



MindSpore

# 智能辅助编程

作者：Chissica

## 智能化软件开发-智能辅助编程时代

### ● 应用场景：

随着深度学习等人工智能技术的发展以及深度学习等人工智能技术在自然语言处理领域取得的成功，软件智能化开发逐渐成为了软件工程领域的一个热门领域。软件智能化开发覆盖了软件开发生命周期中的众多场景，包括代码推荐、代码搜索、注释生成、缺陷检测、程序转换等场景。代码推荐与代码搜索能够提升开发人员的开发编码效率，解决编码过程中遇到的编码相关的问题，缩短开发周期（更进一步我们可以考虑推荐最佳实践，不仅实现开发效率的性能倍增，还实现代码运行效率的性能倍增）；注释生成能够帮助开发人员理解代码片段，从而提高代码阅读理解能力，缺陷检测能够提升开发人员编码的质量，在开发阶段尽可能多的发现潜在缺陷；程序转换能够辅助开发人员进行语言迁移，更快的将某种特性用另一种语言实现。

### IDE 1.0



- 辅助开发
- 完全依靠人的主观能动性
- 基本的语法高亮、简单的代码推荐、代码重构等

### IDE 2.0



- 智能化辅助开发
- 以人为主，机器为辅
- 代码推荐、代码搜索、注释生成、缺陷检测、程序转换等

### IDE 3.0



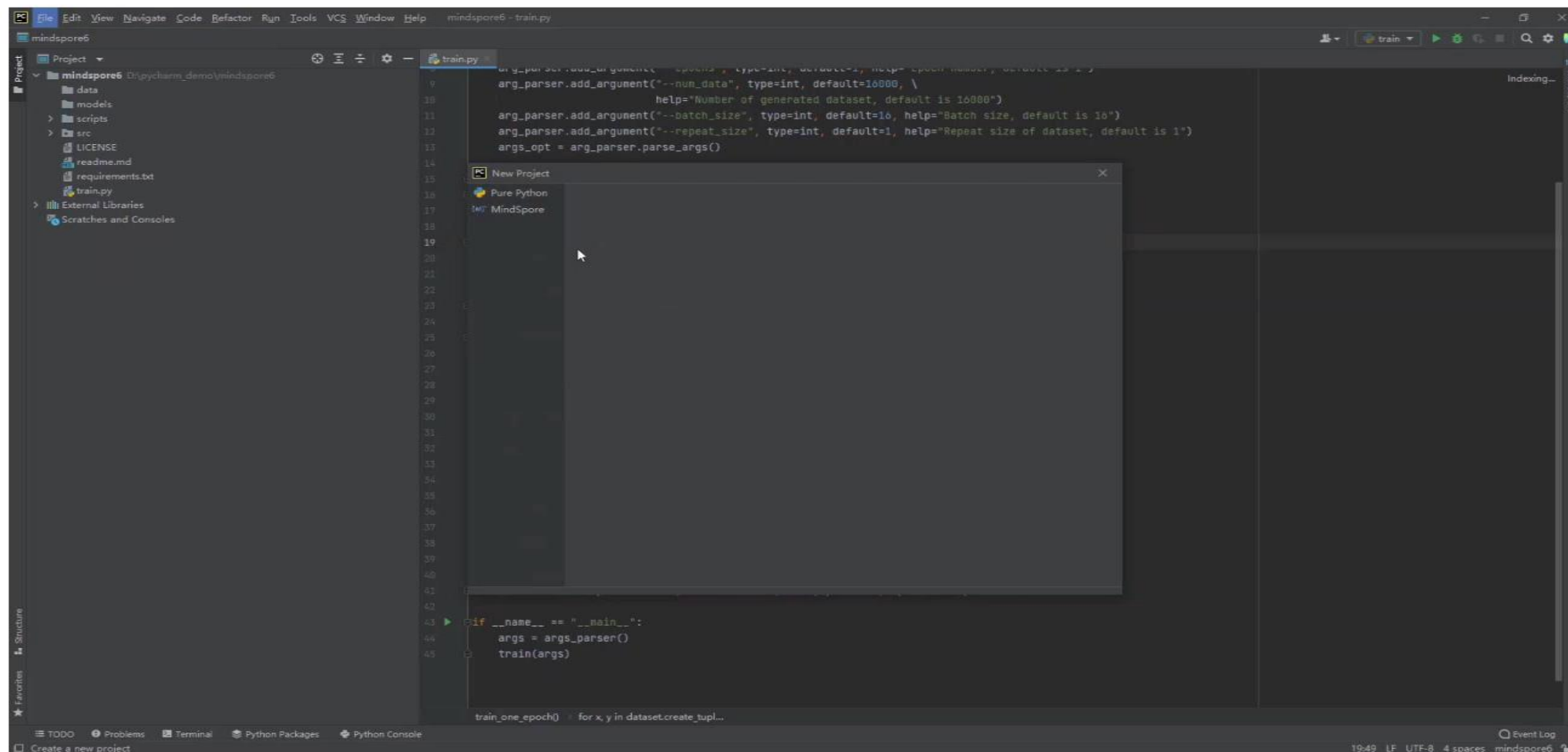
- 智能化开发
- 机器为主，以人为辅
- 自动编程，人在必要的时候解决机器不能解决的问题

## What we do?

一键安装  
MindSpore

智能补全  
MindSpore代码

智能搜索相关算  
子、文档



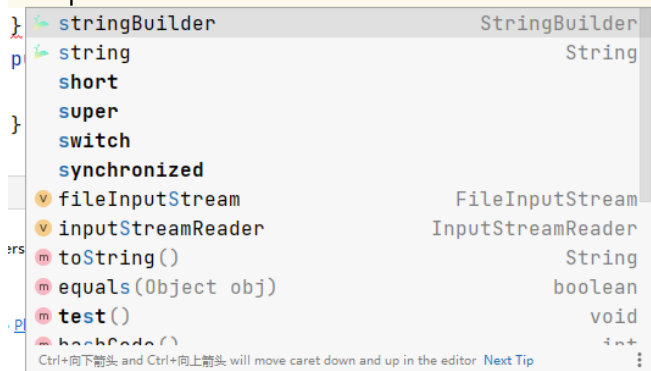
## 智能补全相关概念术语

英文	中文	描述
token	标识符	标识符是将源代码进行分词后的最小单位。通常源代码以空格进行切分，而每一个标识符内部又可以通过驼峰命名法等方式继续进行分词。代码中的关键词、变量名、方法名等都属于标识符。
token sequence	标识符序列	标识符按顺序构成的序列称为标识符序列。
API call	API调用	第三方库的API方法调用以及API成员变量访问统称为API调用。
API sequence	API序列	API调用按照一定调用顺序构成的序列称为API调用序列，简称为API 序列。
API usage pattern	API使用模式	API使用模式是包含多个API调用以及API调用之间的调用关系的模式。通常情况下，只有达到一定出现次数的API使用方式才能称其为模式。经常出现的API调用序列可以看作序列化的API使用模式。
incomplete code	不完整代码	开发人员正在编写的方法由于缺少某些代码（例如API调用）而形成的代码称为不完整代码。
hole	空缺部分	代码中需要进行补全的地方称为空缺部分。

## 智能补全场景-token补全

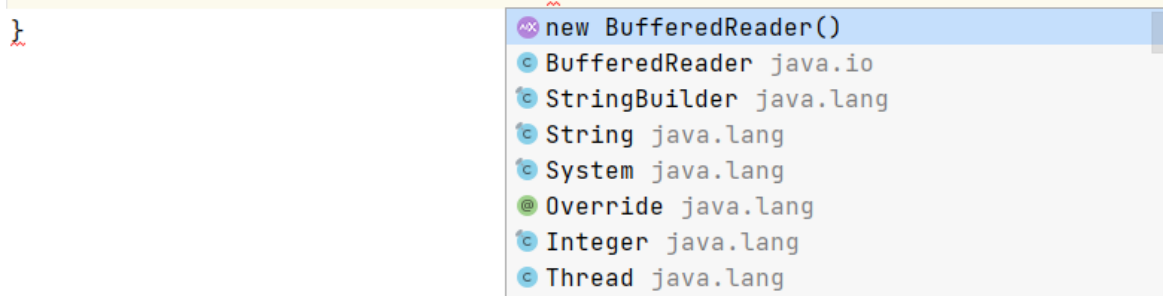
**单个标识符推荐 (next token recommendation) :** 对于一段给定的包含空缺部分的不完整代码，如果空缺部分需要推荐的是一个代码标识符，那么这种场景称为下一个标识符推荐（单个标识符推荐）。下一个标识符推荐中所推荐的标识符可以为某一个API中的方法或成员变量。

```
public String readFile(String path) throws Exception{
    File file = new File(path);
    FileInputStream fileInputStream = new FileInputStream(file);
    InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    StringBuilder stringBuilder = new StringBuilder();
    String string = null;
    while ((string = bufferedReader.readLine()) != null) {
        stringBuilder = stringBuilder.append(string);
    }
    s|
```



**标识符序列推荐 (token sequence recommendation) :** 对于一段给定的包含空缺部分的不完整代码，如果空缺部分需要推荐的是代码标识符序列，那么这种场景称为标识符序列推荐。

```
public String readFile(String path) throws Exception{
    File file = new File(path);
    FileInputStream fileInputStream = new FileInputStream(file);
    InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
    BufferedReader bufferedReader =
```



## 智能补全场景-API补全

**单个API推荐 (next API recommendation)**：对于一段给定的包含空缺部分的不完整代码，如果空缺部分需要推荐的是一个API调用，那么这种场景称为单个API推荐。单个API推荐又可细分为两种场景，一种场景为基于给定的某个API类的对象，在键入“.”后推荐最符合当前代码上下文的API调用；另一种场景为无须提供指定的API类并且无须键入“.”，而是以行为单位，推荐该行需要最符合当前代码上下文的API调用。

```
public String readFile(String path) throws Exception{
    File file = new File(path);
    FileInputStream fileInputStream = new FileInputStream(file);
    InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    StringBuilder stringBuilder = new StringBuilder();
    String string = null;
    while ((string = bufferedReader.readLine()) != null) {
        stringBuilder.|
    }
}
```

append(int i)	StringBuilder
append(char c)	StringBuilder
append(float f)	StringBuilder
append(double d)	StringBuilder
append(long lng)	StringBuilder
append(boolean b)	StringBuilder
append(char[] str)	StringBuilder
append(Object obj)	StringBuilder

```
public String readFile(String path) throws Exception{
    File file = new File(path);
    FileInputStream fileInputStream = new FileInputStream(file);
    InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    StringBuilder stringBuilder = new StringBuilder();
    String string = null;
    while ((string = bufferedReader.readLine()) != null) {
        |
    }
}
```

stringBuilder = stringBuilder.append(string);
stringBuilder = stringBuilder.append(string)...
if(){}
stringBuilder = stringBuilder.append(string)...
arrayList.add(string);
stringBuilder = stringBuilder.append(charVar...
stringBuilder = stringBuilder.append(string);
string = stringBuilder.toString();

## 智能补全场景-API补全

**API序列推荐 (API sequence recommendation)**：对于一段给定的包含空缺部分的不完整代码，如果空缺部分需要推荐的是API序列，那么这种场景称为API序列推荐。

```
import java.io.FileReader;

public class Example1 {
    public String readFile(String path) throws Exception {
        FileReader fileReader = new FileReader(path);
    }
}
```

Select

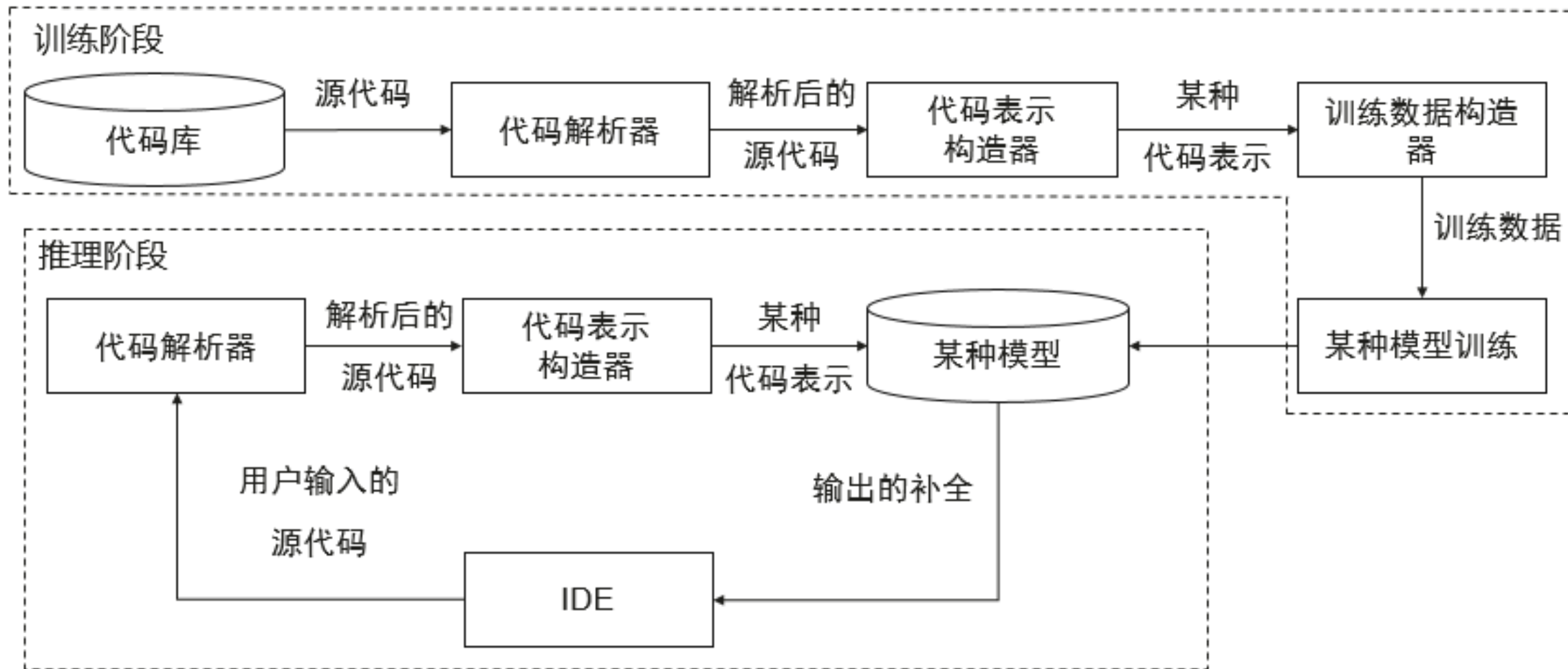
BufferedReader bufferedReader = new BufferedReader(fileReader);  
String string;  
StringBuffer stringBuffer = new StringBuffer();  
while((string = bufferedReader.readLine()) != null){  
 stringBuffer.append(string);  
}  
bufferedReader.close();  
string = stringBuffer.toString();  
return string;

Select

BufferedReader bufferedReader = new BufferedReader(fileReader);  
StringBuilder stringBuilder = new StringBuilder();  
String string;  
while((string = bufferedReader.readLine()) != null){  
 stringBuilder.append(string);  
}  
bufferedReader.close();  
string = stringBuilder.toString();  
return string;



## 智能补全流程架构





## 业界现状

## 序列化代码表示 + 深度学习大模型



• 功能：代码补全、代码搜索

without Kite	with Kite
<pre>1 import numpy as np 2 3 a = np.array([1, 2, 3]) 4 a.</pre>	<pre>1 import numpy as np 2 3 a = np.array([1, 2, 3]) 4 a.    shape = (    shape    T    astype(...)</pre>



• 功能：代码补全、代码搜索

```
private void demo() throws Exception {
    URLConnection s = new URL(url).openConnection();
    InputStreamReader content =
        new InputStreamReader(s.getInputStream());
    new java.io.InputStreamReader(dstream, url);
    new InputStreamReader(inputStream, url);
}
```



• 功能：代码补全、代码搜索

```
1 from argparse import ArgumentParser
2 from boto3 import Session
3 from botocore.exceptions import BotocoreError, ClientError
4
5 cli = ArgumentParser(description="GetLexicon example")
6 cli.add_argument("name", type=str, metavar="LEXICON_NAME")
7 arguments = cli.parse_args()
8
9 sess = Session(profile
               session = Session(client
               session = Session
```



• 功能：代码补全、代码搜索

```
public class FooConsumerBootstrap {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext consumerContext = new
        consumerContext.register(ConsumerConfiguration.class);
        d
        consumerContext
        consumerContext.
        char
        class
        cxf
        Generate CXF Rest web se
        ^↓ and ^↑ will move caret down and up in the editor Next Tip
```



• 功能：代码生成

```
1 #!/usr/bin/env ts-node
2
3 import { fetch } from "fetch-h2";
4
5 // Determine whether the sentiment of text is positive
6 // Use a web service
7 async function isPositive(text: string): Promise<boolean> {
8     const response = await fetch('http://text-processing.com/api/sentiment/', {
9         method: "POST",
10         body: `text=${text}`,
11         headers: {
12             "Content-Type": "application/x-www-form-urlencoded",
13         },
14     });
15     const json = await response.json();
16     return json.label === "pos";
17 }
```




• 功能：代码生成

```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5 + 10;
vector<int> g[N];
int n, d[N], cnt[N];
void dfs(int x, int fa) {
    d[x] = d[fa] + 1;
    for (int i = 0; i < g[x].size(); i++) {
        int v = g[x][i];
        if (v == fa) continue;
        dfs(v, x);
    }
}
int ans;
void getans(int x, int fa) {
    int tot = 0;
    for (int i = 0; i < g[x].size(); i++) {
        int v = g[x][i];
        if (v == fa) continue;
        getans(v, x);
        tot += cnt[v];
    }
}
```


# 学术界现状

部分与业界技术类似，部分在代码表示上深入探索





computer science bibliography

a service of  SCHLOSS DAGSTUHL  
Leibniz Center for Informatics

Search dblp

powered by CompleteSearch, courtesy of Hannah Bast, University of Freiburg

[Home](#)

Publication search results

found 125 matches

2022

Kang Yang, Huiqun Yu, Guisheng Fan, Xingguang Yang, Zijie Huang:  
A graph sequence neural architecture for code completion with semantic structure features. J. Softw. Evol. Process. 34(1) (2022)

Qi Liu, Xiao Peng Li, Jicheng Yang:  
Optimum CodeSgen for Image Denoising Between Type-2 Fuzzy Identifier and Matrix Completion Denoiser. IEEE Trans. Fuzzy Syst. 30(1): 287-292 (2022)

Maliheh Izadi, Roberta Gismonti, Georgios Gousios:  
CodeFill: Multi-token Code Completion by Jointly Learning from Structure and Naming Sequences. CoRR abs/2202.06689 (2022)

2021

Kenta Terada, Yutaka Watanobe:  
Code completion for programming education based on deep learning. Int. J. Comput. Intell. Stud. 10(2/3): 78-98 (2021)



computer science bibliography

a service of  SCHLOSS DAGSTUHL  
Leibniz Center for Informatics

Search dblp

powered by CompleteSearch, courtesy of Hannah Bast, University of Freiburg

[Home](#)

Publication search results

found 882 matches

2022

Muhammad Hammad, Onder Babur, Hamid Abdul Basit, Mark van den Brand:  
Clone-Seeker: Effective Code Clone Search Using Annotations. IEEE Access 10: 11696-11713 (2022)

Marco Bertini, Francesco Ferrante, Dario Duca:  
Empathes: A general code for nudged elastic band transition states search. Comput. Phys. Commun. 271: 108224 (2022)

Chao Liu, Xin Xia, David Lo, Cuiyun Gao, Xiaohu Yang, John C. Grundy:  
Opportunities and Challenges in Code Search Tools. ACM Comput. Surv. 54(9): 196:1-196:40 (2022)

Zhonghui Mei, Xiaoyan Zhou:  
Maximum Decoding Clique Based Maximum Weight Vertex Search Algorithm for Buffered Instantly Decodable Network Codes. IEEE Commun. Lett. 26(2): 229-233 (2022)

Chaozheng Wang, Zhenhao Nong, Cuiyun Gao, Zongjie Li, Jichuan Zeng, Zhenchang Xing, Yang Liu:  
Enriching query semantics for code search with reinforcement learning. Neural Networks 145: 22-32 (2022)



computer science bibliography

a service of  SCHLOSS DAGSTUHL  
Leibniz Center for Informatics

Search dblp

powered by CompleteSearch, courtesy of Hannah Bast, University of Freiburg

[Home](#)

Publication search results

found 115 matches

2022

Jessie Galasso, Michalis Famelis, Houari A. Sahraoui:  
Code Sophistication: From Code Recommendation to Logic Recommendation. CoRR abs/2201.07674 (2022)

Jiyang Zhang, Chandra Shekhar Maddila, Ram Bairi, Christian Bird, Ujjwal Raizada, Apoorva Agrawal, Yamini Jhavar, Kim Herzig, Arie van Deursen:  
Using Large-scale Heterogeneous Graph Representation Learning for Code Review Recommendations. CoRR abs/2202.02385 (2022)

2021

Fengshuang Li, Chao Zhang, Kewu Peng, Aleksei Krylov, Aleksandr A. Katyshtny, Andrey V. Rashich, Dmitry A. Tkachenko, Sergey B. Makarov, Jian Song:  
Review on 5G NR LDPC Code: Recommendations for DTTB System. IEEE Access 9: 155413-155424 (2021)

Moataz Chouchen, Ali Ouni, Mohamed Wiem Mkaouer, Raula Gaikovina Kula, Katsuro Inoue:



computer science bibliography

a service of  SCHLOSS DAGSTUHL  
Leibniz Center for Informatics

Search dblp

powered by CompleteSearch, courtesy of Hannah Bast, University of Freiburg

[Home](#)

Publication search results

found 102 matches

2022

Guang Yang, Ke Liu, Xiang Chen, Yanlin Zhou, Chi Yu, Hao Lin:  
CCGIR: Information retrieval-based code comment generation method for smart contracts. Knowl Based Syst. 237: 107858 (2022)


2021

Zheng Li, Yonghao Wu, Bin Peng, Xiang Chen, Zeyu Sun, Yong Liu, Deli Yu:  
SeCNN: A semantic CNN parser for code comment generation. J. Syst. Softw. 181: 111036 (2021)


Guang Yang, Xiang Chen, Jinxin Cao, Shuyuan Xu, Zhanqi Cui, Chi Yu, Ke Liu:  
ComFormer: Code Comment Generation via Transformer and Fusion Method-based Hybrid Code Representation. DSA 2021: 30-41

Alexandr Kuznetsov, Katerina Kuznetsova:  
Comment on "Particle Swarm Optimization Based Highly Nonlinear Substitution-Boxes Generation for Security Applications". IDAACS 2021: 485-488

Zehua Zeng, Chenyang Tu, Neng Gao, Cong Xue, Cunqiang Ma, Yiwei Shant:  
CMVCG: Non-autoregressive Conditional Masked Live Video Comments Generation Model. IJCNN 2021: 1-8



computer science bibliography

a service of  SCHLOSS DAGSTUHL  
Leibniz Center for Informatics

Search dblp

powered by CompleteSearch, courtesy of Hannah Bast, University of Freiburg

[Home](#)

Publication search results

found 106 matches

2022

Chi Chen, Xin Peng, Bihuan Chen, Jun Sun, Zhenchang Xing, Xin Wang, Wenyun Zhao:  
"More Than Deep Learning": post-processing for API sequence recommendation. Empir. Softw. Eng. 27(1): 15 (2022)

Honghao Gao, Xi Qin, Ramón J. Durán Barroso, Walayat Hussain, Yueshen Xu, Yuyu Yin:  
Collaborative Learning-Based Industrial IoT API Recommendation for Software-Defined Devices: The Implicit Knowledge Discovery Perspective. IEEE Trans. Emerg. Top. Comput. Intell. 6(1): 66-76 (2022)

Zarrin Tasnim Sworna, Chadni Islam, Muhammad Ali Babar:  
APIRO: A Framework for Automated Security Tools API Recommendation. CoRR abs/2201.07959 (2022)

2021

Yueshen Xu, Yinchun Wu, Honghao Gao, Shengli Song, Yuyu Yin, Xichu Xiao:  
Collaborative APIs recommendation for Artificial Intelligence of Things with information fusion.



复旦大学



北京大学



南京大学



facebook



德克萨斯达拉斯分校

## 智能补全技术路线概述

### 基于类型匹配的方法

基于类型匹配的可用于单个API 推荐的代码推荐方法首先获取指定API 类中的API方法和成员变量，然后通过API 流行度、启发式规则等方式对该API类中的API 方法和成员变量进行排序。

### 基于模式挖掘的方法

基于模式挖掘的代码推荐方法主要利用挖掘算法或模型从源代码中挖掘出API 使用模式，从而基于挖掘得到的API 使用模式对开发人员提供的代码片段或API 使用方式的查询进行匹配，推荐相应的API 使用模式。

### 基于特征匹配的方法

基于特征匹配的代码推荐方法首先定义代码中的特征并通过特征提取将代码转换为某种代码特征表示。在进行推荐时，基于特征匹配的代码推荐方法将开发人员正在编写的代码中的特征与从代码库中提取到的代码中的特征进行搜索匹配，从而推荐特征较为匹配的代码给开发人员。

### 基于传统统计模型的方法

基于传统统计模型的代码推荐方法的核心思想为将源代码解析并处理为某种代码表示（如代码标识符序列、控制流数据流图等），并利用传统统计模型（如n-gram 模型等）对代码表示进行建模、学习和训练，进而利用训练好的统计模型进行代码推荐。

### 基于深度学习的方法

基于深度学习的代码推荐方法的核心思想为将源代码解析并处理为某种代码表示（如代码标识符序列、抽象语法树等），并利用深度学习模型（如长短时记忆神经网络等）对代码表示的语义进行学习和训练，进而利用训练好的深度学习模型进行代码推荐。

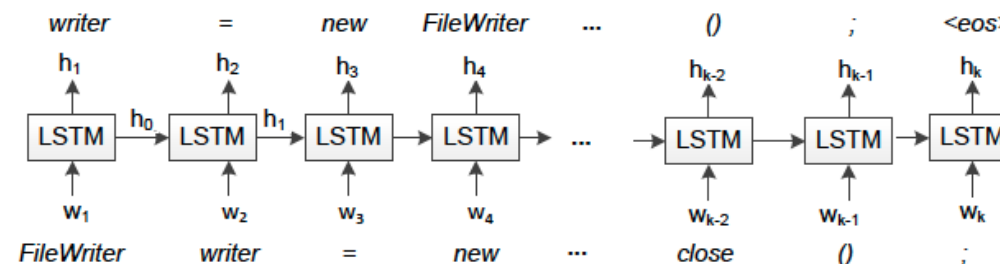
## 基于自然语言模型的智能补全

将代码处理为标识符（即）序列，简单的处理方式为直接将代码token化。每一个token也可以附加额外信息，如这个token的数据类型，这个token的角色(如是一个变量还是一个方法调用)等。利用n-gram模型或sequence-based深度学习模型（如LSTM）进行训练。

### n-gram模型

$$P(t_0 \dots t_M) = \prod_{m=0}^M P(t_m | t_{m-1} \dots t_{m-n+1})$$

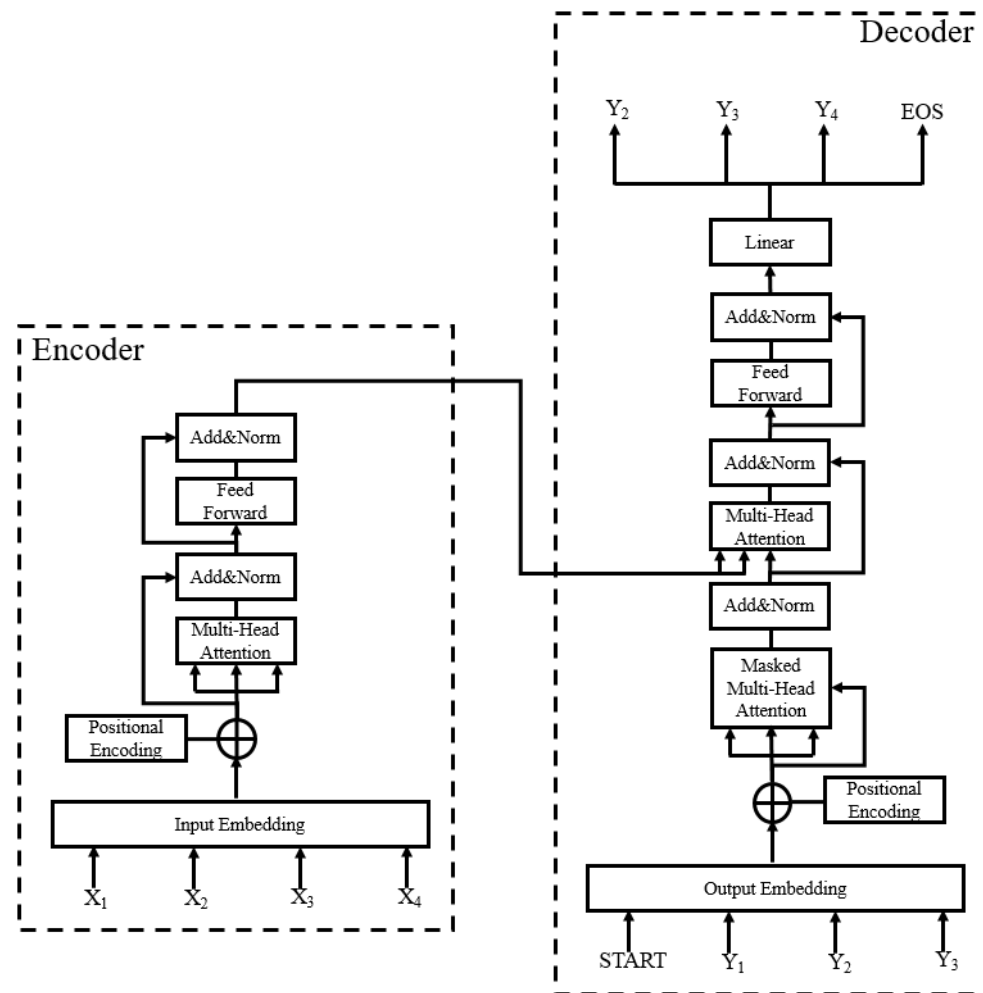
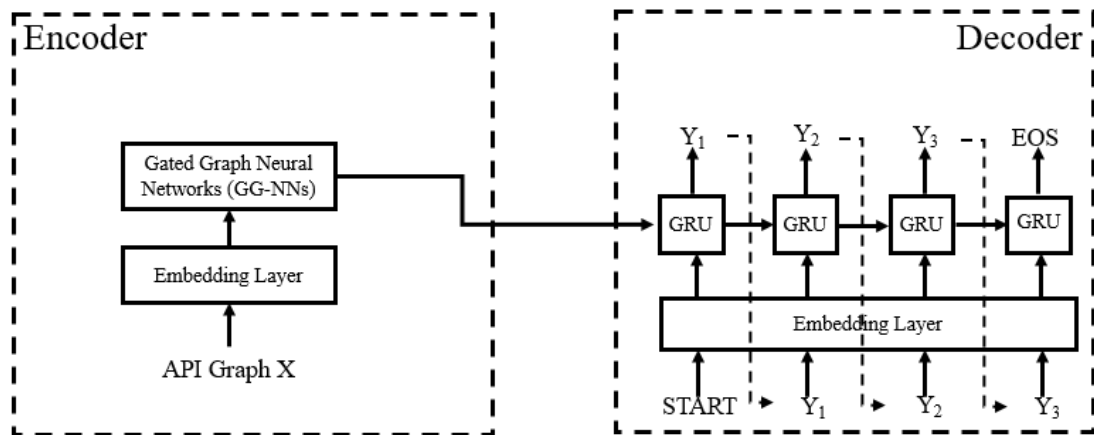
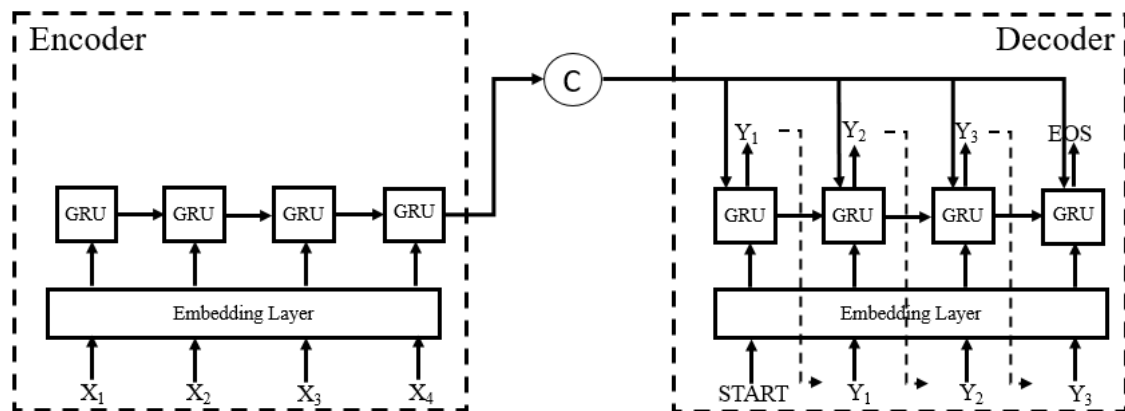
### LSTM模型



局限性：缺乏对代码本身程序语法以及结构特性的考虑

## 基于自然语言模型的智能补全

局限性：依赖深度学习大模型的学习能力



## 智能补全整体技术架构

## 功能特性

多平台多底座适配

多层次代码补全粒度

多语言动态适配

## 技术特性

深度学习+程序分析+知识图谱

预处理+后处理

关键特性

关键方法与技术

底座环境

Token补全

Token序列补全

API补全

API序列补全

代码片段补全

参数补全

## 1 程序分析

代码表示分析

控制流分析

数据流分析

代码语义提取

文本表示分析

文本分词

文本语义提取

文本组合

## 2 深度学习

模型构建

大模型

图神经网络 (GNN)

编码器-解码器框架...

模型训练

单任务训练

预训练-微调

迁移学习...

模型部署

无损本度化部署

有损本度化部署

云端部署

## 3 知识图谱

知识图谱构建

实体抽取

概念抽取

实体链接...

知识图谱融合

AI+Embedding

AI+知识提取

AI+关键路径...

## 4 数据预处理

数据爬取

平台多元化

语言多元化

高质量化

数据清洗

去重

噪音过滤

最佳实践提取...

## 5 后处理

词法语法分析

基于知识的  
协同过滤

包含关系过滤...

系统平台

windows

linux

mac os

IDE底座

IntelliJ IDEA

PyCharm

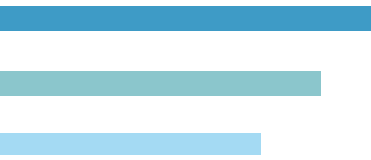
VS Code

Deveco Studio...

已实现

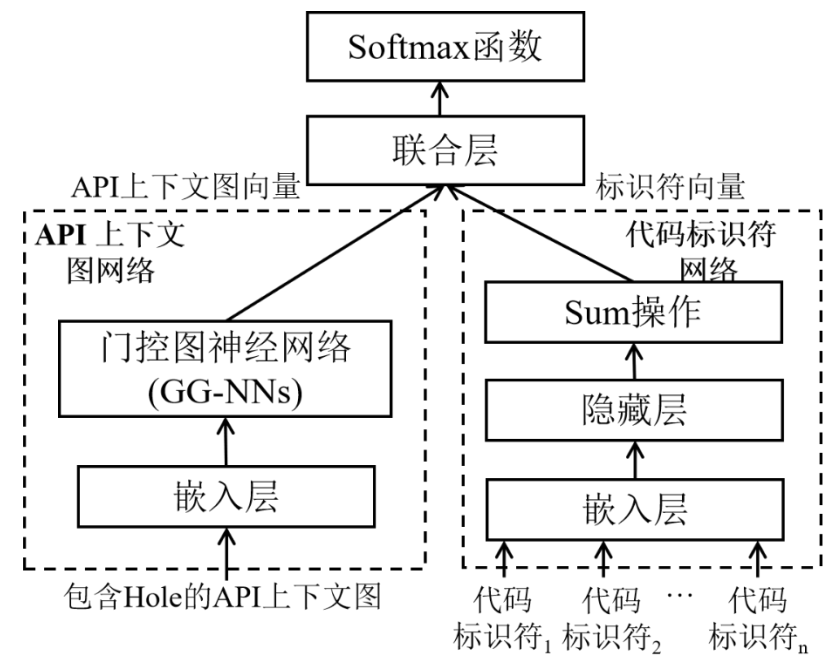
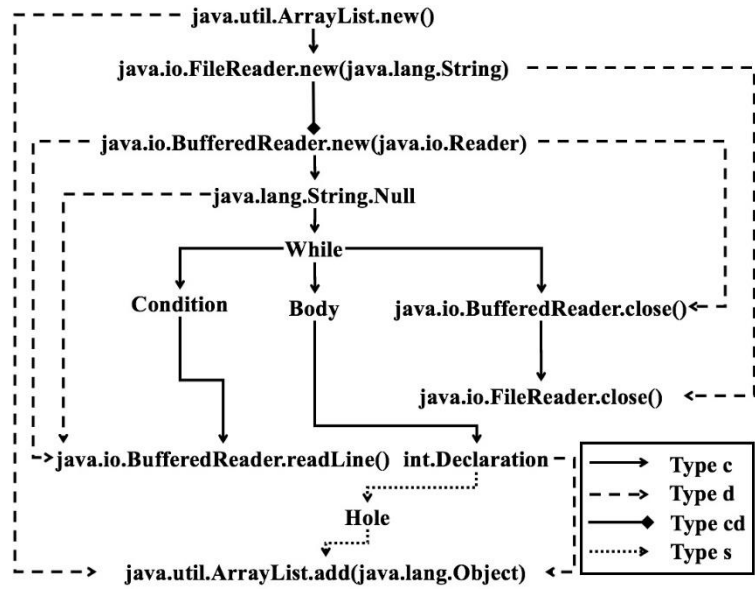
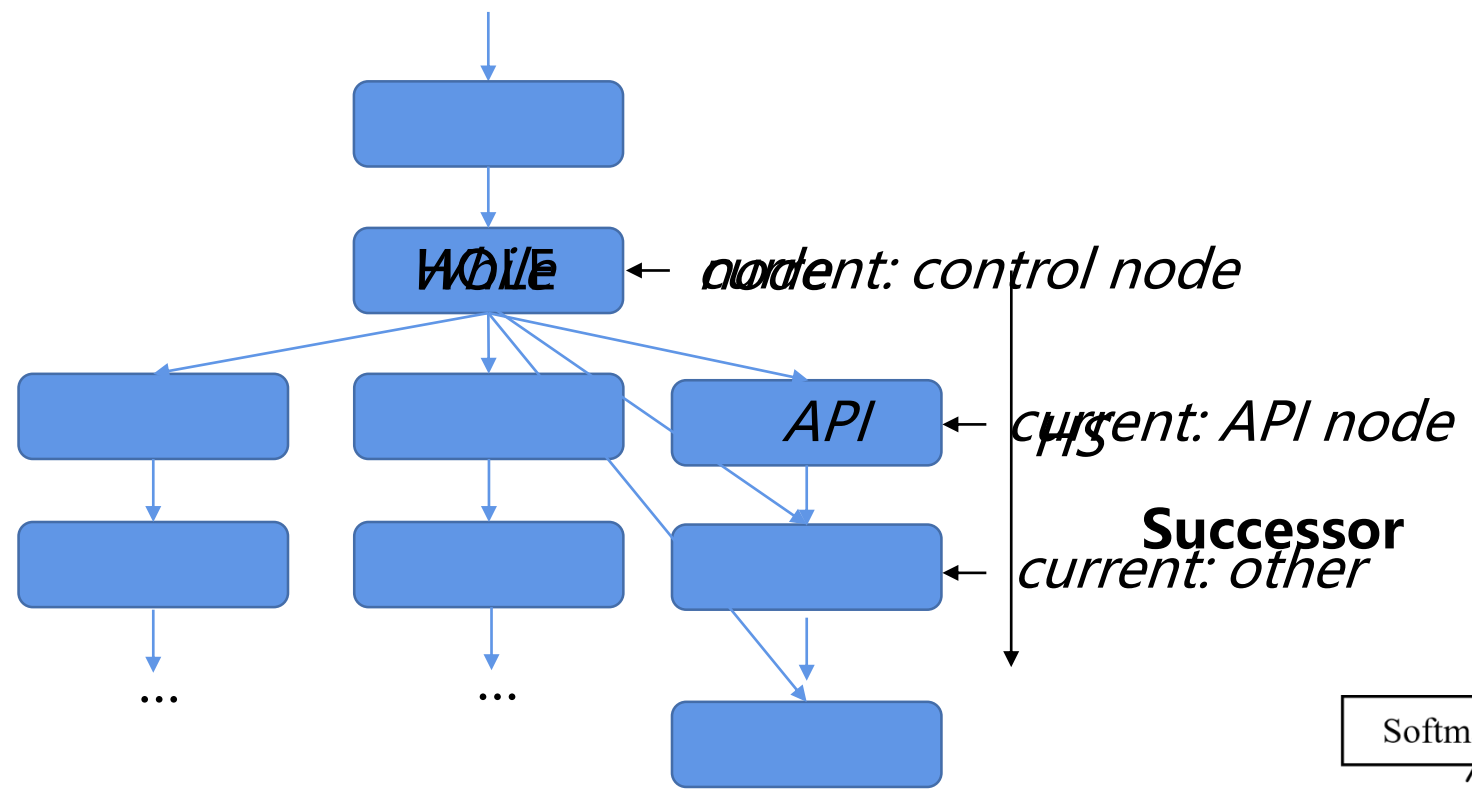
构建中

待实现



# APIRec-CST

MindSpore



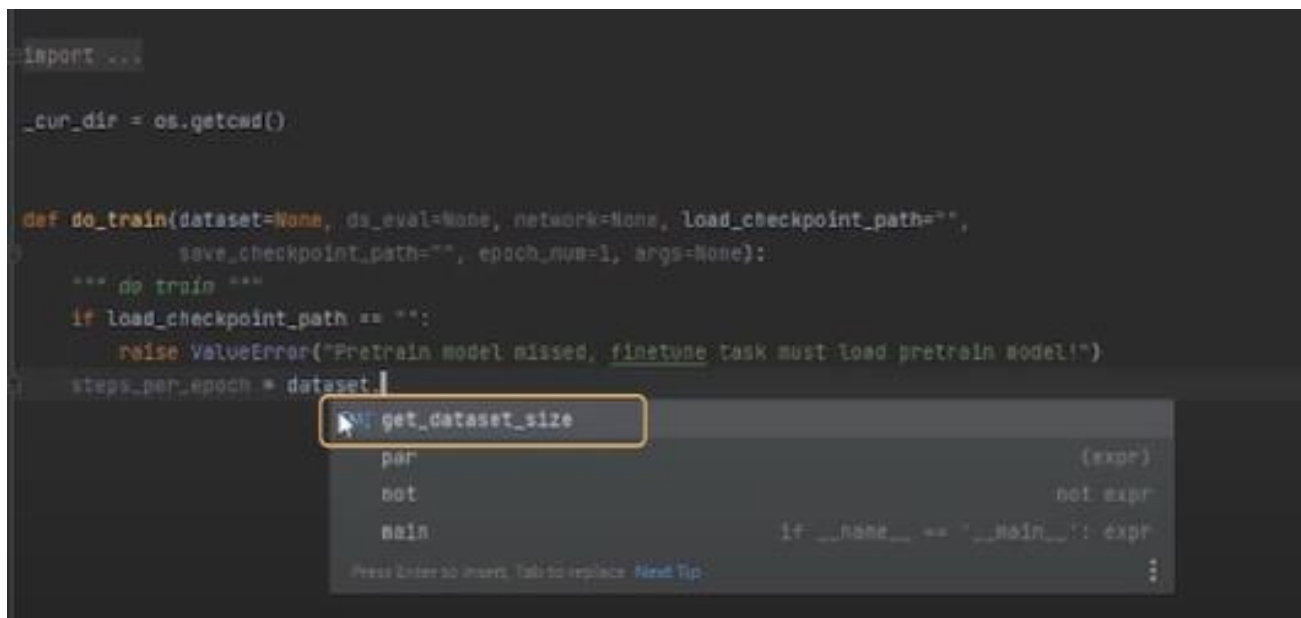


## AI补全与传统补全的区别

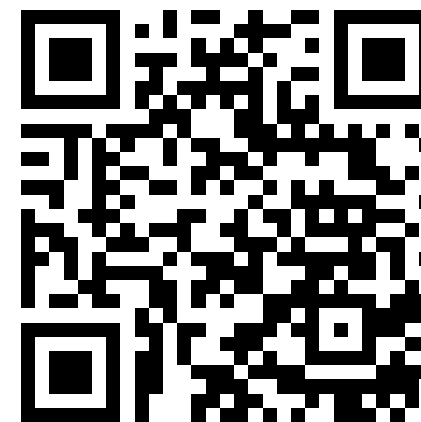
- 与传统IDE中的补全相比：传统IDE中的补全上下文不敏感，而AI补全上下文敏感。
- 与传统的基于模式挖掘的补全相比：传统的基于模式挖掘的补全需要显示挖掘得到相关模式，并且挖掘得到的模式的数量有限，而AI补全所有模式通过向量隐式表示，能够表示的模式数量非常多。
- 与传统的基于特征匹配的补全相比：传统的基于特征匹配的补全非常依赖于特征工程，如果特征工程做的不够好，那么会直接影响效果，而AI补全无需刻意做特征工程，特征可以通过模型进行学习。
- 与传统的基于统计模型的补全相比：传统的基于统计模型的补全的模型能力 < AI补全的模型能力。传统的基于统计模型的补全无法捕获长距离token之间的依赖关系。

## MindSpore Dev Toolkit

- 我们的补全技术会落到MindSpore Dev Toolkit欢迎大家来使用。目前补全可以降低MindSpore开发30%的键盘敲击次数
- 欢迎大家加入我们的MindSpore Dev Toolkit 开源社区，一起探讨AI For 软件工程



```
import ...  
  
_cur_dir = os.getcwd()  
  
def do_train(dataset=None, ds_eval=None, network=None, load_checkpoint_path="",  
             save_checkpoint_path="", epoch_num=1, args=None):  
    """ do train """  
    if load_checkpoint_path == "":  
        raise ValueError("Pretrain model missed, finetune task must load pretrain model!")  
    steps_per_epoch = dataset.  
    get_dataset_size  
    par (expr)  
    not not expr  
    main if __name__ == '__main__': expr  
    Press Enter to insert, Tab to replace. Next Tip
```



MindSpore Dev Toolkit开源社区  
<https://gitee.com/mindspore/ide-plugin>

THANK YOU