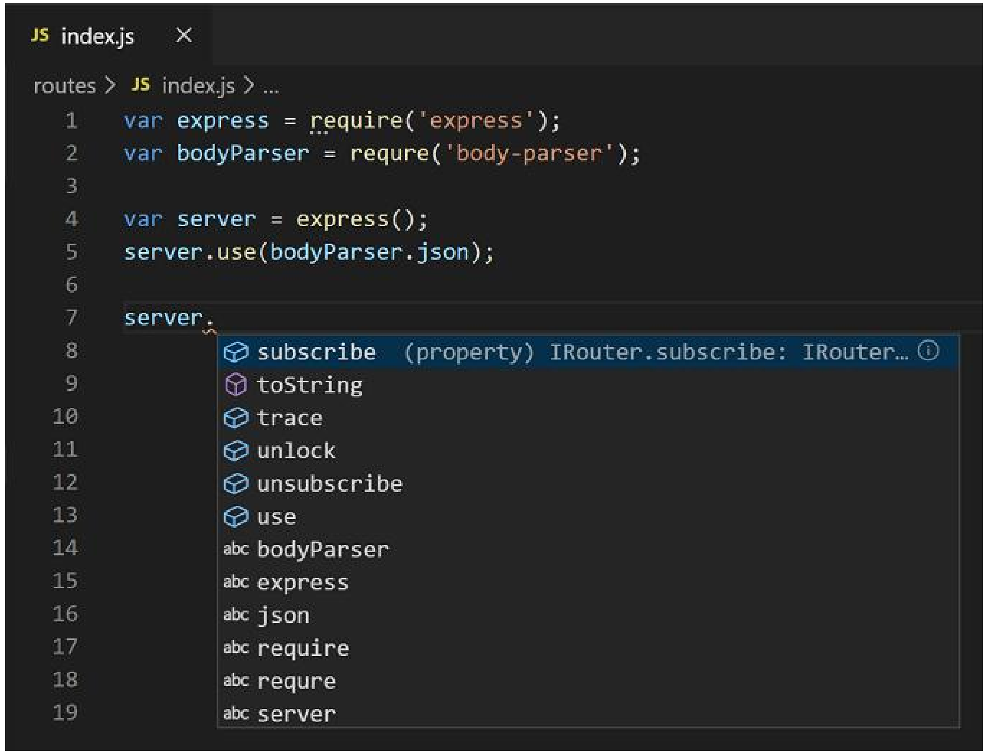
Represent Code as Action Sequence for Predicting Next Method Call

Yu Jiang, Liang Wang, Hao Hu, Xianping Tao
State Key Laboratory For Novel Software Technology,
Nanjing University

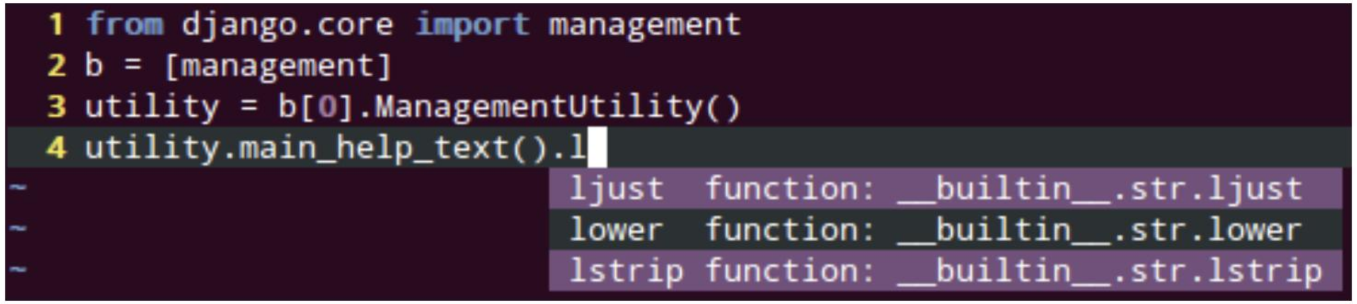
Nanjing University

Research Problem

- Code Completion
  - One of the most frequent used functions in modern IDEs
  - Imporve developers’ coding efficiency



code completion in VS Code



code completion in Vim

Code Completion

- Traditional approach
  - often static analysis
- Machine learning approach
  - naturalness[1]
    - repeated elements in natural language
  - localness[2]
    - like cache, near elements are more likely to appear
  - Big Code
    - open source code repositories
    - provide training data for machine learning

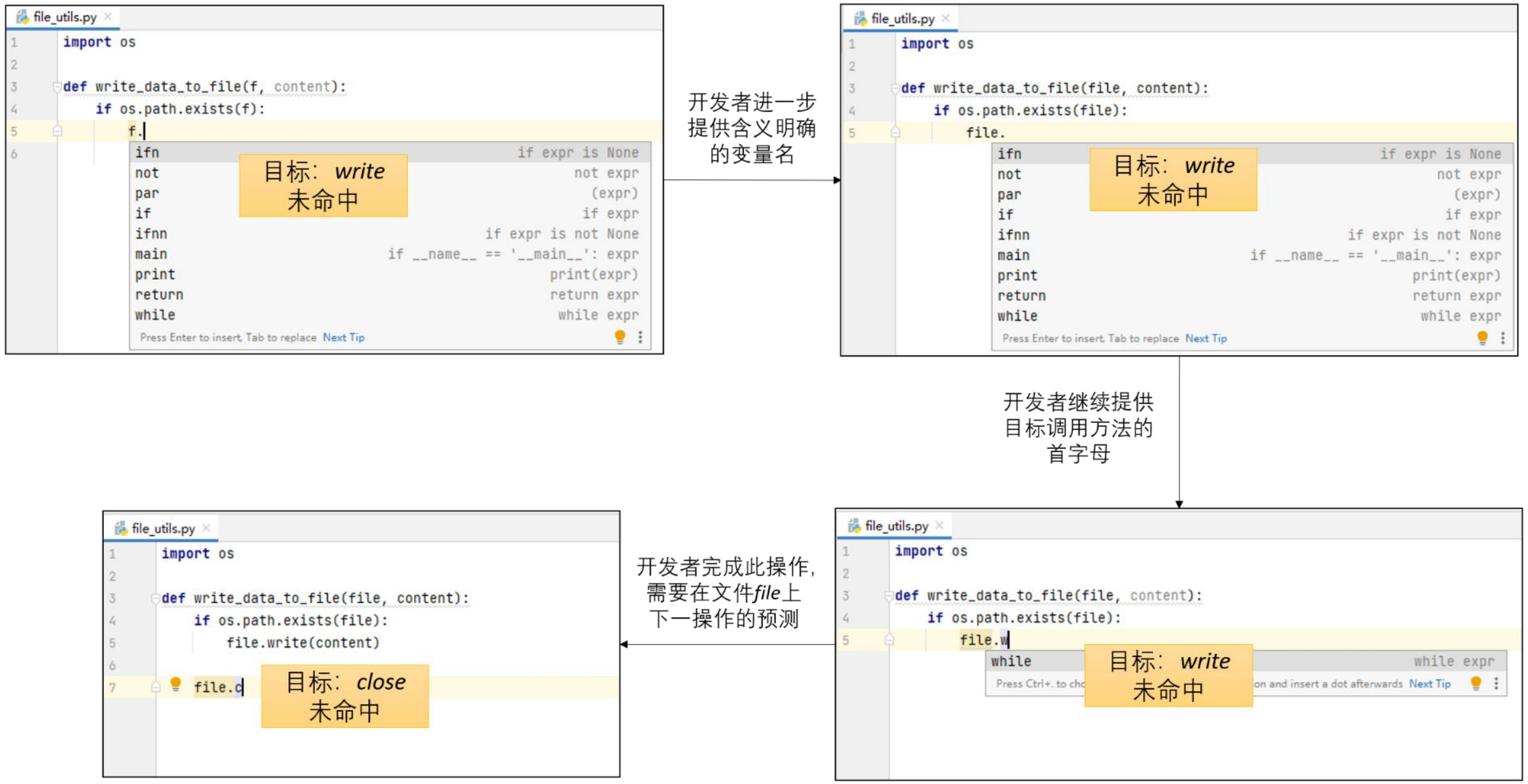
```
int func() {
    int count_1 = 0;
    for(int i = 0;i < 10;i++) {
        if (i % 2 == 0) {
            continue;
        } else {
            count_1 += 1;
        }
    }

    int count_2 = 0;
    for(int i = 0;i < 10;i++) {
        if (i % 3 == 0) {
            continue;
        } else {
            count_2 += 1;
        }
    }
}
```

[1]A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the naturalness of software," in 2012 34th International Conference on Software Engineering (ICSE), Jun. 2012, pp. 837–847. doi: 10.1109/ICSE.2012.6227135.

[2]Z. Tu, Z. Su, and P. Devanbu, "On the localness of software," in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, NY, USA, Nov. 2014, pp. 269–280. doi: 10.1145/2635868.2635875.

Motivation -- Improvement

- Recent papers have pointed out some performance differences of code completion application in real world[1][2]
- PyCharm code completion failure in real word
  - developer has provided with meaningful information but the code completion system could not give the proper choices(the developer gives simple and direct description of method)

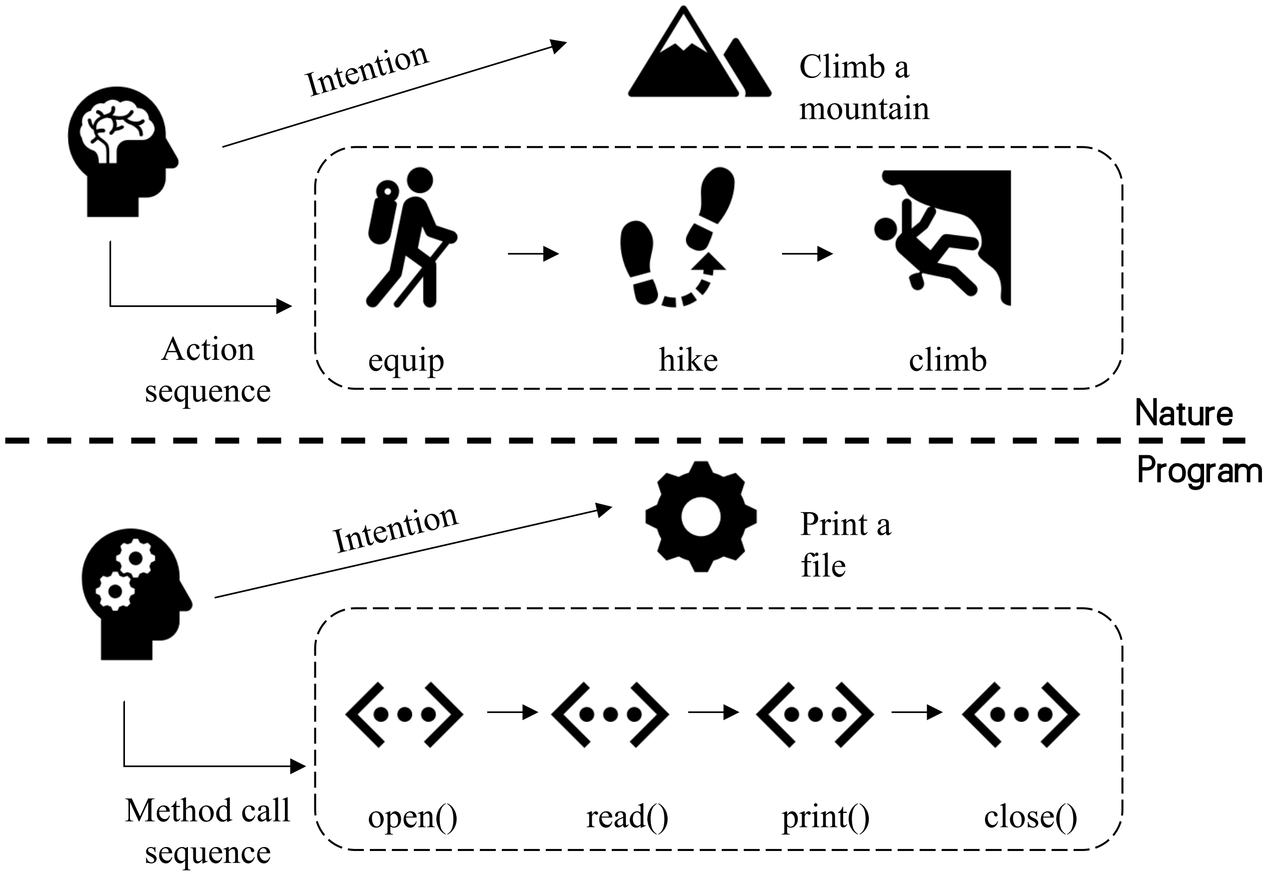


[1]V. J. Hellendoorn, S. Proksch, H. C. Gall, and A. Bacchelli, "When Code Completion Fails: A Case Study on Real-World Completions," in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019, pp. 960 - 970. doi: 10.1109/ICSE.2019.00101.

[2]杨博，张能，李善平，等. 智能代码补全研究综述[J]. Journal of Software，2020,31(5): 1435-1453.

Inspiration: Action Sequence

- Similarity of action sequence in program and nature
- Algorithms can be seen as a sequence of actions
- Developers think about action sequences in their mind and then write the code
- Predict the next action to complete code



Action Sequence -- Natural Language Semantics

- According to our statistics on 440,000+ Python source code files:
  - More than 60% first words of method names are verb
  - Compare words in method names to words in Google News, over 90% of the words are the same
  - Developers use same words with different interpretation in code because of the different contexts
- Natural language semantics could be extracted from code to describe the action sequence

	Google News	Code(method names)	Intersection	Ratio
Original	3,000,000	19,367	15,504	80.05%
Case insensitive	2,702,150	15,281	13,126	85.90%
Case insensitive & alpha only	706,977	14,908	13,085	87.77%

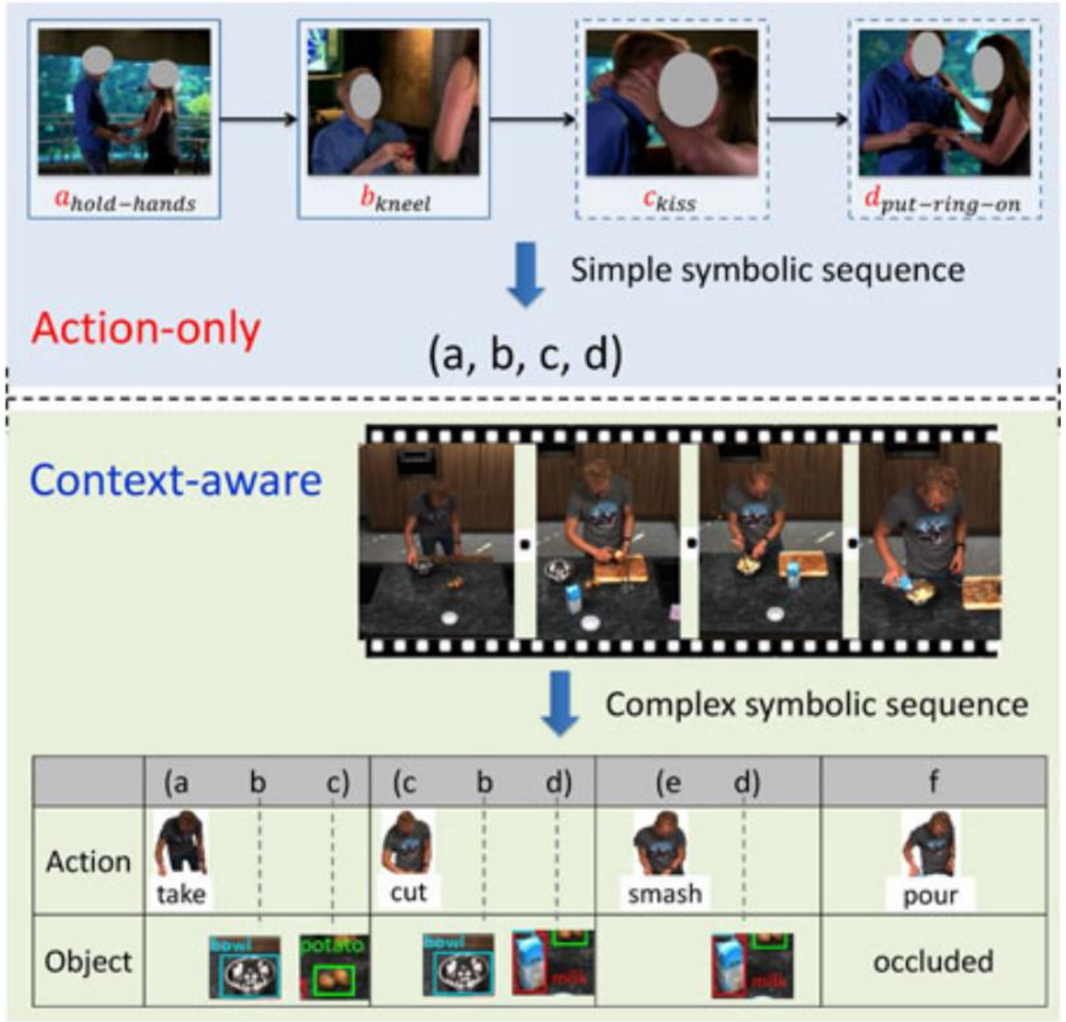
delete a server
delete an element
delete a file
delete a node
delete a task
...

delete	push	run	linux
detach	subscribe	execute	android
detail	rollback	runs	osx
remove	publish	launch	windows
unlink	pull	runner	ubuntu
cancel	consume	executor	desktop
deleted	pushed	runs	osx
edit	nudge	drive	unix
overwrite	propel	ran	Windoze
untag	move	go	KDE4
uninstall	pull	walk	filesystems

Action

- Type: Assign, Return, Call (from corresponding statements)
- Context
  - code editing context
  - related to project, dependency, function and coding style
- Actor
  - the source of the action
  - like an object and a class(`str.index`)
- Call/Params(only for calls)

$$action = (context, actor, call, parameters)$$



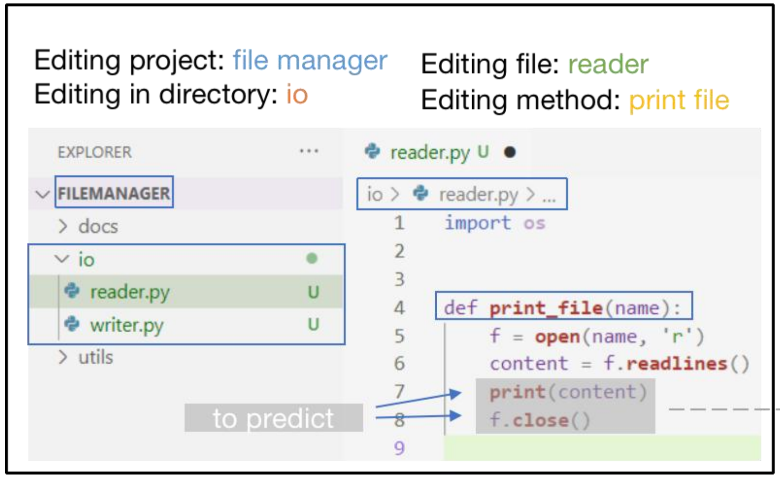
action modeling approaches in action recognition related work[1]

[1] LI K, FU Y. Prediction of human activity by discovering temporal sequence pat_x0002_terns[J]. IEEE transactions on pattern analysis and machine intelligence, 2014, 36(8): 1644—1657.

Action Sequence Representation

- Examples
  - abstraction process
  - real world example

Developer code editing environment



model

actions to predict

Action sequence

action	context		actor	call	params
1	-	-	-	-	-
2	file manager, io, reader, print file	<Assign>	f	-	-
3		<Call>	<Global>	open	name, ‘r’
4		<Assign>	content	-	-
5		<Call>	f	readlines	-
6		?	?	?	?
7		?	?	?	?

```python
def save_data(output_file):
    # read and save data to a given file
    database = db_connect('raw_db')
    target_rows =
          database.execute(QUERY_SQL_CMD).get()
    database.disconnect()

    target_file = open(output_file)
    target_file.write(target_rows)
    target_file.close()
```

a method in db_utils.py Python source file

```
db utils save data <Assign> database,
db utils save data <Call> $ - db connect @ $Const$,
db utils save data <Assign> target rows,
db utils save data <Call> database - execute @ QUERY SQL CMD,
db utils save data <Call> database execute - get @ ,
db utils save data <Call> database - disconnect @ ,
db utils save data <Assign> target file,
db utils save data <Call> $ - open @ output file,
db utils save data <Call> target file - write @ target rows,
db utils save data <Call> target file - close @ ,
```

action sequence built from the method save_data

Language Modeling

- Sequence

$$S_n = (w_1, w_2, ..., w_n)$$

- Language Model(N-gram)

$$P(w_1, ..., w_m) \approx \prod_{i=1}^{m} P(w_i | w_{i-n+1}, ..., w_{i-1})$$

- Causal language modeling loss functuin

$$L = -\sum_{i=2}^{n} log P(w_i | S_{i-1})$$

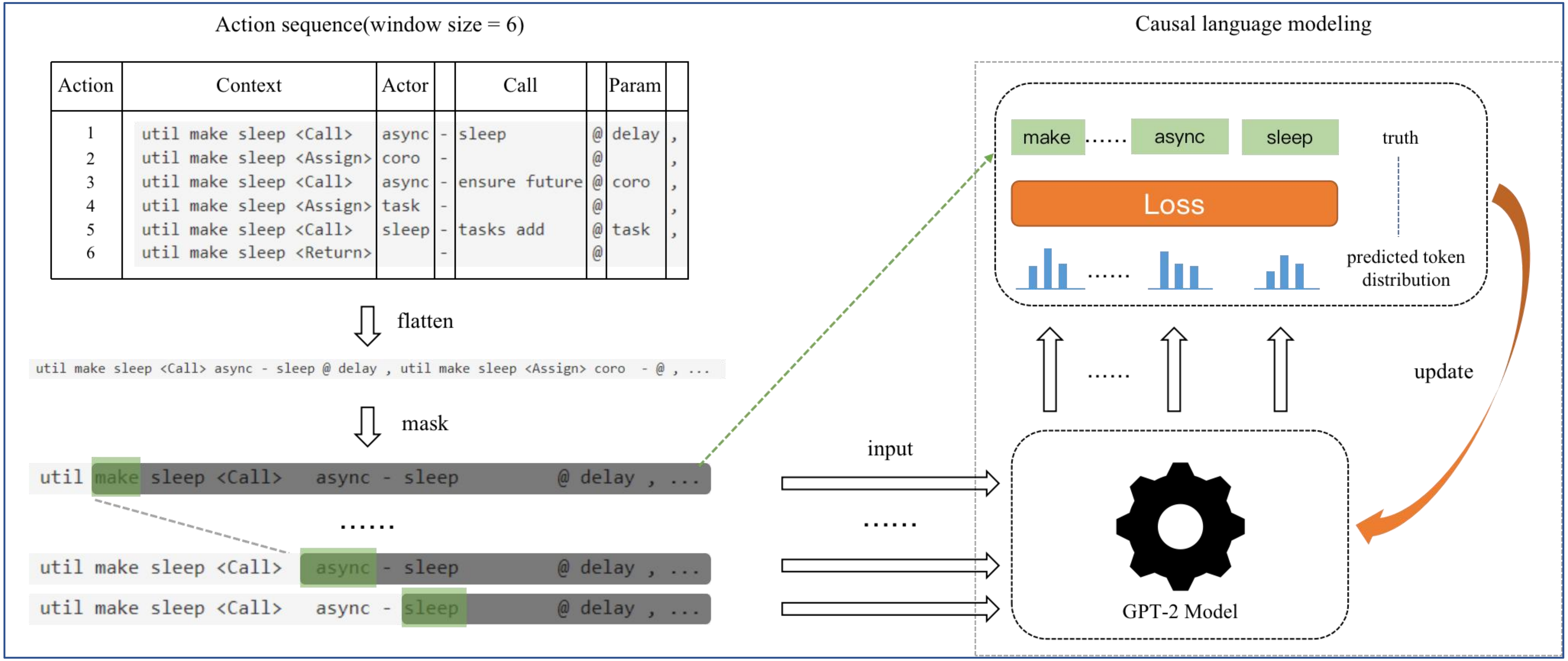
⇨

Action sequence modeling

$$P(act_1, ..., act_n) = \prod_{i=1}^{n} P(act_i | act_1, ..., act_{i-1})$$
$$\approx \prod_{i=1}^{n} P(flat(act_i) | flat(act_1), ..., flat(act_{i-1}))$$

the way for machines to comprehend action sequences in code

Training



Notice that the gpt-2 model here is pre-trained with natural language data. It comprehends the words in action sequences.

Evaluation -- Dataset and RQ

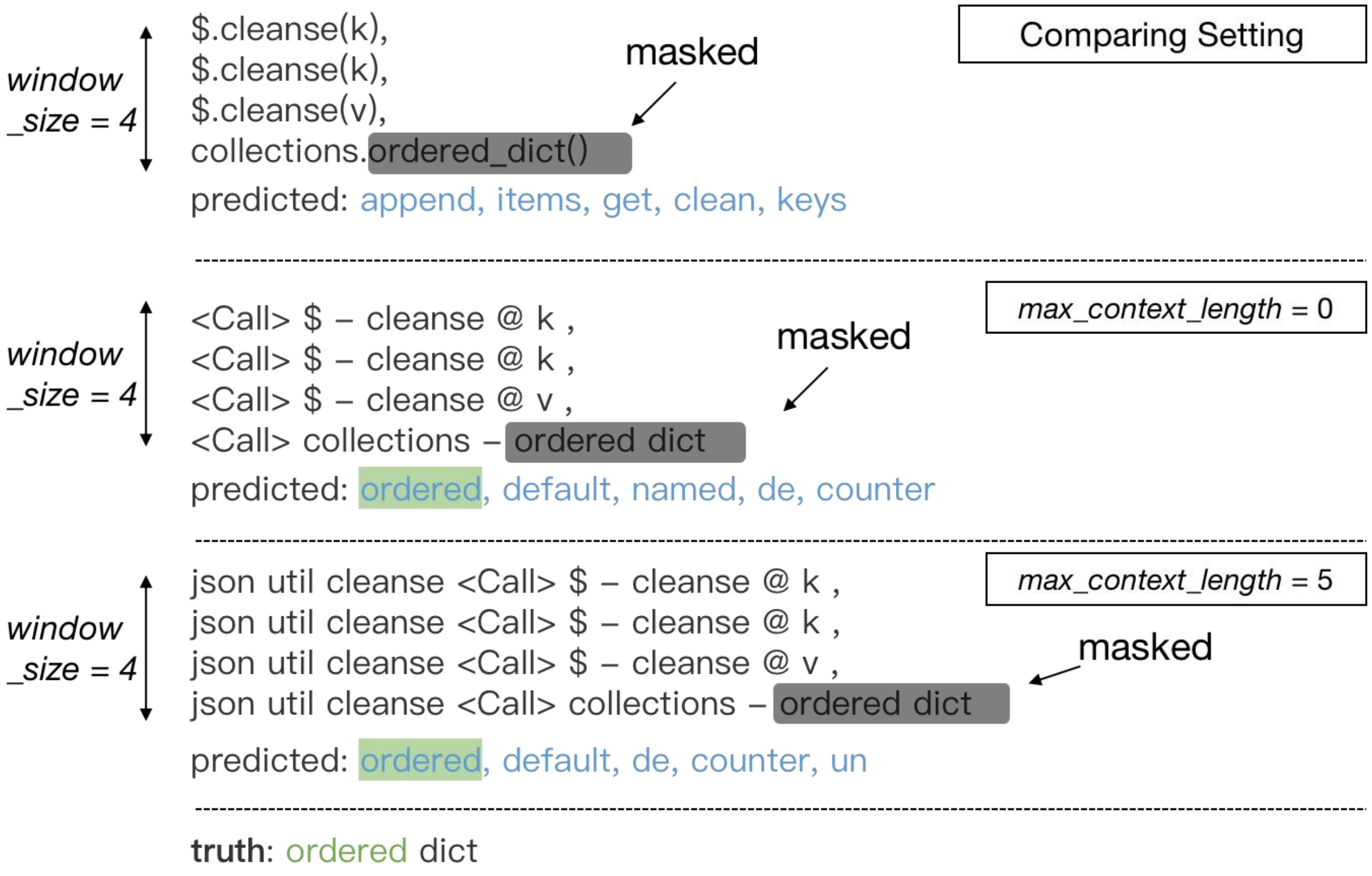
- Dataset
  - Collect from open source community
- Research questions
  1. How useful is the action sequence representation aoproach on method call prediction task?
  2. How useful is the proposed context on method call prediction task?
  3. How does window size influence the method call prediction performance?

Table 2: Dataset Overview

Item	Count
#Repositories	13,006
#Source code files	441,290
#Methods	3,550,572
#Lines	85,238,675
#Tokens	73,335,634
#Unique tokens	175,143

Evaluation

- Baseline model and metrics
  - According to related work[1] experiment setting, we use GPT-2 pre-trained model and formatted data from same dataset to train the baseline model
  - metrics: next method call token prediction top-k accuracy and MRR

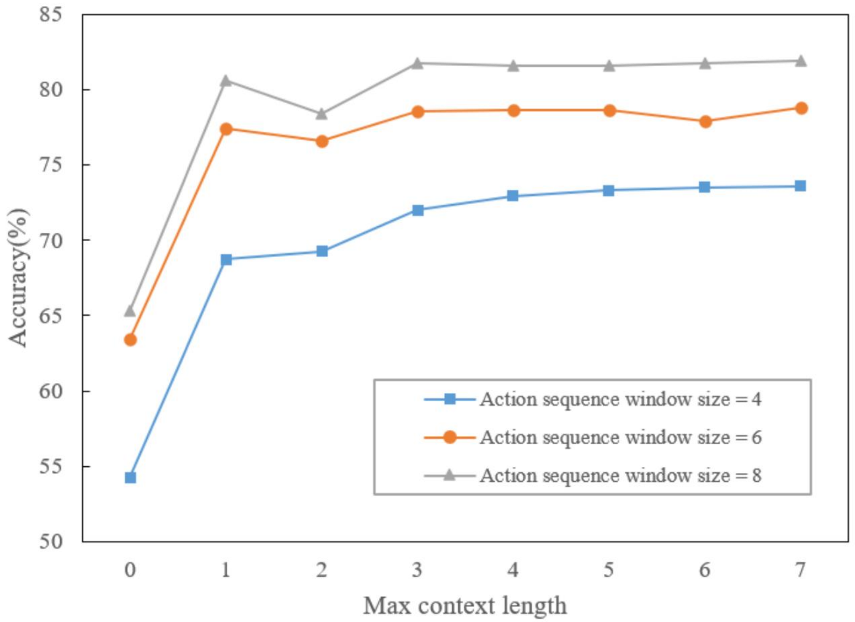


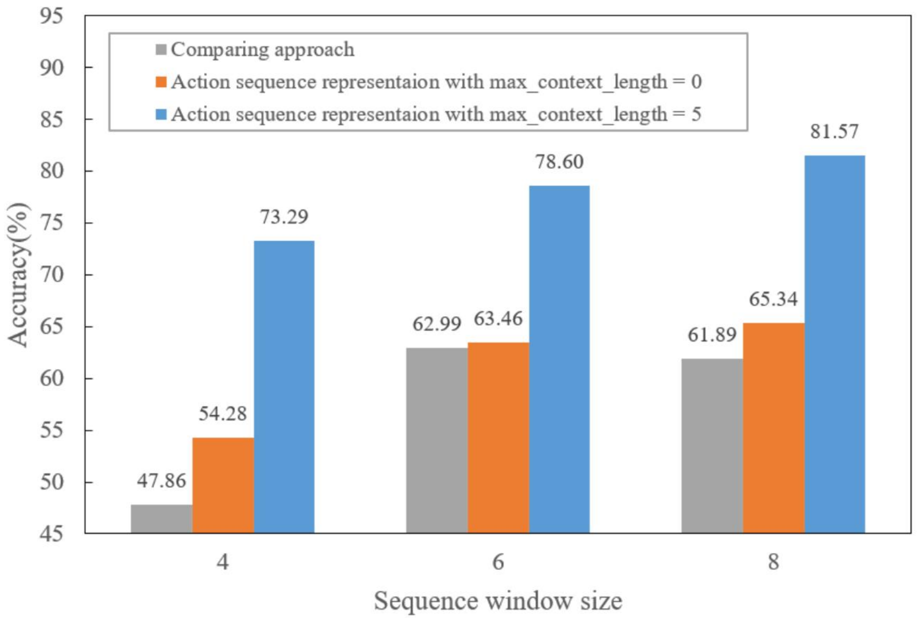
[1]Liu, F., Li, G., Wei, B., Xia, X., Fu, Z., & Jin, Z. (2020, July). A self—attentional neural architecture for code completion with multi—task learning. In Proceedings of the 28th International Conference on Program Comprehension (pp. 37—47).

Evaluation -- Overall

- Code completion performance
  - in method call token prediction our model reaches 81.93% top-5 accuracy and 0.649 MRR
- Training time
  - every 1,000 lines of code < 1s
  - 1, 000, 000 lines of code < 16.7 min
- Prediction time
  - model loading time when first starts < 2s
  - prediction time for one completion position < 9.1 ms

Evaluation - Effctiveness





- Using action sequence code representation we propose impoves the code completion performance compared to baseline model
- The more context given, the better the code completion perfomance
- The larger the action sequence window size, the better the performance

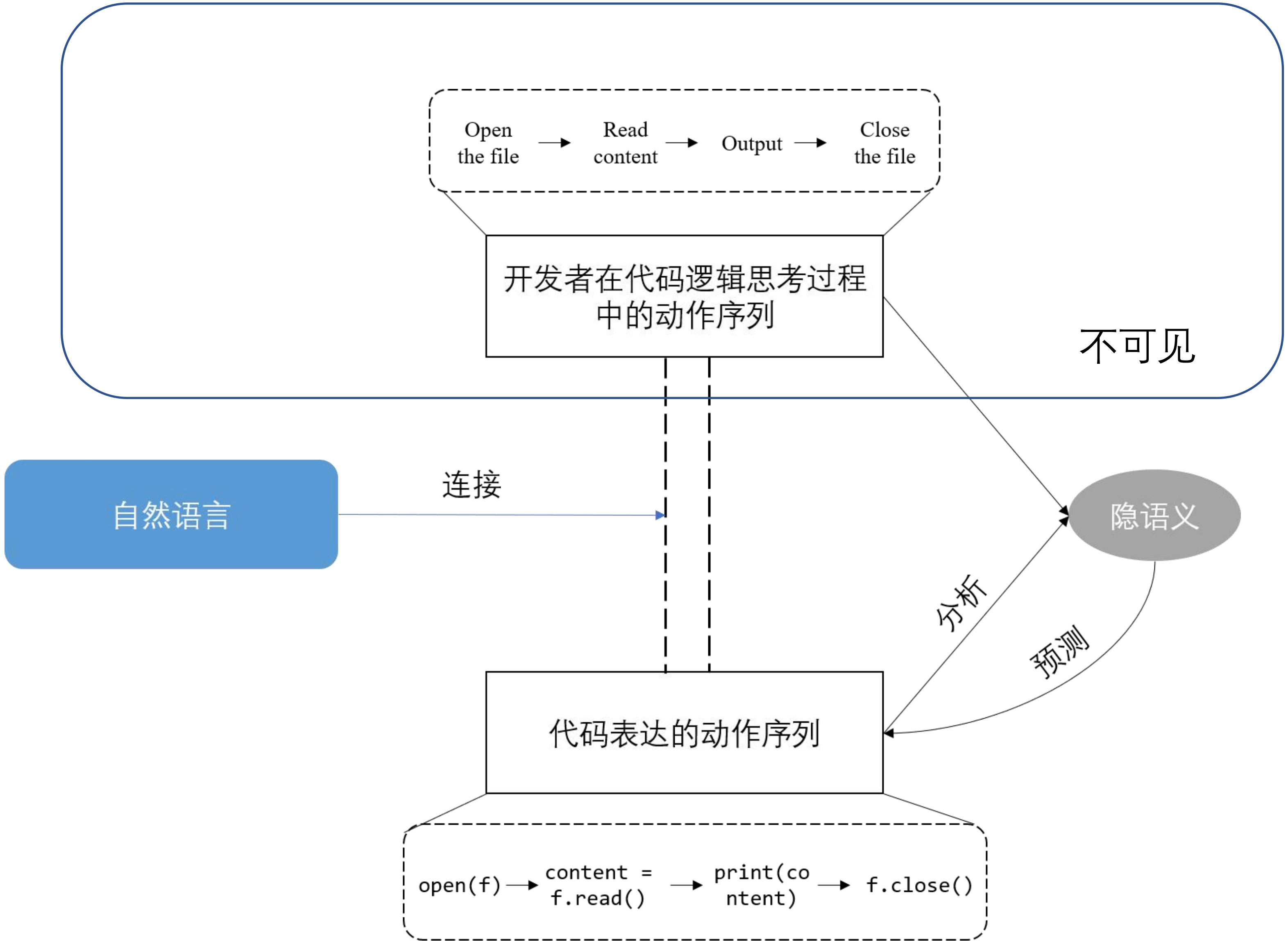
Summary

- We propose a code representation approach focused on action sequence and natural language semantics, the approach could be further used to transfer learning for code data from the Internet
- Based on the representation approach above, we propose a code completion technology taking advantages of open source data and natural language pre-trained models, with both high training and high prediction efficiency

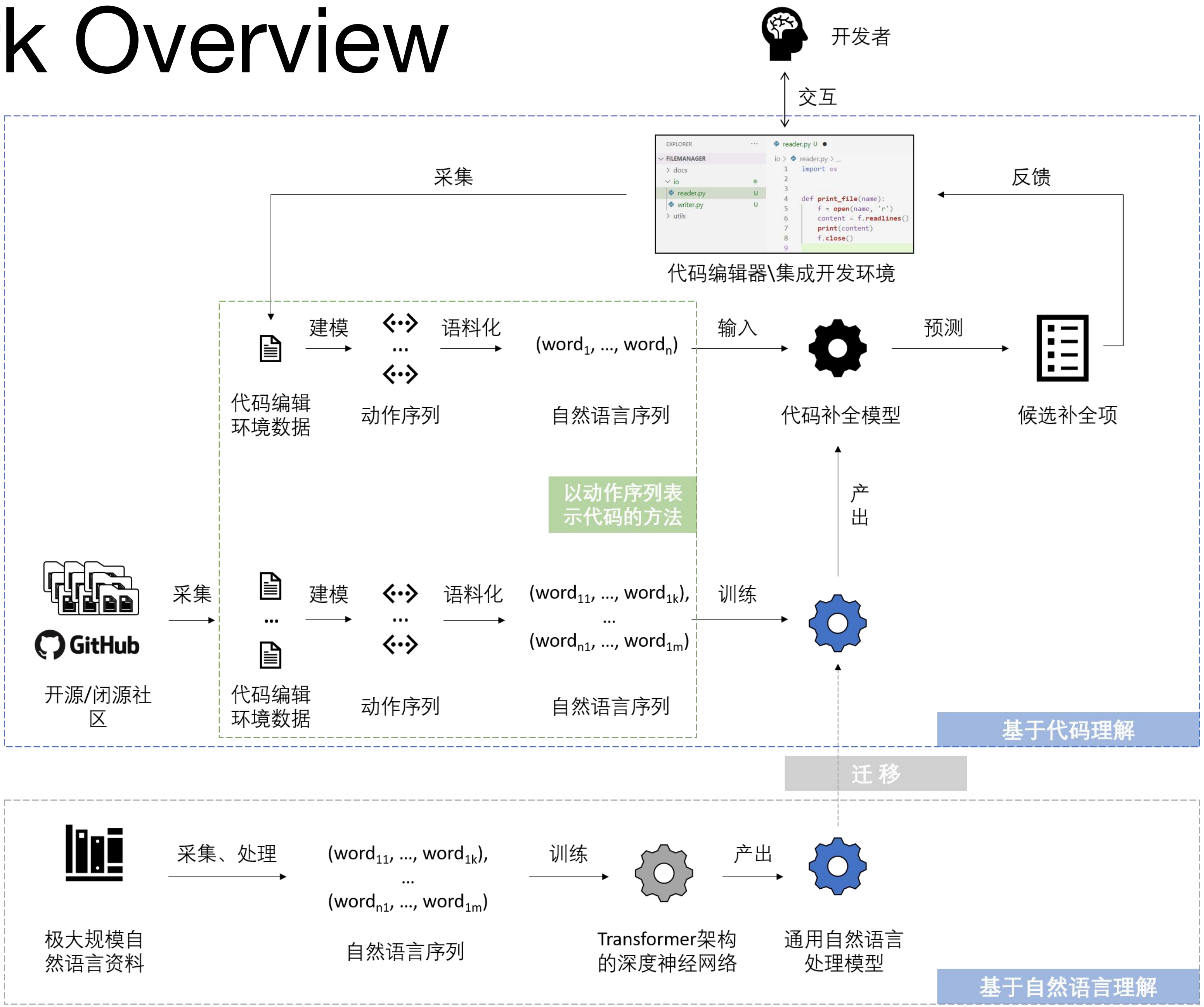
Thanks for listening!

Approach

- The bridge between developers' code and thoughts or logics in their minds aboud program



Framework Overview



The illustrative figure is from hte author’s theis.

实验效果——实验环境与数据采集

- 实验环境
- 数据采集
  - 没有使用公开数据集的原因（论文4.2.1小节）
    - 缺失代码编辑环境的必要信息
    - 缺失代码编辑过程的信息
  - 自行采集数据
    - 使用GitHub API采集数据
    - 应用AST分析工具与标识符分割算法对数据进行预处理

实验环境

处理器	AMD Ryzen 7 5700G
显卡	Nvidia Tesla P100
专用显示内存	16GB
内存	32GB
学习框架	pytorch-CUDA

数据集统计数据

项目	数量
# 代码仓库	13,006
# 源代码文件	441,290
# 方法	3,550,572
# 代码行	85,238,675
# 标识符（分割后）	73,335,634
# 不重复标识符（分割后）	175,143

实验效果——代码补全效果与效率

- 整体补全效果
  - 在测试集上使用Top-5准确率和MRR指标对预测所得方法调用首词进行评估
  - 在动作序列窗口大小为8、最大动作上下文长度为7的参数设置下，Top-5准确率达到81.92%，MRR达到0.649
- 模型效率
  - 实验设置：动作上下文长度≤7的不同配置
  - 训练效率：每千行代码训练时间小于1秒，百万行代码训练时间小于17分钟
  - 预测效率：模型加载时间平均为1.8375秒，预测一次的平均时间约为9.09毫秒
  - 训练效率比对：最近的基于语言模型的用于代码的模型CodeT5[1]，同等规模训练时间为2402058.24秒（约41倍），且对比工作所使用的GPU为A100（40GB显示内存，性能高于本文所用硬件）
  - 预测效率比对：专为预测效率设计的模型[2]，预测一次的平均时间为8毫秒（-1.09毫秒）

[1]Y. Wang, W. Wang, S. Joty, and S. C. H. Hoi, "CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation," arXiv:2109.00859 [cs], Sep. 2021, Accessed: Jan. 19, 2022. [Online]. Available: http://arxiv.org/abs/2109.00859

[2]A. Svyatkovskiy, S. Lee, A. Hadjitofi, M. Riechert, J. Franco, and M. Allamanis, "Fast and Memory-Efficient Neural Code Completion," arXiv:2004.13651 [cs], Mar. 2021, Accessed: Apr. 13, 2021. [Online]. Available: http://arxiv.org/abs/2004.13651

实验效果——有效性验证

- 实验方法
  - 训练数据集不同，不能使用标准数据集验证
  - 对比模型的设置参照本领域相关工作[1]的比对模型设置方法，对比模型使用SOTA自然语言模型训练和验证
- 有效性验证
  - 程序动作序列代码表示方法有效性（基准模型设置1、2）
    - 结构有效性
    - 动作上下文有效性
  - 自然语言迁移至代码的方法有效性（基准模型设置3）
    - 未使用迁移学习的模型效果显著低于基准模型1、基准模型2以及本文模型

基准模型设置 1 使用同样的预训练模型（DistilGPT2）、训练流程、数据集训练，此设置下仅去除动作序列表示方法的介入而采用与原始代码结构相近的建模方法处理数据。

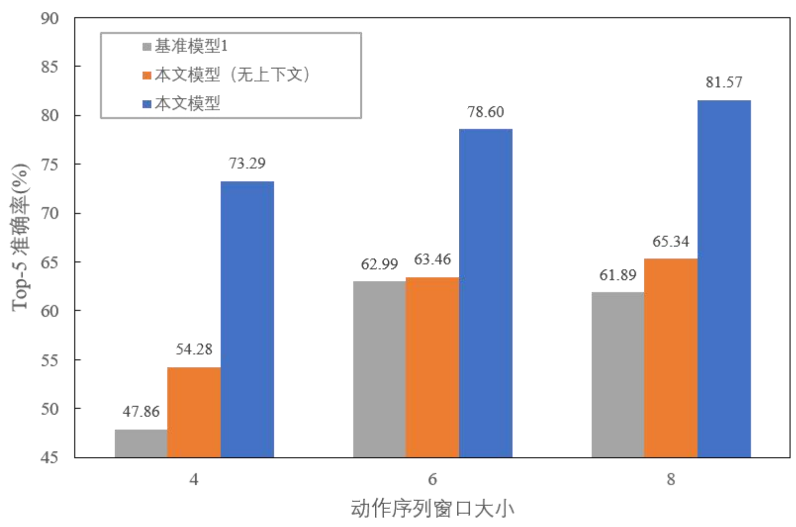
基准模型设置 2 使用同样的预训练模型、训练流程、数据集训练，此设置下仅去除动作序列表示方法的介入而采用与原始代码结构相近的建模方法处理数据，区别于基准模型 1，我们在输入代码的每一行前加入了已格式化的程序上下文信息。图5.3展示了实际验证阶段基准模型 1 与基准模型 2 对代码建模方式的区别，所展示的为同一方法片段输入模型前的语料。

基准模型设置 3 使用同样的预训练模型，但不进行代码语境的迁移学习训练。

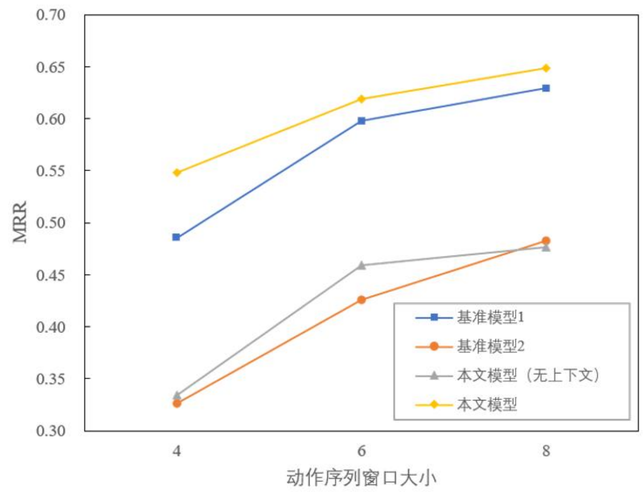
[1]Liu, F., Li, G., Wei, B., Xia, X., Fu, Z., & Jin, Z. (2020, July). A self-attentional neural architecture for code completion with multi-task learning. In Proceedings of the 28th International Conference on Program Comprehension (pp. 37-47).

实验效果——程序动作序列代码表示方法有效性

与基准模型1对比结果



与基准模型1、基准模型2对比结果



基准模型设置1

window_
size = 4

$.cleanse(k),
$.cleanse(k),
$.cleanse(v),
collections.ordered_dict()

被掩码

预测输出：append, items, get, clean, keys

本文模型（无上下文）

window_
size = 4

<Call> $ - cleanse @ k ,
<Call> $ - cleanse @ k ,
<Call> $ - cleanse @ v ,
<Call> collections - ordered dict

被掩码

预测输出：ordered, default, named, de, counter

本文模型

window_
size = 4

json util cleanse <Call> $ - cleanse @ k ,
json util cleanse <Call> $ - cleanse @ k ,
json util cleanse <Call> $ - cleanse @ v ,
json util cleanse <Call> collections - ordered dict

被掩码

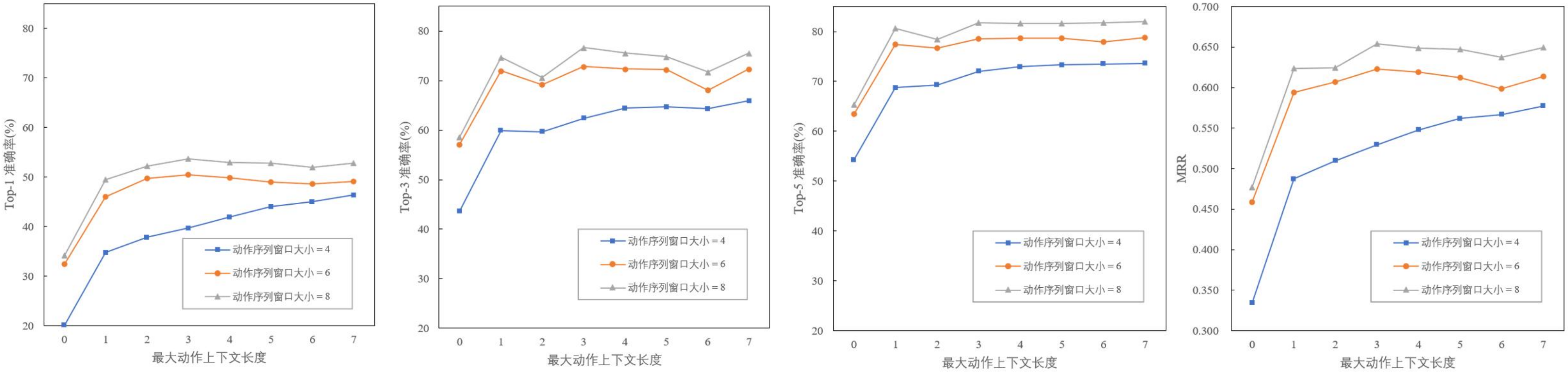
预测输出：ordered, default, de, counter, un

正确输出： ordered dict

真实预测样例

动作序列建模方式在窗口大小较小时可显著提高补全效果；增加动作上下文可在所有对比情况中显著提高补全效果

实验效果——程序动作序列代码表示方法有效性



不同最大动作上下文长度与动作序列窗口大小对补全效果的影响

动作序列窗口越大，效果一般越好；
最大动作上下文长度越大，效果一般越好

实验效果——模型解读

- Group2为模型对动作上下文的使用情况

window_ size = 4

```
$.cleanse(k),
$.cleanse(k),
$.cleanse(v),
collections.ordered_dict()
```

被掩码

基准模型设置1

预测输出：append, items, get, clean, keys

window_ size = 4

```
<Call> $ - cleanse @ k ,
<Call> $ - cleanse @ k ,
<Call> $ - cleanse @ v ,
<Call> collections - ordered dict
```

被掩码

本文模型（无上下文）

预测输出：ordered, default, named, de, counter

window_ size = 4

```
json util cleanse <Call> $ - cleanse @ k ,
json util cleanse <Call> $ - cleanse @ k ,
json util cleanse <Call> $ - cleanse @ v ,
json util cleanse <Call> collections - ordered dict
```

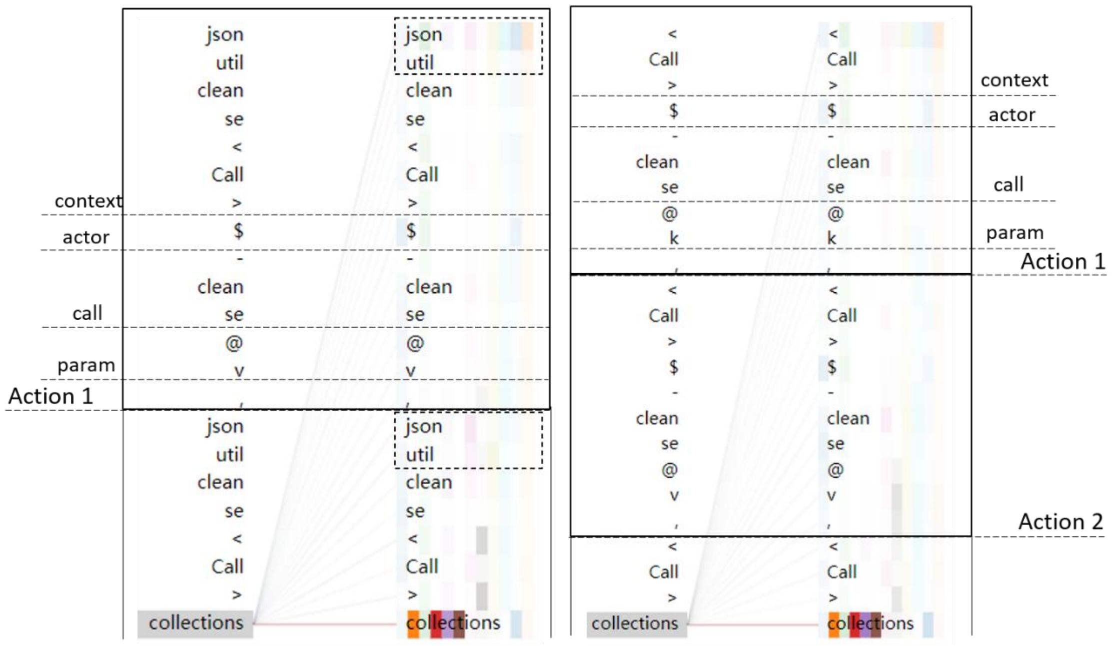
本文模型

被掩码

预测输出：ordered, default, de, counter, un

正确输出： ordered dict

Group 2: Explanation of *collections*(actor) with and without context



Group2, Sample 2
max_context_length = 6

Group2, Sample 1
max_context_length = 0

实验效果——模型解读

- 右图为模型对动作序列中不同动作在整个输入序列中界限的学习情况

Group 3: Learned border of belonging action(the gap between actions)