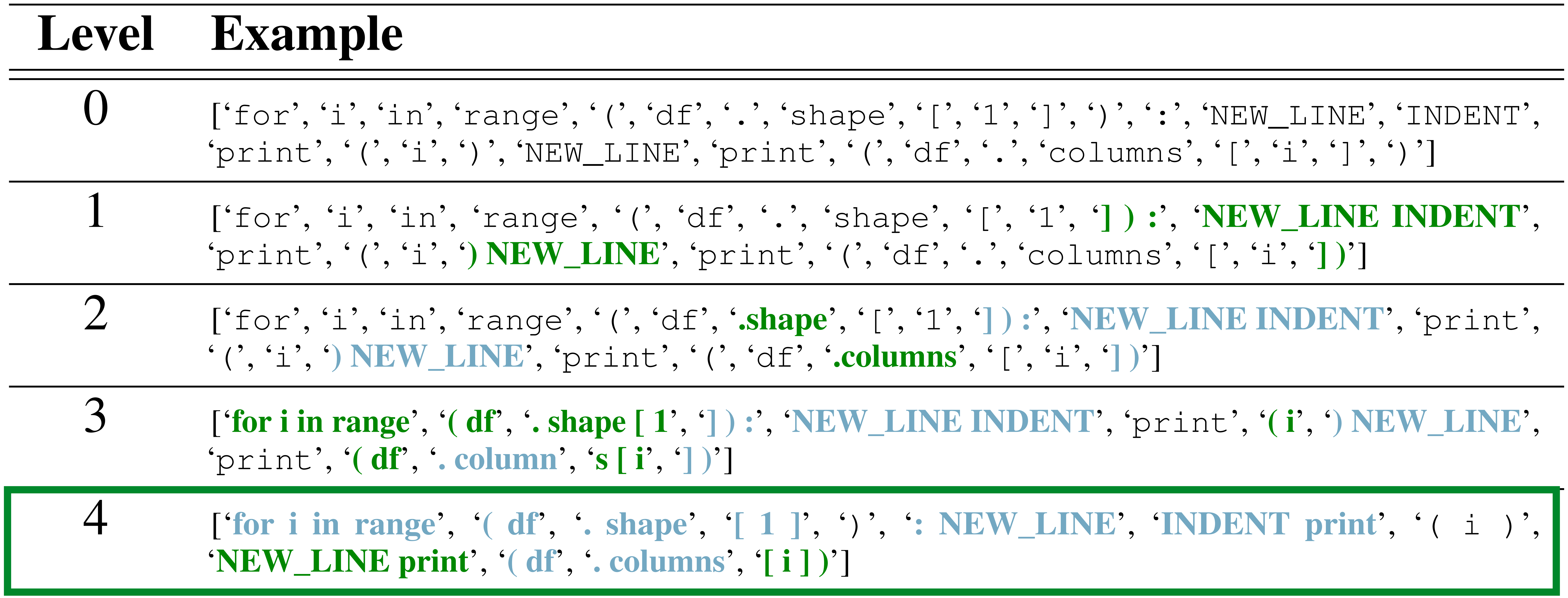


Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’]
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘]) :’, ‘NEW_LINE INDENT’, ‘print’, ‘(’, ‘i’, ‘) NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘])’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘]) :’, ‘NEW_LINE INDENT’, ‘print’, ‘**(i**’, ‘) NEW_LINE’, ‘print’, ‘**(df**’, ‘. column’, ‘**s [i**’, ‘])’]
4	[‘for i in range’, ‘(df’, ‘. shape’, ‘[1]’, ‘)’, ‘: NEW_LINE’, ‘INDENT print’, ‘(i)’, ‘**NEW_LINE print**’, ‘(df’, ‘. columns’, ‘**[i])**’]

allow spaces inside tokens

Subtokenization granularity

Various levels of including spaces and punctuation in tokens:



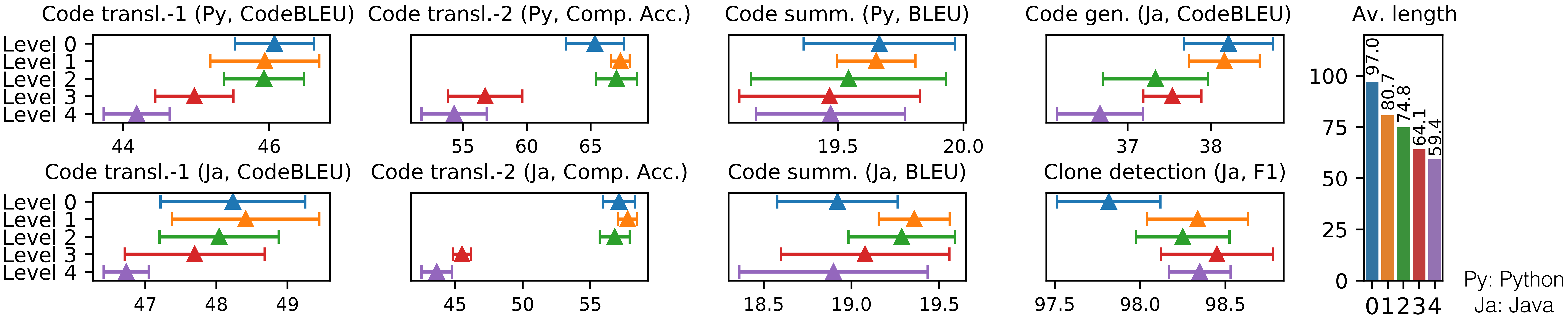
Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’]
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘**])**’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘**(i**’, ‘**) NEW_LINE**’, ‘print’, ‘**(df**’, ‘**. column**’, ‘**s [i**’, ‘**])**’]
4	[‘**for i in range**’, ‘**(df**’, ‘**. shape**’, ‘**[1]**’, ‘)’, ‘**: NEW_LINE**’, ‘**INDENT print**’, ‘(i)’, ‘**NEW_LINE print**’, ‘**(df**’, ‘**. columns**’, ‘**[i])**’]

allow new lines and ; inside tokens

Subtokenization granularity

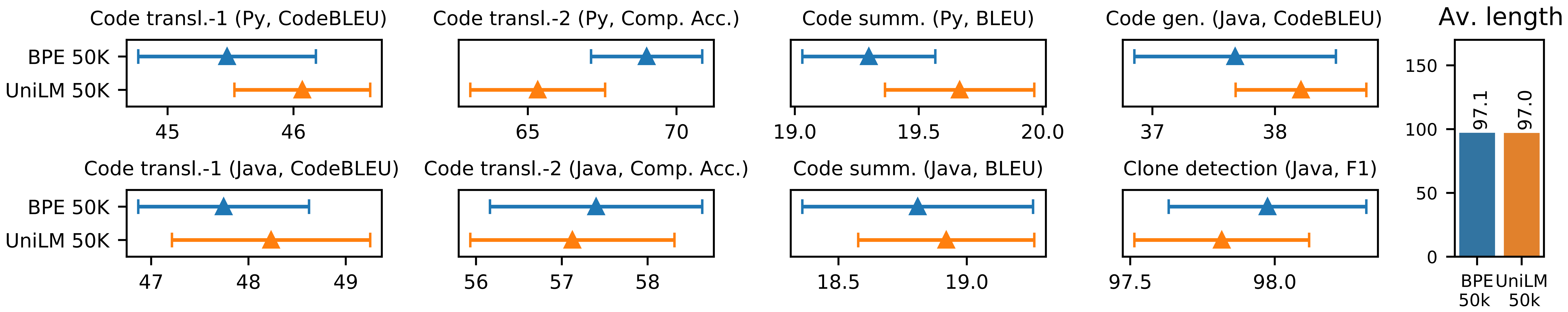
Various levels of including spaces and punctuation in tokens:

Level	Example
0	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘]’, ‘)’, ‘:’, ‘NEW_LINE’, ‘INDENT’, ‘print’, ‘(’, ‘i’, ‘)’, ‘NEW_LINE’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘]’, ‘)’]
1	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘.’, ‘shape’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘.’, ‘columns’, ‘[’, ‘i’, ‘**])**’]
2	[‘for’, ‘i’, ‘in’, ‘range’, ‘(’, ‘df’, ‘**.shape**’, ‘[’, ‘1’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘(’, ‘i’, ‘**) NEW_LINE**’, ‘print’, ‘(’, ‘df’, ‘**.columns**’, ‘[’, ‘i’, ‘**])**’]
3	[‘**for i in range**’, ‘**(df**’, ‘**. shape [1**’, ‘**]) :**’, ‘**NEW_LINE INDENT**’, ‘print’, ‘**(i**’, ‘**) NEW_LINE**’, ‘print’, ‘**(df**’, ‘**. column**’, ‘**s [i**’, ‘**])**’]
4	[‘**for i in range**’, ‘**(df**’, ‘**. shape**’, ‘**[1]**’, ‘)’, ‘**: NEW_LINE**’, ‘**INDENT print**’, ‘(i)’, ‘**NEW_LINE print**’, ‘**(df**’, ‘**. columns**’, ‘**[i])**’]



*Main conclusion: **Level 1** compresses lengths by 17% without performance drop, comp. to **Level 0***

Subtokenization algorithm: BPE vs UnigramLM



*Main conclusion: **UnigramLM** slightly outperforms or performs on par with **BPE** in 7 tasks*

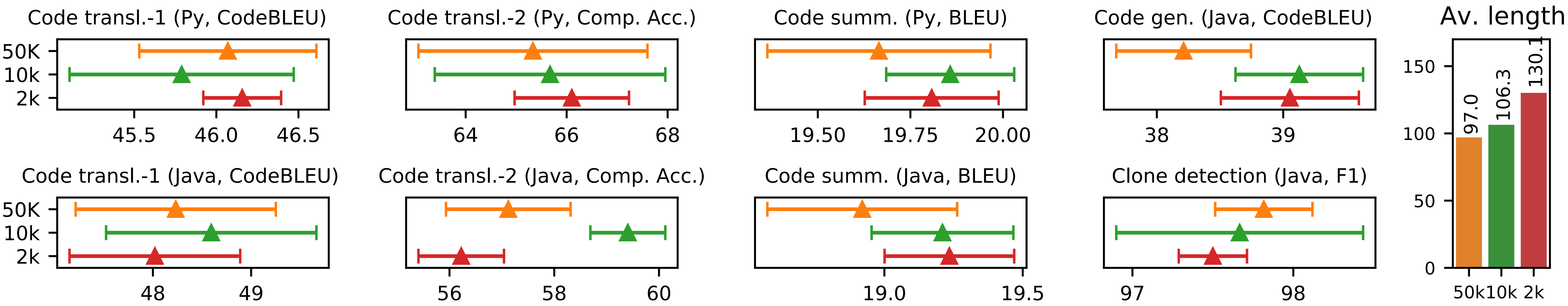
Subtokenization algorithm: BPE vs UnigramLM

UnigramLM is better aligned with splitting identifiers by CamelCase and snake_case:

Original token	UnigramLM subtokenization	BPE subtokenization	Native subtokenization (Camel- or snake_case)
fromDottedString	[’from’, ’Dotted’, ’String’]	[’from’, ’**Dot**’, ’**ted**’, ’String’]	[’from’, ’Dotted’, ’String’]
isInstantiated	[’is’, ’Instantiate’, ’d’]	[’**isIn**’, ’**stanti**’, ’**ated**’]	[’is’, ’Instantiated’]
GridBagConverter	[’Grid’, ’Bag’, ’Converter’]	[’**GridBag**’, ’Converter’]	[’Grid’, ’Bag’, ’Converter’]
isSameSizeHorizontally	[’isSame’, ’Size’, ’Horizontally’]	[’isSame’, ’Size’, ’**H**’, ’orizontally’]	[’is’, ’Same’, ’Size’, ’Horizontally’]
PA_Hierarchy_ID	[‘PA’, ‘_’, ‘Hierarchy’, ‘_ID’]	[‘PA’, ‘**_H**’, ‘ierarchy’, ‘_ID’]	[‘PA’, ‘_’, ‘Hierarchy’, ‘_’, ‘ID’]

Vocabulary size

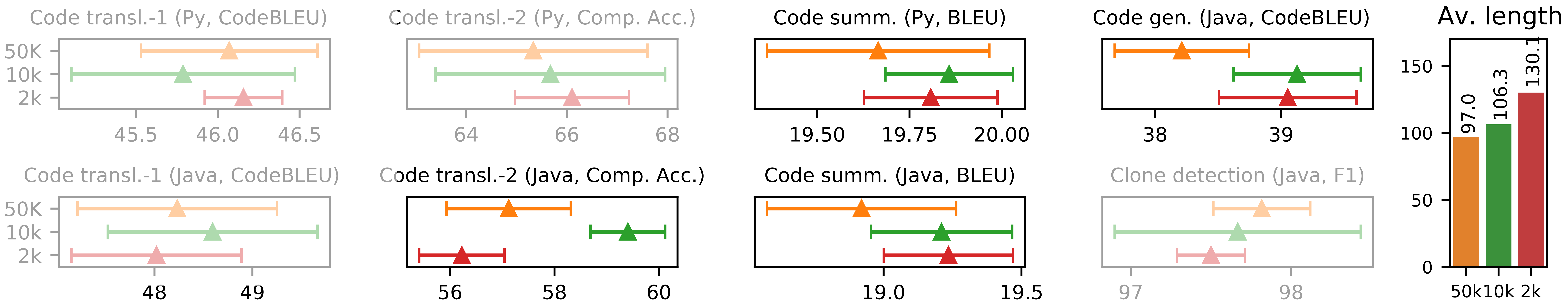
Unigram LM: **50k**, **10k**, **2k**



*Main conclusion: **10k** > **50k** in 4 tasks, in other tasks similar performance*

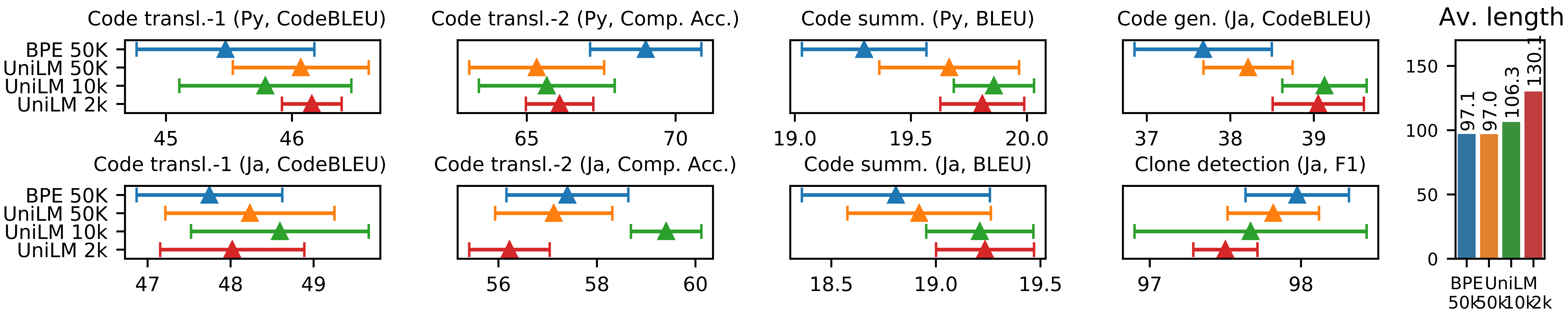
Vocabulary size

Unigram LM: **50k**, **10k**, **2k**



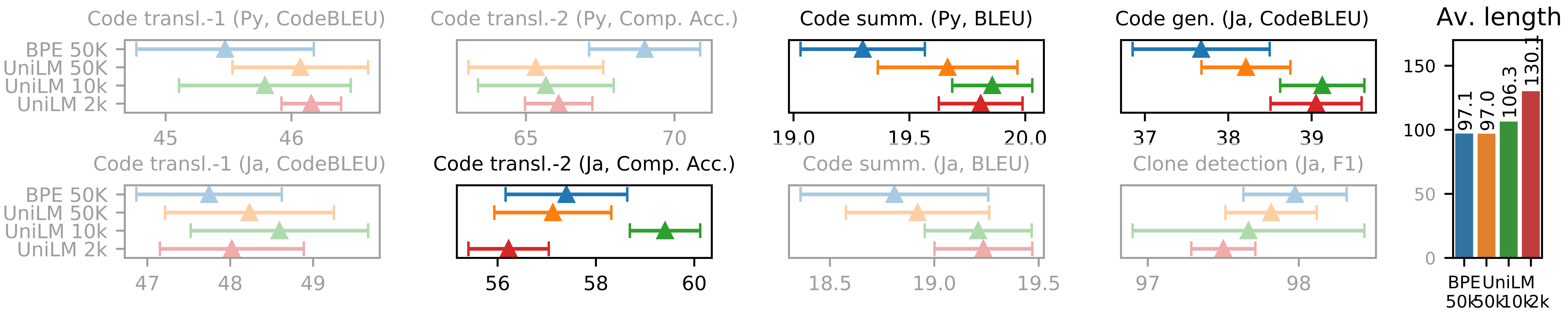
*Main conclusion: **10k** > **50k** in 4 tasks, in other tasks similar performance*

Subtokenization algorithm + vocabulary size



__UnigramLM 10k__ > commonly used __BPE 50k__ in 3 tasks substantially and 2 tasks by one std

Subtokenization algorithm + vocabulary size



UnigramLM 10k *> commonly used* ***BPE 50k*** *in 3 tasks significantly and 2 tasks by one std*

Summary

BPE-50k

Commonly used

```
F req Lists = [ [ 0 , 0 ] for i in range ( voc Sz ) ]
```

UnigramLM-10k

+0.5-2% quality
(3-19% length increase)

```
Freq List s = [ [ 0 , 0 ] for i in range ( vo c S z ) ]
```

Grouping punctuation

17% length reduction
without quality drop

```
Freq Lists =[[ 0 , 0 ] for i in range ( voc S z )]
```

CodeBPE / CodeUnigramLM