



GOTC 2023

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE, INTO THE FUTURE

大模型与软件开发自动化

李戈



北京大学

AIxcoder

2022年05月27日

报告提纲



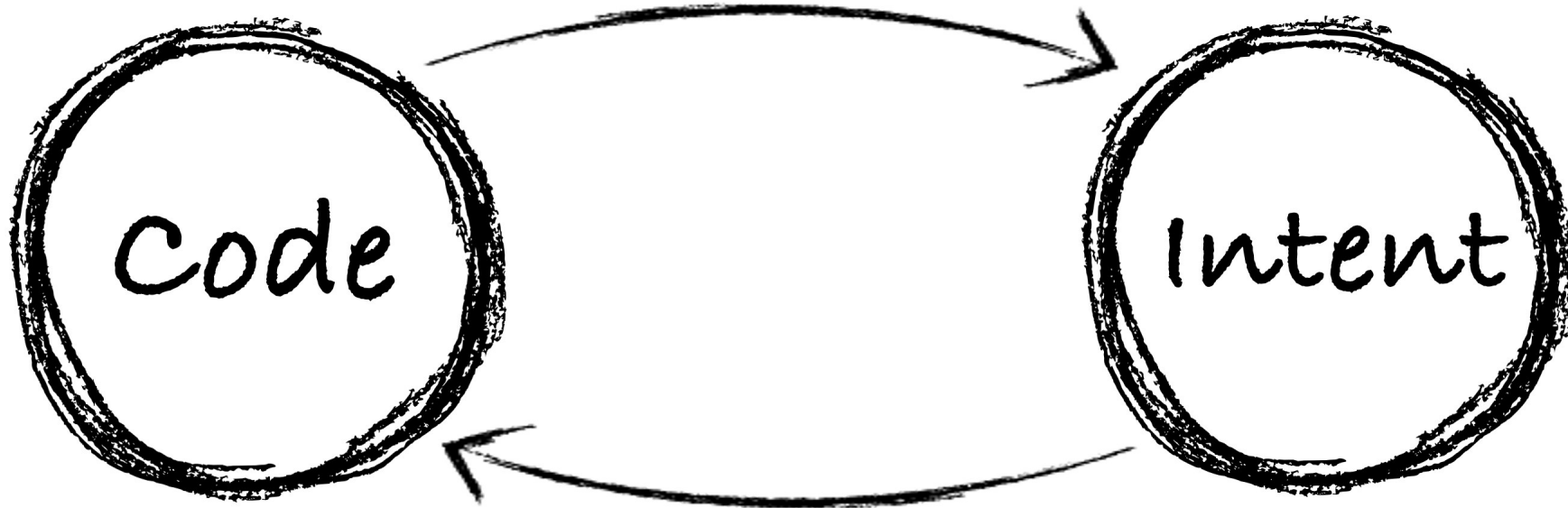
① 简要介绍我们的工作

② 大家可能关心的8个问题

My Research



Code Comprehension



Code Generation

The 1st Step...

Known as the 1st paper on DL based Program Representation



Computer Science > Software Engineering

[Submitted on 11 Sep 2014]

Building Program Vector Representations for Deep Learning

Lili Mou, Ge Li, Yuxuan Liu, Hao Peng, Zhi Jin, Yan Xu, Lu Zhang

Deep learning has made significant breakthroughs in various fields of artificial intelligence. Advantages of deep learning include the ability to capture highly complicated features, weak involvement of human engineering, etc. However, it is still virtually impossible to use deep learning to analyze programs since deep architectures cannot be trained effectively with pure back propagation. In this pioneering paper, we propose the "coding criterion" to build program vector representations, which are the premise of deep learning for program analysis. Our representation learning approach directly makes deep learning a reality in this new field. We evaluate the learned vector representations both qualitatively and quantitatively. We conclude, based on the experiments, the coding criterion is successful in building program representations. To evaluate whether deep learning is beneficial for program analysis, we feed the representations to deep neural networks, and achieve higher accuracy in the program classification task than "shallow" methods, such as logistic regression and the support vector machine. This result confirms the feasibility of deep learning to analyze programs. It also gives primary evidence of its success in this new field. We believe deep learning will become an outstanding technique for program analysis in the near future.

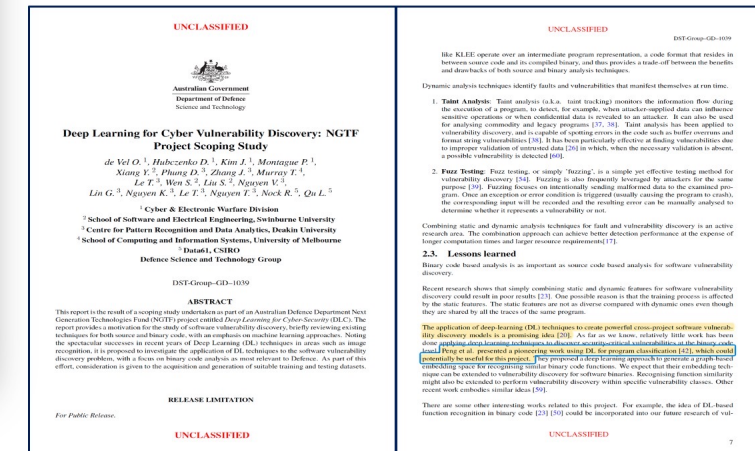
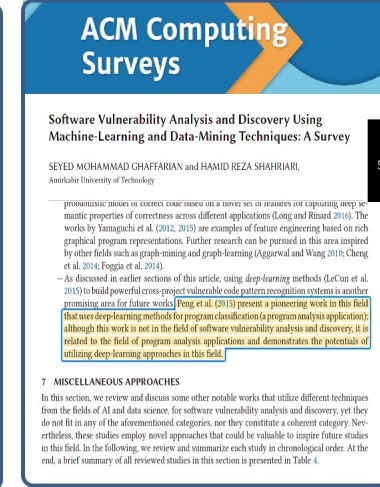
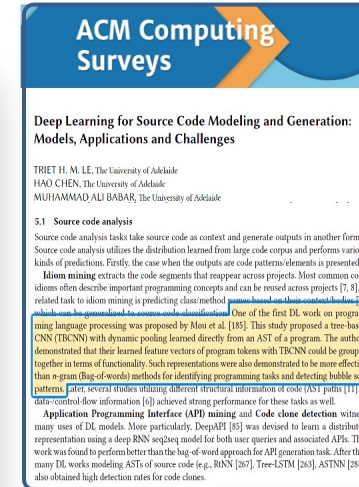
Comments: This paper was submitted to ICSE'14
Subjects: **Software Engineering (cs.SE)**; Machine Learning (cs.LG); Neural and Evolutionary Computing (cs.NE)
Cite as: **arXiv:1409.3358 [cs.SE]**
(or **arXiv:1409.3358v1 [cs.SE]** for this version)
<https://doi.org/10.48550/arXiv.1409.3358>

Submission history

From: Lili Mou [view email]
[v1] Thu, 11 Sep 2014 08:44:28 UTC (189 KB)

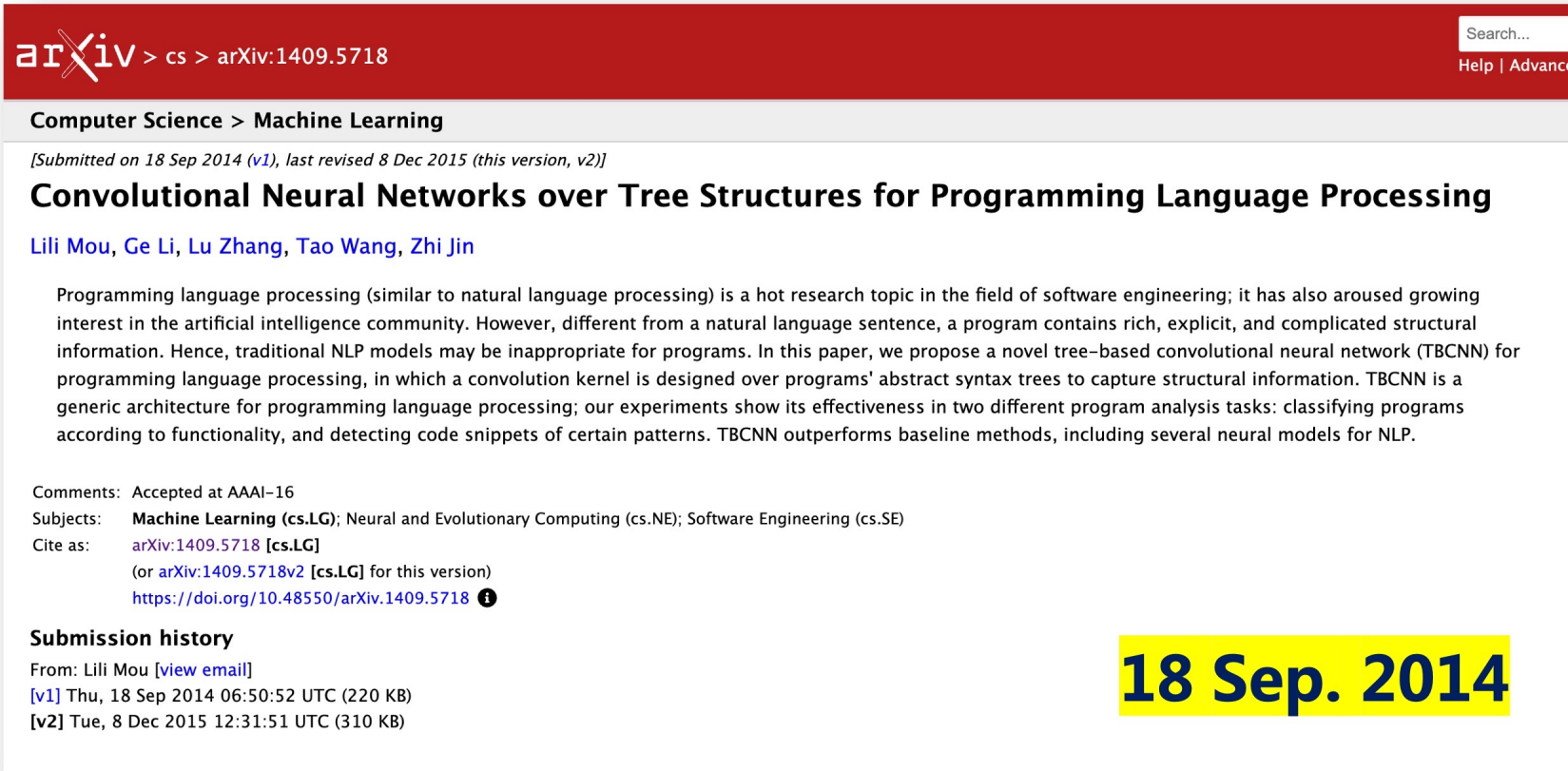
11 Sep. 2014

Lili Mou, Ge Li, Yuxuan Liu, Hao Peng, Zhi Jin, Yan Xu, Lu Zhang,
Building Program Vector Representations for Deep Learning. arXiv preprint arXiv:1409.3358, 2014.



The 1st Step...

Known as the 1st paper on DL based Program Representation



arXiv > cs > arXiv:1409.5718

Search...
Help | Advanced

Computer Science > Machine Learning

[Submitted on 18 Sep 2014 (v1), last revised 8 Dec 2015 (this version, v2)]

Convolutional Neural Networks over Tree Structures for Programming Language Processing

Lili Mou, Ge Li, Lu Zhang, Tao Wang, Zhi Jin

Programming language processing (similar to natural language processing) is a hot research topic in the field of software engineering; it has also aroused growing interest in the artificial intelligence community. However, different from a natural language sentence, a program contains rich, explicit, and complicated structural information. Hence, traditional NLP models may be inappropriate for programs. In this paper, we propose a novel tree-based convolutional neural network (TBCNN) for programming language processing, in which a convolution kernel is designed over programs' abstract syntax trees to capture structural information. TBCNN is a generic architecture for programming language processing; our experiments show its effectiveness in two different program analysis tasks: classifying programs according to functionality, and detecting code snippets of certain patterns. TBCNN outperforms baseline methods, including several neural models for NLP.

Comments: Accepted at AAAI-16

Subjects: **Machine Learning (cs.LG)**; Neural and Evolutionary Computing (cs.NE); Software Engineering (cs.SE)

Cite as: arXiv:1409.5718 [cs.LG]
(or arXiv:1409.5718v2 [cs.LG] for this version)
<https://doi.org/10.48550/arXiv.1409.5718>

Submission history

From: Lili Mou [view email]

[v1] Thu, 18 Sep 2014 06:50:52 UTC (220 KB)

[v2] Tue, 8 Dec 2015 12:31:51 UTC (310 KB)

18 Sep. 2014

Lili Mou, Ge Li, Lu Zhang, Tao Wang, Zhi Jin, Convolutional Neural Networks over Tree Structures for Programming Language Processing, arXiv preprint arXiv:1409.5718, 2014.

ACM Computing Surveys

Deep Learning for Source Code Modeling and Generation: Models, Applications and Challenges

TRIET H. M. LE, The University of Adelaide
HAO CHEN, The University of Adelaide
MUHAMMAD ALI BABAR, The University of Adelaide

5.1 Source code analysis

Source code analysis tasks take source code as context and generate outputs in another format. Source code analysis utilizes the distribution learned from large code corpus and performs various kinds of predictions. Firstly, the case when the outputs are code patterns/elements is presented.

Idiom mining extracts the code segments that reappear across projects. Most common code idioms often describe important programming concepts and can be reused across projects [7, 8]. A related task to idiom mining is predicting class/method names based on their context/bodies [4], which can be generalized to source code classification. One of the first DL work on programming language processing was proposed by Mou et al. [185]. This study proposed a tree-based CNN (TBCNN) with dynamic pooling learned directly from an AST of a program. The authors demonstrated that their learned feature vectors of program tokens with TBCNN could be grouped together in terms of functionality. Such representations were also demonstrated to be more effective than n -gram (Bag-of-words) methods for identifying programming tasks and detecting bubble sort patterns. Later, several studies utilizing different structural information of code (AST paths [11] or data-/control-flow information [6]) achieved strong performance for these tasks as well.

Application Programming Interface (API) mining and **Code clone detection** witness many uses of DL models. More particularly, DeepAPI [85] was devised to learn a distributed representation using a deep RNN seq2seq model for both user queries and associated APIs. This work was found to perform better than the bag-of-words approach for API generation task. After that, many DL works modeling ASTs of source code (e.g., RtNN [267], Tree-LSTM [263], ASTNN [284]) also obtained high detection rates for code clones.

The first person to eat crabs is not easy.



At that time,

No TensorFlow, No Pytorch, or even No Theano, No GPU.

We have to build everything from scratch.

The 1st Step...

Known as the 1st paper on DL based Code Generation

arXiv > cs > arXiv:1510.07211

Search...

Help | Advanced

Computer Science > Software Engineering

[Submitted on 25 Oct 2015]

On End-to-End Program Generation from User Intention by Deep Neural Networks

Lili Mou, Rui Men, Ge Li, Lu Zhang, Zhi Jin

This paper envisions an end-to-end program generation scenario using recurrent neural networks (RNNs): Users can express their intention in natural language, and the RNN then automatically generates corresponding code in a character-by-character fashion. We demonstrate its feasibility through empirical analysis. To fully make such technique useful in practice, we also point out several cross-disciplinary challenges, including model architecture, training data, and model evaluation. Although much long-term research shall be addressed in this new field, we believe end-to-end program generation is a reality in future decades, and we are looking forward to its practice.

Comments: Submitted to 2016 International Conference of Software Engineering "Vision of 2025 and Beyond" track

Subjects: **Software Engineering (cs.SE)**; Machine Learning (cs.LG)

Cite as: arXiv:1510.07211 [cs.SE]

(or arXiv:1510.07211v1 [cs.SE] for this version)

<https://doi.org/10.48550/arXiv.1510.07211>

Submission history

From: Lili Mou [view email]

[v1] Sun, 25 Oct 2015 06:52:45 UTC (217 KB)

25 Oct. 2015

```
#include<stdio.h>
void main()
{
    (1)n
    int a[100],i,max1,max2;
    scanf("%d",&n);
    for(i=0;i<=n-1;i++)
    {
        scanf("%d",&a[i]);
        if (a[i]>max1)
            max2=a[i];
        for(i=1;i<=n;i++)
        {
            (2)2 (3)<
            if(a[i]>max&&a[i]<max1)
                max2=a[i];
        }
        printf("%d\n%d",max1,max2);
        return (4) idell
    }
}
```

(a) Generated code

```
#include<stdio.h>
int main(){
    int n,i,j,sz[100],max=0,ci=0;
    scanf("%d",&n);
    for(i=0;i<n;i++){
        scanf("%d",&sz[i]);
        if(sz[i]>max){
            max=sz[i];
        }
        for(i=0;i<n;i++){
            if(sz[i]>ci&&sz[i]<max){
                ci=sz[i];
            }
        }
        printf("%d\n%d",max,ci);
    }
}
```

(b) Training sample 1

```
#include<stdio.h>
void main()
{
    int n,i,a[100],j,max1,max2;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    max1=a[0];
    for(i=0;i<n;i++)
    {
        if(a[i]>max1)
            max1=a[i];
    }
    for(i=0;i<n;i++)
    {
        if(max1==a[i])
            j=i;
    }
    if(max1!=a[0])
        max2=a[0];
    else max2=a[j];
    for(i=0;i<n;i++)
    {
        if(i==j) continue;
        if(a[i]>max2)
            max2=a[i];
    }
    printf("%d\n%d",max1,max2);
}
```

(c) Training sample 2

Figure 2: (a) Code generated by RNN. The code is almost correct except 4 wrong characters (among ~280 characters in total), highlighted in the figure. (b) Code with the most similar structure in the training set, detected by cfinder. (c) Code with the most similar identifiers in the training set, also detected by cfinder. Note that (1) the identifier "x" should be "n"; (2) "max" should be "max2"; (3) "=" should be "<"; (4) return type should be void.

Lili Mou, Rui Men, Ge Li, Lu Zhang, Zhi Jin, On End-to-End Program Generation from User Intention by Deep Neural Networks, arXiv preprint, arXiv:1510.07211, 2015.

The 1st Step...



Known as the 1st Tool on DL based Code Generation

```
import sys

class Model():
    def __init__(self, args, infer = False):
        self.args = args
        if infer :
            args.batch_size = 1
            args.seq_length = 1
        if args.model == 'rnn':
            cell_fn = rnn_cell.BasicRNNCell
        elif args.model == 'gru':
            cell_fn = rnn_cell.GRUCell
        elif args . model == <str> : cell_fn = <UNK> .
```

16 May 2017



0:41.08



The 1st Step...



12 Jun. 2017

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
intellij-plugin | src | com | seke | autocomplete | config | ConfigState.java
ConfigState.java
1 package com.seke.autocomplete.config;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 /**
17  * Created by oc on 2017/11/27.
18  */
19 @State(name = "ConfigState", storages = {@Storage("autoCompleteConfig.xml")})
20 public class ConfigState implements PersistentStateComponent<ConfigState> {
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
test | src | test.py
test.py
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

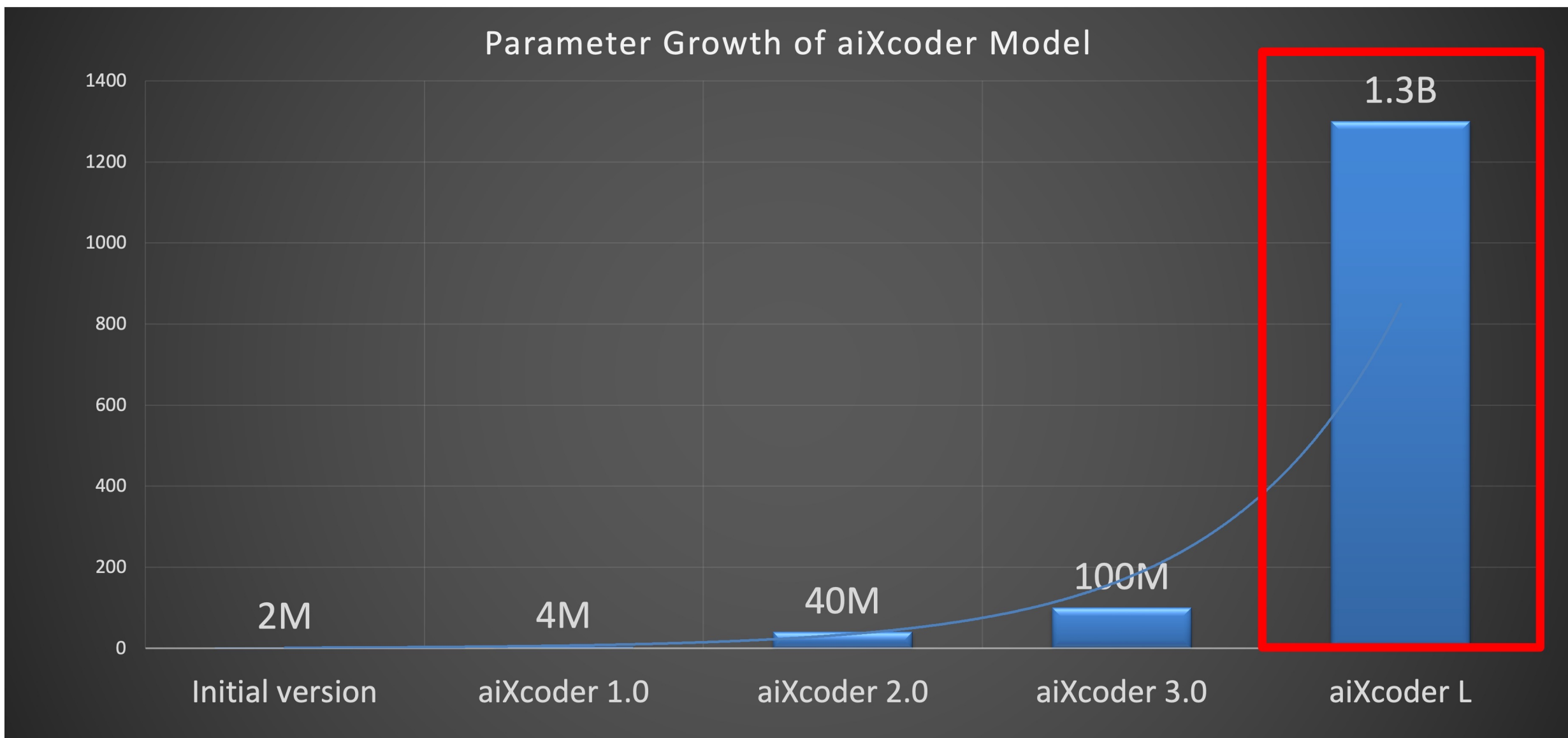
>78%

AIXcoder 1.0

```
Run | Plugin
Console | idea.log
D:\intellij\bin\idea.properties: idea.config.path already defined
D:\IntelliJ\bin\idea.properties: 'idea.system.path' already defined
十二月 26, 2017 6:36:57 下午 java.util.prefs.WindowsPreferences <init>
WARNING: Could not open/create prefs root node Software\JavaSoft\Prefs at root 0x80000002. Windows Registry does not have this path.
Removing listener...
Adding listener...
All files are up-to-date (3 minutes ago)
```

Platform and Plugin Updates
IntelliJ IDEA is ready to update.

Model capabilities vs. Number of parameters



aiXcoder Full Line Code Completion

```
private static void compressFile(File file, ZipOutputStream zos, String baseDir) {  
    if (!file.exists()) {  
        return;  
    }  
}
```

15 Jul. 2020

aiXcoder L

aiXcoder

MacBook Pro

AIxcoder Full Line Code Completion

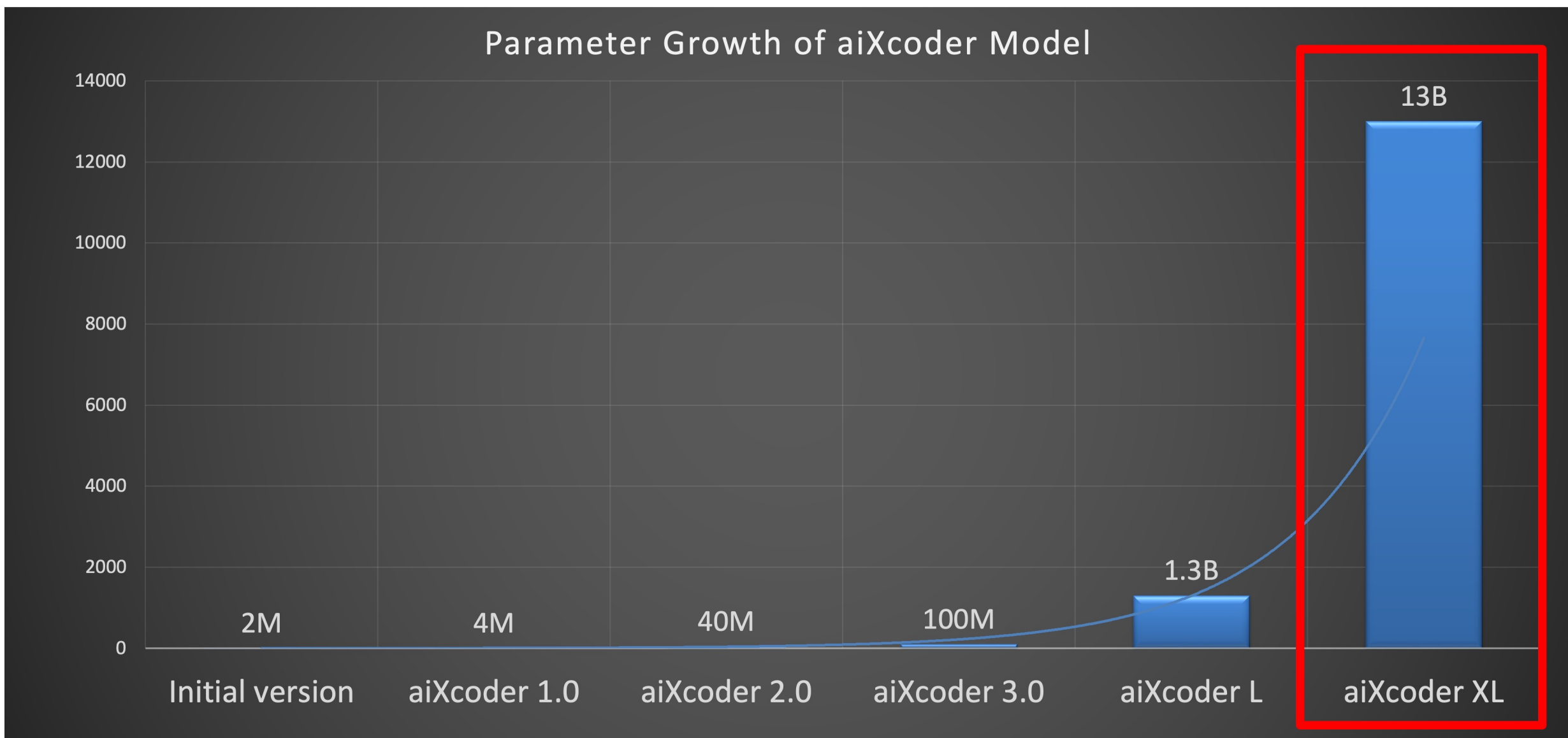
```
public class LogUtils {  
    public static void loadFileConfig(String configFilePath) throws IOException, JoranException {  
        LoggerContext loggerContext = (LoggerContext) LoggerFactory.getILoggerFactory();  
        File externalConfigFile = new File(configFilePath);  
        |  
    }  
}
```

15 Jul. 2020

AIxcoder

MacBook Pro

Model capabilities vs. Number of parameters



aiXcoder XL

支持方法级代码生成

根据自然语言功能描述，生成完整程序代码



对标 Copilot, 国内首个: 自然语言
一键生成方法级代码 aiXcoder XL...

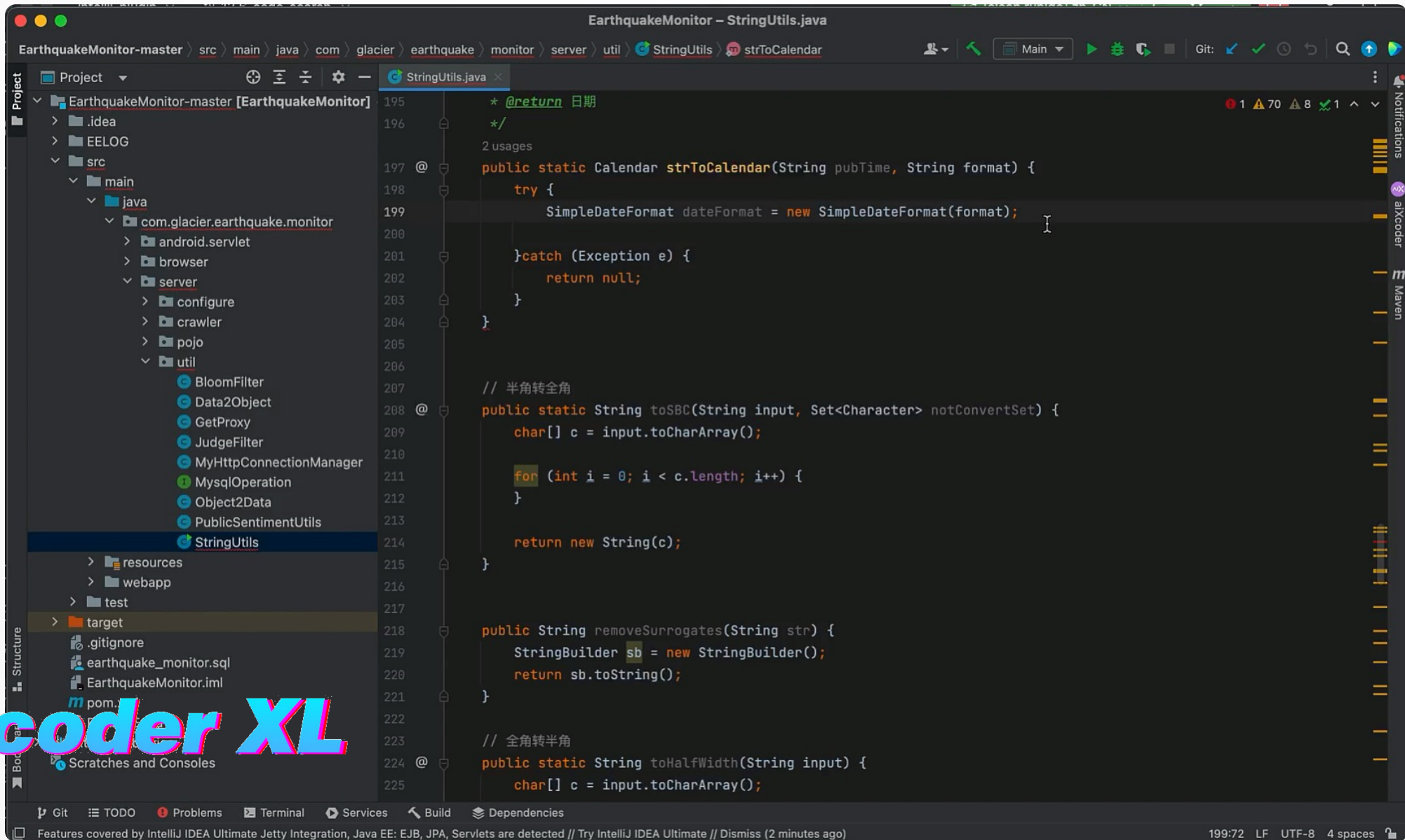
国内首个能够根据开发者给出的
“自然语言描述”一键生成“完整
方法级代码”的智能编程应...



22 Jun. 2022

机器之心

aiXcoder 多行代码自动补全



aiXcoder XL

aiXcoder 开发框架代码自动补全

The screenshot shows an IDE window titled "web.core - DemoService.java". The code in the editor is as follows:

```
1 package com.ld.web.core.modules.demo.service;
2
3
4 import ...
5
6 @Service
7 public class DemoService {
8
9     @Autowired
10    JdbcTemplate jdbcTemplate;
11
12    public Demo findById(int id){
13        return jdbcTemplate.queryForObject( sql: "SELECT * FROM demo WHERE id = ?", new Object[] {id}, new BeanPropertyRowMapper<Demo>());
14    }
15 }
16
17
18
19
20
21
22
23
```

The IDE interface includes a file explorer on the left showing the project structure, a top toolbar with application and Git icons, and a bottom status bar with build and terminal information.

aiXcoder XL

为节省时间，对视频做2倍加速

aiXcoder Full Method Code Completion

The screenshot displays an IDE window for a project named "litchi-master". The main editor shows the file "HttpServerHandler.java" with the following code:

```
100     }
101   }
102
103   /**
104    *
105    */
106
107
108
109
110
111   private void scanRouters() {
112     for (Class<? extends HttpController> clazz : controllerFactory.getControllers()) {
113       addRoute(clazz);
114     }
115     LOGGER.info("scan controller is completed! router size:{}", routers.size());
116   }
117
118   @Override
119   @RequestMapping
120   public void channelRead0(ChannelHandlerContext ctx, FullHttpRequest req) {
121     if (req.uri().equals("/favicon.ico")) {
122       writeHttpResponse(ctx, HttpStatus.NOT_FOUND);
123       return;
124     }
125     runController(ctx, req);
126   }
127
128   @Override
129   public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws Exception {
130     if (cause instanceof IOException) {
131       //LOGGER.error("{}", cause.getMessage());
132     } else {
133       LOGGER.error("{}", cause);
134     }
135   }
136 }
```

The IDE interface includes a sidebar on the left with "Project" and "Structure" views, and a sidebar on the right with "aiXcoder" branding, a "云端服务" (Cloud Service) toggle, and a "打开代码搜索" (Open Code Search) button. The bottom status bar shows "正在初始化: 启动大概需要3-15分钟请耐心等待 // 确定 (26 minutes ago)" and "104:8 LF UTF-8 4 spaces".

AIxcoder Full Method Code Completion

The screenshot shows an IDE window titled "litchi-master - HttpServerHandler.java [litchi-master.main]". The code editor displays the following Java code:

```
100     }
101   }
102
103   /**
104   *
105   */
106
107
108
109
110
111
112   private void scanRouters() {
113     for (Class<? extends HttpController> clazz : controllerFactory.getControllers()) {
114       addRoute(clazz);
115     }
116     LOGGER.info("scan controller is completed! router size:{}", routers.size());
117   }
118
119   @Override
120   @RequestMapping("/favicon.ico")
121   public void channelRead0(ChannelHandlerContext ctx, FullHttpRequest req) {
122     if (req.uri().equals("/favicon.ico")) {
123       writeHttpResponse(ctx, HttpStatus.NOT_FOUND);
124       return;
125     }
126     runController(ctx, req);
127   }
128
129   @Override
130   public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws Exception {
131     if (cause instanceof IOException) {
132       //LOGGER.error("{} ", cause.getMessage());
133     } else {
134       //LOGGER.error("{} ", cause);
135     }
136   }
137 }
```

The AIxcoder sidebar on the right includes a "云端服务" (Cloud Service) toggle, the AIxcoder logo, a "打开代码搜索" (Open Code Search) button, and links for "设置" (Settings), "帮助" (Help), and "关于" (About). The bottom status bar shows "正在初始化: 启动大概需要3-15分钟请耐心等待 // 确定 (37 minutes ago)" and the page number "104:8".

What we're trying to do



Growth of aiXcoder Model





问题1：大模型的代码生成能力到什么程度？

- 如果开发者能把需求描述得足够清楚，大模型似乎都能生成可用代码。



问题2：大模型的“能力”哪里来？

什么是大模型？



LLM = Large Language Model

- “大模型”是个俗称，如果您理解的大模型是类似于ChatGPT的模型，那么：
 - ◆ 错误认知一：参数量“非常大”的模型就叫“大模型”
 - ◆ 错误认知二：不依赖自然语言也可以做大模型



Table 1: Summary of PFMs in Text. The pretraining task includes language model (LM), masked LM (MLM), permuted LM (PLM), denoising autoencoder (DAE), knowledge graphs (KG), and knowledge embedding (KE).

Year	Conference	Model	Architecture	Embedding	Training method	Code
2013	NeurIPS	Skip-Gram [66]	Word2Vec	Probabilistic	-	https://github.com/.../models
2014	EMNLP	GloVe [67]	Word2Vec	Probabilistic	-	-
2015	NeurIPS	LM-LSTM [68]	LSTM	Probabilistic	LM	https://github.com/.../GloVe
2016	IJCAI	Shared LSTM [69]	LSTM	Probabilistic	LM	https://github.com/.../adversarial_text
2017	TACL	FastText [70]	Word2Vec	Probabilistic	-	https://github.com/.../fastText
2017	NeurIPS	CoVe [71]	LSTM+Seq2Seq	Probabilistic	-	https://github.com/.../cove
2018	NAACL-HLT	ELMO [51]	LSTM	Contextual	LM	https://allenlp.org/elmo
2018	NAACL-HLT	BERT [13]	Transformer Encoder	Contextual	MLM	https://github.com/.../bert
2018		OpenAI GPT [48]	Transformer Decoder	Autoregressive	LM	https://github.com/.../transformer-lm
2019	ACL	ERNIE(THU)	Transformer Encoder	Contextual	MLM	https://github.com/.../ERNIE
2019	ACL	Transformer-XL [72]	Transformer-XL	Contextual	-	https://github.com/.../transformer-xl
2019	ICLR	InfoWord [73]	Transformer Encoder	Contextual	MLM	-
2019	ICLR	StructBERT [74]	Transformer Encoder	Contextual	MLM	-
2019	ICLR	ALBERT [45]	Transformer Encoder	Contextual	MLM	https://github.com/.../ALBERT
2019	ICLR	WKLm [75]	Transformer Encoder	Contextual	MLM	-
2019	ICML	MASS [57]	Transformer	Contextual	MLM(Seq2Seq)	https://github.com/.../MASS
2019	EMNLP-IJCNLP	KnowBERT [76]	Transformer Encoder	Contextual	MLM	https://github.com/.../kb
2019	EMNLP-IJCNLP	Unicoder [77]	Transformer Encoder	Contextual	MLM+TLM	-
2019	EMNLP-IJCNLP	MultiFit [78]	QRNN	Probabilistic	LM	https://github.com/.../multifit
2019	EMNLP-IJCNLP	SciBERT [79]	Transformer Encoder	Contextual	MLM	https://github.com/.../scibert
2019	EMNLP-IJCNLP	BERT-PKD [80]	Transformer Encoder	Contextual	MLM	https://github.com/.../Compression
2019	NeurIPS	Xlnet [14]	Transformer-XL Encoder	Permutation	PLM	https://github.com/.../xlnet
2019	NeurIPS	UNILM [58]	LSTM + Transformer	Contextual	LM + MLM	https://github.com/.../unilm
2019	NeurIPS	XLM [81]	Transformer Encoder	Contextual	MLM+CLM+TLM	https://github.com/.../XLM
2019	OpenAI Blog	GPT-2 [49]	Transformer Decoder	Autoregressive	LM	https://github.com/.../gpt-2
2019	arXiv	RoBERTa [53]	Transformer Encoder	Contextual	MLM	https://github.com/.../fairseq
2019	arXiv	ERNIE(Baidu) [59]	Transformer Encoder	Contextual	MLM+DLM	https://github.com/.../ERNIE
2019	EMC2@NeurIPS	QSBERT [82]	Transformer Encoder	Contextual	MLM	https://github.com/.../quantized_bert.py
2019	arXiv	DistilBERT [83]	Transformer Encoder	Contextual	MLM	https://github.com/.../distillation
2020	ACL	fastBERT [84]	Transformer Encoder	Contextual	MLM	https://github.com/.../FastBERT
2020	ACL	SpanBERT [42]	Transformer Encoder	Contextual	MLM	https://github.com/.../SpanBERT
2020	ACL	BART [43]	Transformer	En: Contextual De: Autoregressive	DAE	https://github.com/.../transformers
2020	ACL	CamemBERT [85]	Transformer Encoder	Contextual	MLM(WWM)	https://camembert-model.fr
2020	ACL	XLM-R [86]	Transformer Encoder	Contextual	MLM	https://github.com/.../XLM
2020	ICLR	Reformer [87]	Reformer	Permutation	-	https://github.com/.../reformer
2020	ICLR	ELECTRA [44]	Transformer Encoder	Contextual	MLM	https://github.com/.../electra
2020	AAAI	Q-BERT [88]	Transformer Encoder	Contextual	MLM	-
2020	AAAI	XNLG [89]	Transformer	Contextual	MLM+DAE	https://github.com/.../xnlng
2020	AAAI	K-BERT [90]	Transformer Encoder	Contextual	MLM	https://github.com/.../K-BERT
2020	AAAI	ERNIE 2.0 [60]	Transformer Encoder	Contextual	MLM	https://github.com/.../ERNIE
2020	NeurIPS	GPT-3 [20]	Transformer Decoder	Autoregressive	LM	https://github.com/.../gpt-3
2020	NeurIPS	MPNet [55]	Transformer Encoder	Permutation	MLM+PLM	https://github.com/.../MPNet
2020	NeurIPS	ConvBERT [91]	Mixed Attention	Contextual	-	https://github.com/.../ConvBert
2020	NeurIPS	MiniLM [92]	Transformer Encoder	Contextual	MLM	https://github.com/.../minilm
2020	TACL	mBART [93]	Transformer	Contextual	DAE	https://github.com/.../mbart
2020	COLING	CoLAKE [94]	Transformer Encoder	Contextual	MLM+KE	https://github.com/.../CoLAKE
2020	LREC	FlauBERT [95]	Transformer Encoder	Contextual	MLM	https://github.com/.../Flaubert
2020	EMNLP	GLM [96]	Transformer Encoder	Contextual	MLM+KG	https://github.com/.../GLM
2020	EMNLP (Findings)	TinyBERT [97]	Transformer	Contextual	MLM	https://github.com/.../TinyBERT
2020	EMNLP (Findings)	RobBERT [98]	Transformer Encoder	Contextual	MLM	https://github.com/.../RobBERT
2020	EMNLP (Findings)	ZEN [62]	Transformer Encoder	Contextual	MLM	https://github.com/.../ZEN
2020	EMNLP (Findings)	BERT-MK [99]	KG-Transformer Encoder	Contextual	MLM	-
2020	ReplANLP@ACL	CompressingBERT [33]	Transformer Encoder	Contextual	MLM(Pruning)	https://github.com/.../bert-prune
2020	JMLR	T5 [100]	Transformer	Contextual	MLM(Seq2Seq)	https://github.com/.../transformer
2021	T-ASL	BERT-wwm-Chinese [61]	Transformer Encoder	Contextual	MLM	https://github.com/.../BERT-wwm
2021	EACL	PET [101]	Transformer Encoder	Contextual	MLM	https://github.com/.../pet
2021	TACL	KEPLER [102]	Transformer Encoder	Contextual	MLM+KE	https://github.com/.../KEPLER
2021	EMNLP	SimCSE [103]	Transformer Encoder	Contextual	MLM+KE	https://github.com/.../SimCSE
2021	ICML	GLaM [104]	Transformer	Autoregressive	LM	-
2021	arXiv	XLM-E [105]	Transformer	Contextual	MLM	-
2021	arXiv	TO [106]	Transformer	Contextual	MLM	https://github.com/.../TO
2021	arXiv	Gopher [107]	Transformer	Autoregressive	LM	-
2022	arXiv	MT-NLG [108]	Transformer	Contextual	MLM	-
2022	arXiv	LaMDA [65]	Transformer Decoder	Autoregressive	LM	https://github.com/.../LaMDA
2022	arXiv	Chinchilla [109]	Transformer	Autoregressive	LM	-
2022	arXiv	PaLM [41]	Transformer	Autoregressive	LM	https://github.com/.../PaLM
2022	arXiv	OPT [110]	Transformer Decoder	Autoregressive	LM	https://github.com/.../MetaSeq

语言是人类思维的工具。

—— 记忆中来自马克思

N. Mercer, Words and minds: How we use language to think together. Psychology Press, 2000.



问题3：训练大模型需要多大代价？

Training Cost of Large Language Models



Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)	Flops per param per token	Mult for bwd pass	Fwd-pass flops per active param per token	Frac of params active for each token
T5-Small	2.08E+00	1.80E+20	60	1,000	3	3	1	0.5
T5-Base	7.64E+00	6.60E+20	220	1,000	3	3	1	0.5
T5-Large	2.67E+01	2.31E+21	770	1,000	3	3	1	0.5
T5-3B	1.04E+02	9.00E+21	3,000	1,000	3	3	1	0.5
T5-11B	3.82E+02	3.30E+22	11,000	1,000	3	3	1	0.5
BERT-Base	1.89E+00	1.64E+20	109	250	6	3	2	1.0
BERT-Large	6.16E+00	5.33E+20	355	250	6	3	2	1.0
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000	6	3	2	1.0
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000	6	3	2	1.0
GPT-3 Small	2.60E+00	2.25E+20	125	300	6	3	2	1.0
GPT-3 Medium	7.42E+00	6.41E+20	356	300	6	3	2	1.0
GPT-3 Large	1.58E+01	1.37E+21	760	300	6	3	2	1.0
GPT-3 XL	2.75E+01	2.38E+21	1,320	300	6	3	2	1.0
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300	6	3	2	1.0
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300	6	3	2	1.0
GPT-3 13B	2.68E+02	2.31E+22	12,850	300	6	3	2	1.0
GPT-3 175B	3.64E+03	3.14E+23	174,600	300	6	3	2	1.0

Table D.1: Starting from the right hand side and moving left, we begin with the number of training tokens that each model was trained with. Next we note that since T5 uses an encoder-decoder model, only half of the parameters are active for each token during a forward or backwards pass. We then note that each token is involved in a single addition and a single multiply for each active parameter in the forward pass (ignoring attention). Then we add a multiplier of 3x to account for the backwards pass (as computing both $\frac{\partial params}{\partial loss}$ and $\frac{\partial acts}{\partial loss}$ use a similar amount of compute as the forwards pass). Combining the previous two numbers, we get the total flops per parameter per token. We multiply this value by the total training tokens and the total parameters to yield the number of total flops used during training. We report both flops and petaflop/s-day (each of which are $2.88e+7$ flops).

The 175B GPT-3 model trained by OpenAI required 14.8 days [17] of compute on 10,000 V100s, and consumed $3.14+23$ FLOPs.

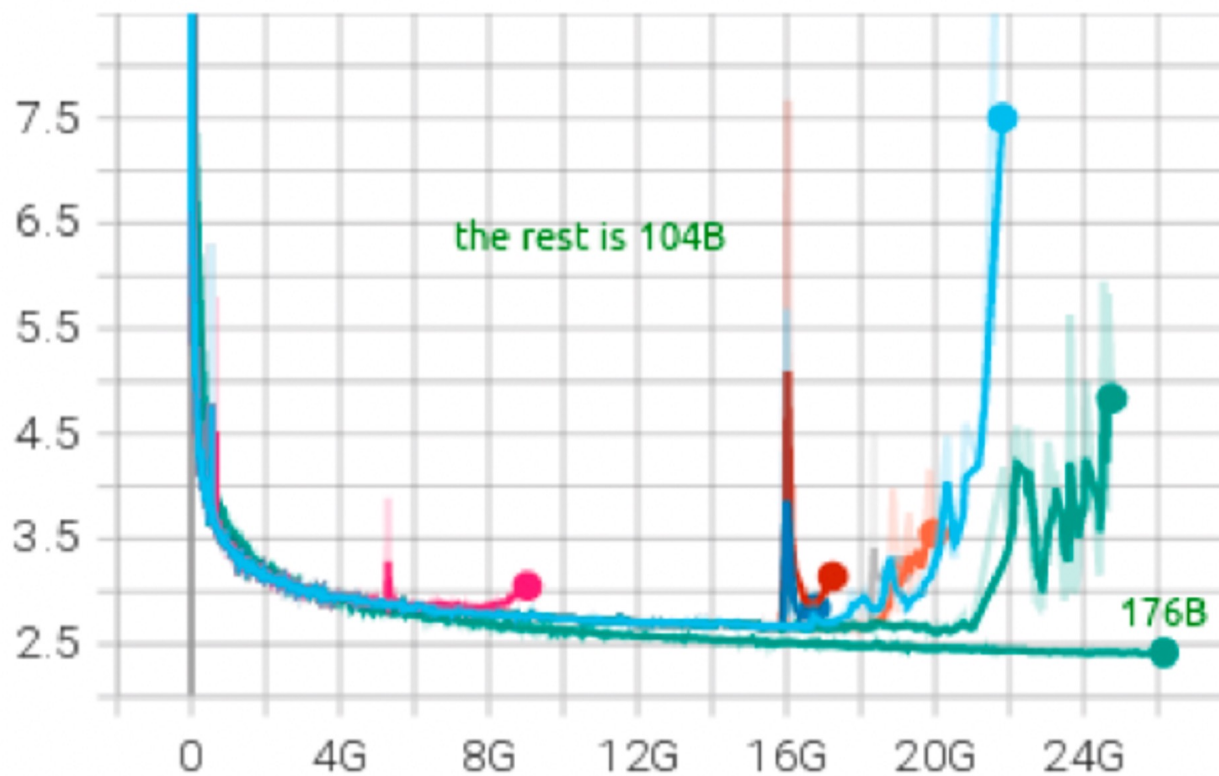


That's just the cost of
taste one time of success!

Multiple tests before success are necessary



lm-loss-training/lm loss vs tokens
tag: lm-loss-training/lm loss vs tokens



(b) BLOOM 176B's experiments

Multiple tests before success are necessary

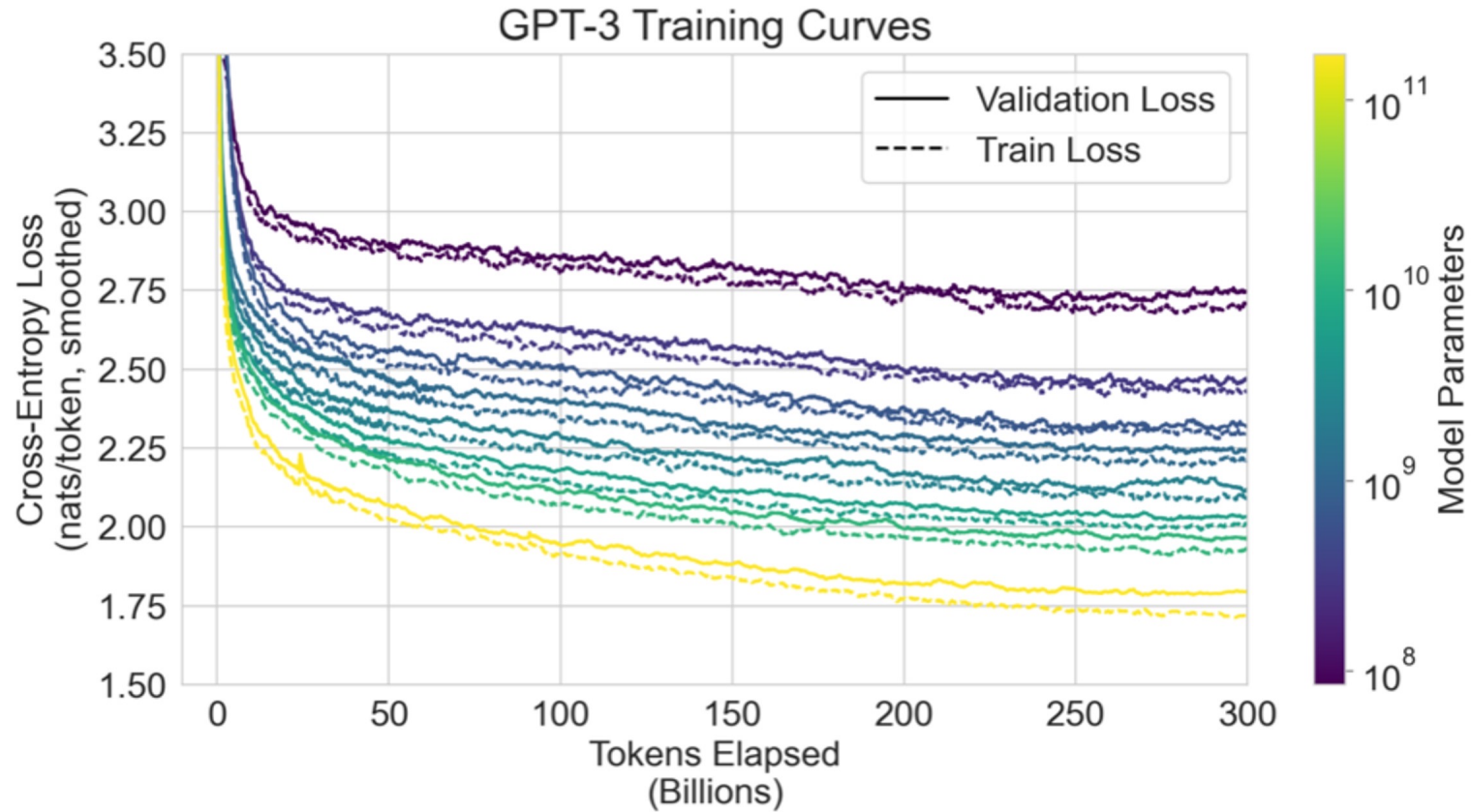
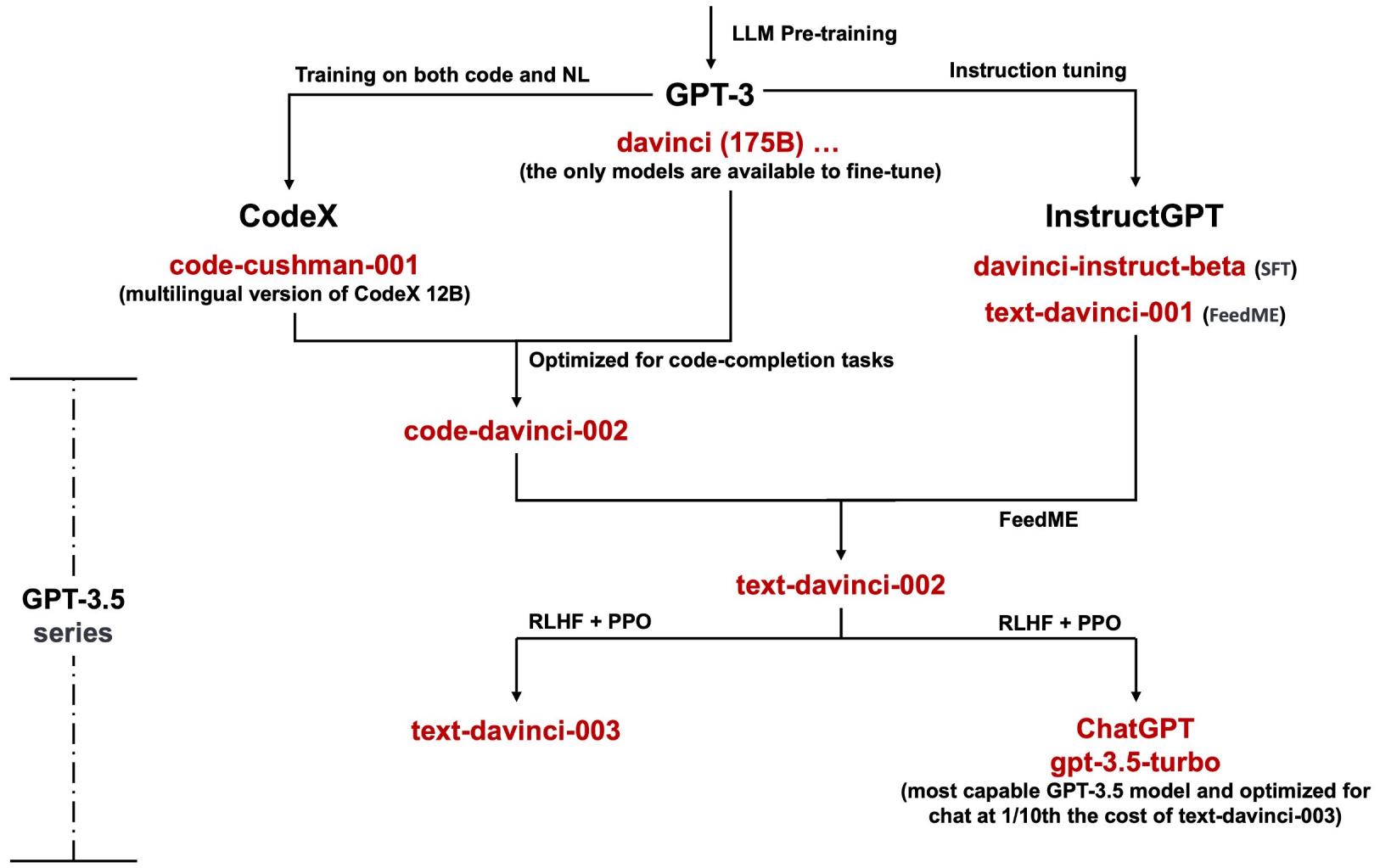


Figure 4.1: GPT-3 Training Curves We measure model performance during training on a deduplicated validation split of our training distribution. Though there is some gap between training and validation performance, the gap grows only minimally with model size and training time, suggesting that most of the gap comes from a difference in difficulty rather than overfitting.



问题3：为什么代码对LLM的训练有帮助？



GPT-3.5 series

SFT Supervised fine-tuning on human demonstrations
FeedME Supervised fine-tuning on human-written demonstrations and on model samples rated 7/7 by human labelers on an overall quality score
RLHF Reinforcement Learning from Human Feedback
PPO Reinforcement learning with reward models trained from comparisons by humans

为什么代码对LLM的训练有帮助？



(1) 程序语言具有自然性

- ◆ 程序语言具有与自然语言相似的统计特性，所以，可以利用语言模型进行建模，只是这种建模方式并不高效。

Naturalness: Software is a form of human communication; software corpora have similar statistical properties to natural language corpora; and these properties can be exploited to build better software engineering tools.

为什么代码对LLM的训练有帮助？



(2) 代码与自然语言之间存在对应性

- ◆ 例如，可以把代码模块映射为自然语言描述，也可以利用自然语言描述的思维链辅助代码的生成。

```
def convert_time_unit(hours, minutes):  
    '''  
    Write a function to convert time unit.  
    '''  
    seconds = hours * 60 * 60  
    seconds = minutes * 60  
    return seconds  
  
# Generate some comments for above code  
def convert_time_unit(hours, minutes):  
    '''  
    Write a function to convert time unit.  
    '''  
    # Convert the time unit from hours to seconds.  
    seconds = hours * 60 * 60  
    # Convert the time unit from minutes to seconds.  
    seconds = minutes * 60  
    # Return the number of seconds.  
    return seconds
```

```
Task Id: 608  
#  
# 1. Define a function "bell_Number"  
def bell_Number(n):  
    # 2. Initialize a list "bell" with n+1 rows and n+1 columns.  
    bell = [[0 for i in range(n+1)] for j in range(n+1)]  
  
    # 3. Set the first element of the list "bell" to 1.  
    bell[0][0] = 1  
    # 4. Fill the first column of the list "bell" with the first n+1 Bell numbers.  
    for i in range(1, n+1):  
        bell[i][0] = bell[i-1][i-1]  
    # 5. Fill the rest of the list "bell" with the next n+1 Bell numbers.  
    for i in range(1, n+1):  
        for j in range(1, i+1):  
            bell[i][j] = bell[i-1][j-1] + bell[i][j-1]  
  
    # 6. Return the first element of the list "bell".  
    return bell[n][0]
```


为什么代码对LLM的训练有帮助？



(3) 程序语言结构性、顺序性、层次性强

- ◆ 代码的语义是对现实世界某个解决方案的描述，而且这种描述通常具有更加清晰的顺序性、结构性和层次性，有助于现实世界知识的学习。

```
def convert\_time\_unit(hours, minutes):  
    '''  
    Write a function to convert time unit.  
    '''  
    seconds = hours \* 60 \* 60  
    seconds = minutes \* 60  
    return seconds  
  
# Generate some comments for above code  
  
def convert\_time\_unit(hours, minutes):  
    '''  
    Write a function to convert time unit.  
    '''  
    # Convert the time unit from hours to seconds.  
    seconds = hours \* 60 \* 60  
    # Convert the time unit from minutes to seconds.  
    seconds = minutes \* 60  
    # Return the number of seconds.  
    return seconds
```

```
Task Id: 608  
#  
# 1. Define a function "bell\_Number"  
def bell\_Number(n):  
    # 2. Initialize a list "bell" with n+1 rows and n+1 columns.  
    bell = [[0 for i in range(n+1)] for j in range(n+1)]  
  
    # 3. Set the first element of the list "bell" to 1.  
    bell[0][0] = 1  
    # 4. Fill the first column of the list "bell" with the first n+1 Bell numbers.  
    for i in range(1, n+1):  
        bell[i][0] = bell[i-1][i-1]  
    # 5. Fill the rest of the list "bell" with the next n+1 Bell numbers.  
    for i in range(1, n+1):  
        for j in range(1, i+1):  
            bell[i][j] = bell[i-1][j-1] + bell[i][j-1]  
  
    # 6. Return the first element of the list "bell".  
    return bell[n][0]
```

Self-planning Code Generation with Large Language Model

Xue Jiang, Yihong Dong, Lecheng Wang, Qiwei Shang, Ge Li*

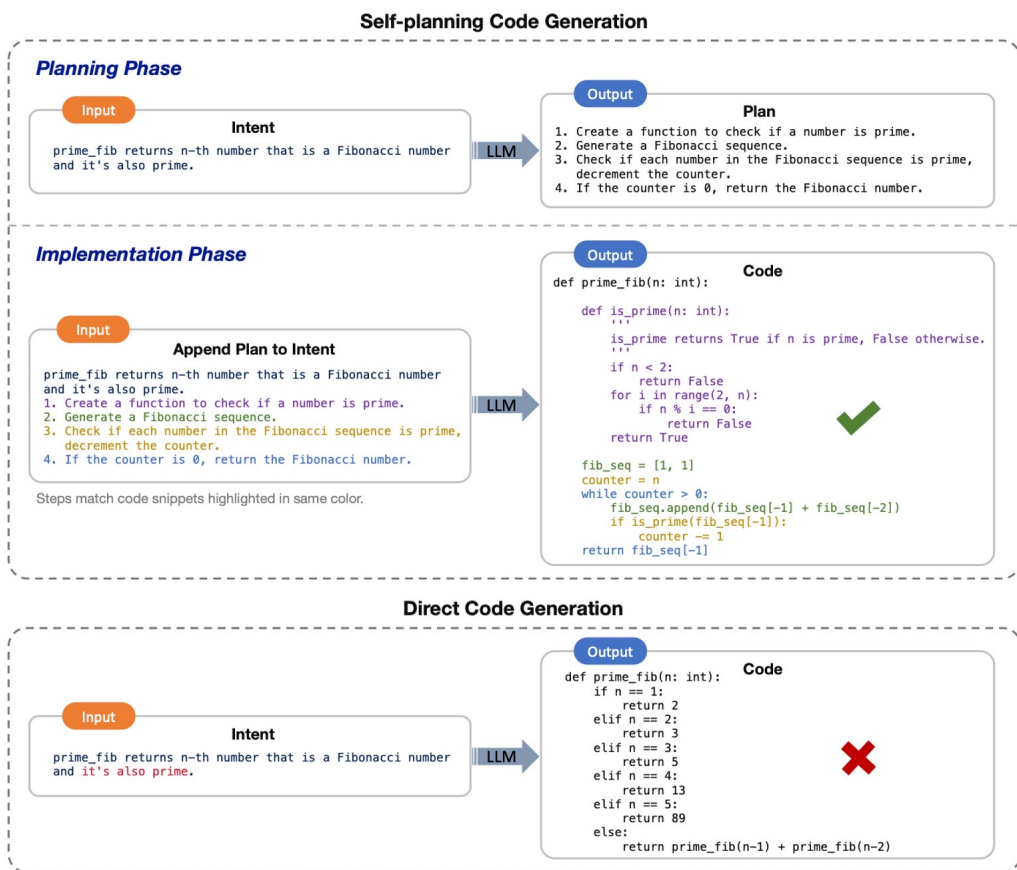


Figure 1: Self-planning code generation is carried out in two phases (i.e., planning phase and implementation phase): 1) In planning phase, LLM decomposes an intent into a set of easily solvable sub-problems and devises a plan for executing the solution steps; 2) In implementation phase, LLM generates code following the intent and plan, which assists self-planning code generation to be capable of handling more difficult problems than direct code generation.

Self-planning Code Generation	Direct Code Generation
<pre>def minSubArraySum(nums): .. Given an array of integers nums, find the minimum sum of any non-empty sub-array of nums. Example minSubArraySum([2, 3, 4, 1, 2, 4]) == 1 minSubArraySum([-1, -2, -3]) == -6 .. 1. Create a function to calculate the sum of a sub-array. 2. Loop the input list, calculate the sum of each sub-array. 3. Return the minimum sum. .. def subArraySum(nums): sum = 0 for i in nums: sum += i return sum min_sum = subArraySum(nums) for i in range(len(nums)): for j in range(i+1, len(nums)+1): sum = subArraySum(nums[i:j]) if sum < min_sum: min_sum = sum return min_sum</pre>	<pre>def minSubArraySum(nums): .. Given an array of integers nums, find the minimum sum of any non-empty sub-array of nums. Example minSubArraySum([2, 3, 4, 1, 2, 4]) == 1 minSubArraySum([-1, -2, -3]) == -6 .. min_sum = float('inf') curr_sum = 0 for num in nums: curr_sum += num min_sum = min(min_sum, curr_sum) curr_sum = max(curr_sum, 0) return min_sum</pre>
Case I	

Figure 2: Three real cases from HumanEval with Self-planning and direct code generation. The input, generated plan, and code are highlighted in green, red, and black respectively.

Table 1: Comparison of self-planning and other methods.

Method	Pass@1	CodeBLEU	Pass@1
	HumanEval		HumanEval-ET
Direct	47.6	24.0	37.2
CoT	53.9	30.4	45.5
Self-planning	60.3 (↑ 26.7%)	28.6	46.2 (↑ 24.1%)
Ground Truth Planning	74.4	41.0	57.7
		MBPP-sanitized	
Direct	49.9	25.6	37.7
CoT	54.7	26.4	39.6
Self-planning	55.3 (↑ 10.9%)	24.9	41.9 (↑ 11.2%)
Ground-truth Planning	65.1	33.7	50.7

AceCoder : Improving In-Context Learning for Code Generation

Jia Li, Kechi Zhang, Ge Li

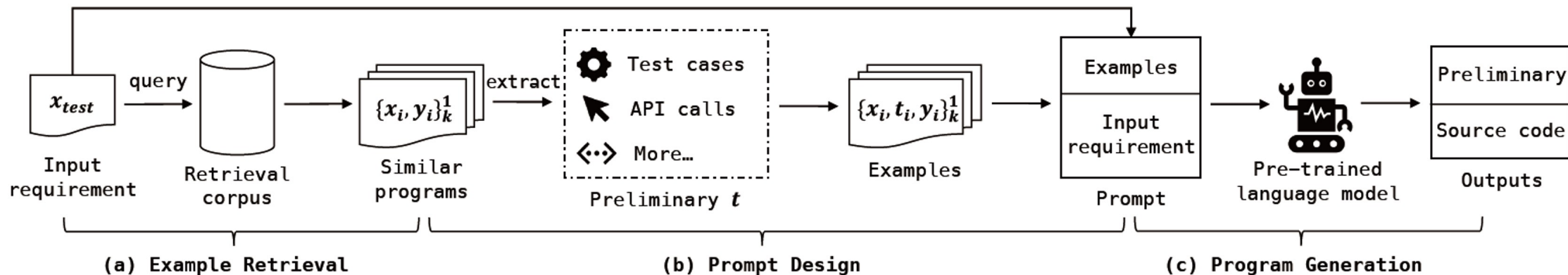


Table 2: The results of ACECODER and ICL-based baselines on three datasets. The values in parentheses are the relative improvements compared to standard ICL.

Base model	Approach	MBPP			MBJP			MBJSP		
		Pass@1 (%)	Pass@3 (%)	Pass@5 (%)	Pass@1 (%)	Pass@3 (%)	Pass@5 (%)	Pass@1 (%)	Pass@3 (%)	Pass@5 (%)
Codex-175B	Zero-shot	25.00	44.20	51.40	14.00	29.61	38.54	24.95	44.42	49.13
	Standard ICL	37.20	54.40	60.60	43.00	58.82	64.91	31.23	46.04	50.71
	ACECODER	47.40 (↑ 27.4%)	62.20 (↑ 14.3%)	66.40 (↑ 9.6%)	49.90 (↑ 16%)	61.26 (↑ 4.1%)	66.51 (↑ 2.5%)	41.38 (↑ 32.5%)	53.75 (↑ 16.8%)	58.42 (↑ 15.2%)
CodeGeeX-13B	Zero-shot	5.20	13.80	19.40	4.46	11.97	18.26	0.20	0.20	0.41
	Standard ICL	20.40	30.60	36.00	16.63	26.17	34.48	11.16	19.88	25.56
	ACECODER	25.40 (↑ 24.5%)	35.20 (↑ 15%)	40.00 (↑ 11.1%)	27.18 (↑ 63.4%)	35.90 (↑ 37.2%)	40.57 (↑ 17.7%)	20.06 (↑ 79.7%)	30.83 (↑ 55.1%)	35.70 (↑ 39.7%)
CodeGen-6B	Zero-shot	10.40	19.40	24.40	14.81	25.76	31.44	8.72	19.67	22.92
	Standard ICL	14.60	24.00	30.20	18.25	30.02	34.68	9.94	19.88	23.12
	ACECODER	21.60 (↑ 47.9%)	33.60 (↑ 40%)	39.00 (↑ 29.1%)	21.55 (↑ 18.1%)	33.48 (↑ 11.5%)	40.16 (↑ 15.8%)	15.64 (↑ 57.3%)	26.57 (↑ 33.7%)	31.42 (↑ 35.9%)
InCoder-6B	Zero-shot	4.20	11.40	16.20	2.23	5.88	9.13	3.65	5.88	8.11
	Standard ICL	12.80	22.80	28.20	10.95	23.53	26.17	12.78	22.52	27.79
	ACECODER	19.40 (↑ 51.6%)	30.40 (↑ 33.3%)	33.80 (↑ 19.9%)	15.82 (↑ 44.5%)	29.01 (↑ 23.3%)	34.08 (↑ 30.2%)	15.42 (↑ 20.7%)	26.77 (↑ 18.9%)	30.22 (↑ 8.7%)



问题4：代码与自然语言有什么不同？

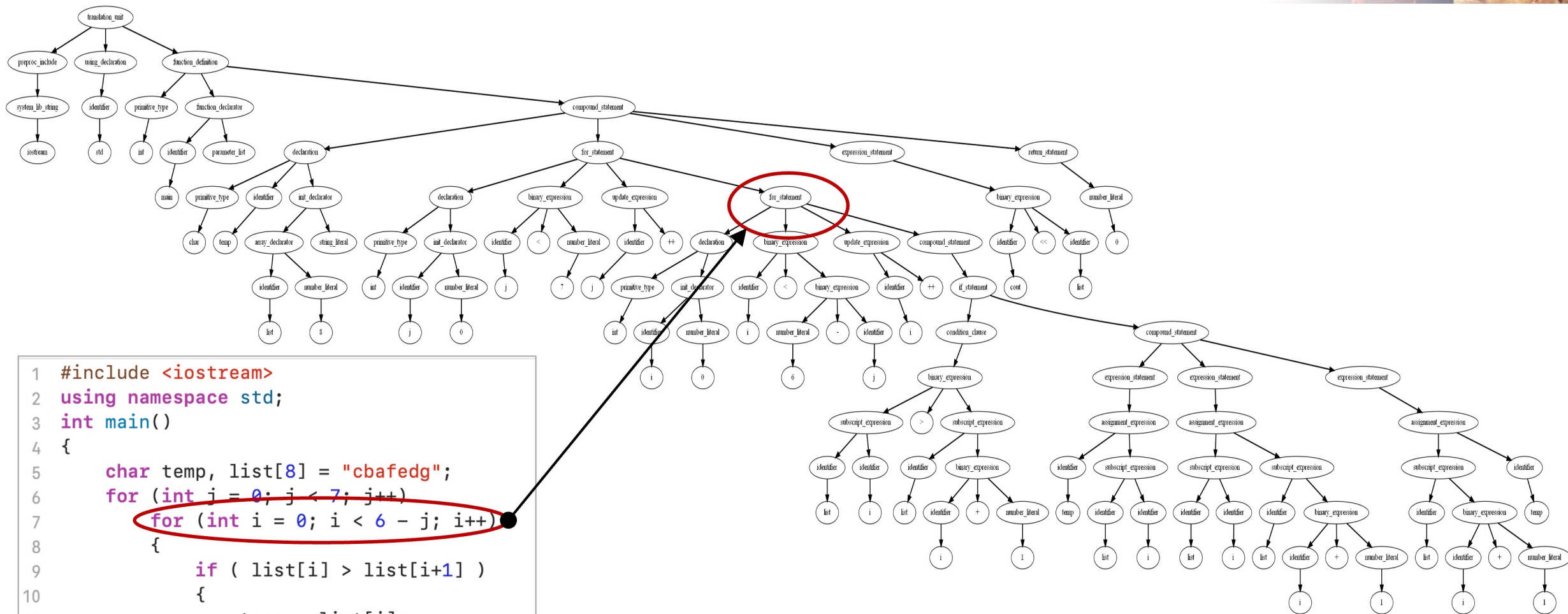


However,

Currently, people process code like they do with natural languages.

```
private static Object[] covertArray(Class<?> clazz, String vals[], int from, int to)
throws NoSuchMethodException, IllegalAccessException, InvocationTargetException
{ int start; int end; if (from > to) { start = to; end = from; } else { start = from;
end = to; } Object result[] = (Object[]) Array.newInstance(clazz, to - from); Method
valueOfMethod = clazz.getMethod("valueOf", new Class[] { String.class }); boolean
accessible = valueOfMethod.isAccessible(); valueOfMethod.setAccessible(true); if
(valueOfMethod != null) { for (int i = start; i < end; i++) { Object val =
valueOfMethod.invoke(clazz, new Object[] { vals[i] }); result[i - start] = val; } }
valueOfMethod.setAccessible(accessible); return result; }
```

(1) 代码的结构性强于自然语言



```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     char temp, list[8] = "cbafedg";
6     for (int j = 0; j < 7; j++)
7         for (int i = 0; i < 6 - j; i++)
8         {
9             if ( list[i] > list[i+1] )
10            {
11                temp = list[i];
12                list[i] = list[i+1];
13                list[i+1] = temp;
14            }
15        }
16     cout<<list;
17     return 0;
18 }
```

- 代码可以映射为抽象语法树或具体语法树，具有严格的结构化特性，而当前的大模型通常依赖于代码的自然性进行建模。

(2) 代码的自然性弱于自然语言



- 表现为编程语言可以利用语法产生式进行严格定义，编写的代码的过程是在一个更加受限的语法空间中描述程序语义。

```
def FordFulkerson(graph, source, sink):
    # This array is filled by BFS and to store path
    parent = [-1] * (len(graph))
    max_flow = 0
    while BFS(graph, source, sink, parent):
        path_flow = float("Inf")
        s = sink

        while s != source:
            # Find the minimum value in select path
            path_flow = min(path_flow, graph[parent[s]][s])
            s = parent[s]

        max_flow += path_flow
        v = sink

        while v != source:
            u = parent[v]
            graph[u][v] -= path_flow
            graph[v][u] += path_flow
            v = parent[v]

    return max_flow
```

is defined by

```
1 single_input: NEWLINE | simple_stmt | compound_stmt NEWLINE
2 file_input: (NEWLINE | stmt)* ENDMARKER
3 eval_input: testlist NEWLINE* ENDMARKER
4 decorator: '@' dotted_name [' (' arglist ') ' ] NEWLINE
5 decorators: decorator*
6 decorated: decorators (classdef | funcdef | async_funcdef)
7 async_funcdef: 'async' funcdef
8 funcdef: 'def' NAME parameters ['>' test] ':' suite
9 parameters: '(' [typedarglist] ')'
10 typedarglist: (tfpdef ['=' test] (',' tfpdef ['=' test])* ['**' tfpdef ['=' test']] | '*' tfpdef ['='] | '*' tfpdef ['='] | '*' tfpdef ['='] | '*' tfpdef ['=']
11 tfpdef: NAME ':' test
12 vararglist: '*' fpdef ['=' test] ':' fpdef ['=' test']* ['**' fpdef ['=' test']] | '*' fpdef ['='] | '*' fpdef ['='] | '*' fpdef ['='] | '*' fpdef ['=']
13 vfpdef: NAME ':' test
14 stmt: simple_stmt | compound_stmt
15 simple_stmt: small_stmt (';' small_stmt)* [';'] NEWLINE
16 small_stmt: (expr_stmt | del_stmt | pass_stmt | flow_stmt | import_stmt | global_stmt | nonlocal_stmt | assert_stmt)
17 expr_stmt: testlist_star_expr (annassign | augassign (yield_expr | testlist) | '=' (yield_expr | testlist_star_expr)*)
18 annassign: ':' test ['=' test]
19 testlist_star_expr: (test | star_expr) [';' (test | star_expr)]*
20 augassign: ('+' | '-' | '*' | '@' | '/' | '%' | '&' | '|' | '^' | '<<' | '>>' | '**' | '//')
21 del_stmt: 'del' exprlist
22 pass_stmt: 'pass'
23 flow_stmt: break_stmt | continue_stmt | return_stmt | raise_stmt | yield_stmt
24 break_stmt: 'break'
25 continue_stmt: 'continue'
26 return_stmt: 'return' [testlist]
27 yield_stmt: yield_expr
28 raise_stmt: 'raise' [test ['from' test]]
29 import_stmt: import_name | import_from
30 import_name: 'import' dotted_as_names
31 import_from: 'from' (('(' | '[') | '.')* dotted_name [' (' | '[') | ':' | ']' | 'import' (('(' | '[') | 'import_as_names') | import_as_names)
32 import_as_name: NAME ['as' NAME]
33 dotted_as_name: dotted_name ['as' NAME]
34 import_as_names: import_as_name (',' import_as_name)*
35 dotted_as_names: dotted_as_name (',' dotted_as_name)*
36 dotted_name: NAME (['.' NAME])*
37 global_stmt: 'global' NAME (',' NAME)*
38 nonlocal_stmt: 'nonlocal' NAME (',' NAME)*
39 assert_stmt: 'assert' test ['test']
40 compound_stmt: if_stmt | while_stmt | for_stmt | try_stmt | with_stmt | funcdef | classdef | decorated | async_stmt
41 if_stmt: 'if' test ':' suite ['elif' test ':' suite]* ['else' ':' suite]
42 while_stmt: 'while' test ':' suite ['else' ':' suite]
43 for_stmt: 'for' exprlist 'in' testlist ':' suite ['else' ':' suite]
44 try_stmt: 'try' ':' suite [(except_clause ':' suite)+ ['else' ':' suite] ['finally' ':' suite] ['finally' ':' suite])
45 with_stmt: 'with' with_item (',' with_item)* ':' suite
46 with_item: test ['as' expr]
47 except_clause: 'except' [test ['as' NAME]]
48 suite: simple_stmt | NEWLINE INDENT stmt* DEDENT
49 test: or_test ['if' or_test 'else' test] | lambdef
50 test_nocond: or_test | lambdef_nocond
51 lambdef: 'lambda' (vararglist) ':' test
52 lambdef_nocond: 'lambda' (vararglist) ':' test_nocond
53 or_test: and_test ('or' and_test)*
54 and_test: not_test ('and' not_test)*
55 not_test: 'not' not_test | comparison
56 comparison: expr (comp_op expr)*
57 comp_op: '<' | '<=' | '>' | '>=' | '<<' | '>>' | 'in' | 'is' | 'is not'
58 star_expr: '**' expr
59 xor_expr: ('^') xor_expr*
60 xor_expr: and_expr ('^') and_expr*
61 and_expr: shift_expr ('&' shift_expr)*
62 shift_expr: arith_expr ('<<' | '>>') arith_expr*
63 arith_expr: term (('+' | '-') term)*
64 term: factor (('*' | '/') | '%' | '//') factor*
65 factor: ('+' | '-') factor | power
66 power: atom_expr ('**') factor
67 atom_expr: ['await'] atom trailer*
68 atom: '[' [yield_expr | testlist_comp] ']' | '[' [testlist_comp] ']' | '[' [dictorsetmaker] ']' | NAME | NUMBER | STRING+ | '...' | 'None' | 'True' | 'False'
69 testlist_comp: (test | star_expr) (comp_for | '(' | '[') [test | star_expr]* (']' | ')')
70 trailer: '(' [arglist] ')' | '[' subscriptlist ']' | '.' NAME
71 subscriptlist: subscript (',' subscript)*
72 subscript: test | testlist | test [sliceop]
73 sliceop: ':' [test]
74 exprlist: (expr | star_expr) (',' (expr | star_expr))*
75 testlist: test (',' test)*
76 dictorsetmaker: ( (('test' | '**' expr) (comp_for | '(' | '[') [test | '**' expr]* (']' | ')') | (('test' | star_expr) (comp_for | '(' | '[') [test | star_expr]* (']' | ')'))
77 classdef: 'class' NAME [' (' arglist ') ' ] ':' suite
78 arglist: argument (',' argument)*
79 argument: (test [comp_for] | test '=' test ['**' test] ['**' test])
80 comp_iter: comp_for | comp_if
81 sync_comp_for: 'for' exprlist 'in' or_test [comp_iter]
82 comp_for: ['async'] sync_comp_for
83 comp_if: 'if' test_nocond [comp_iter]
84 encoding_decl: NAME
85 yield_expr: 'yield' [yield_arg]
86 yield_arg: 'from' test | testlist
```

(3) 代码的局部性强于自然语言



- 表现为程序代码中的诸多成分的语义，重度依赖于所在的上下文环境，仅在该受限的上下文环境中才具备特定语义。

```
def FordFulkerson(graph, source, sink):  
    # This array is filled by BFS and to store path  
    parent = [-1] * (len(graph))  
    max_flow = 0  
    while BFS(graph, source, sink, parent):  
        path_flow = float("Inf")  
        s = sink  
        while s != source:  
            # Find the minimum value in select path  
            path_flow = min(path_flow, graph[parent[s]][s])  
            s = parent[s]  
        max_flow += path_flow  
        v = sink  
        while v != source:  
            u = parent[v]  
            graph[u][v] -= path_flow  
            graph[v][u] += path_flow  
            v = parent[v]  
    return max_flow
```

parent ?

graph ?

s ? v ? u ?

source ?

sink ?

n ?

temp ?

path_flow ?

(4) 代码的环境依赖性强于自然语言



- 表现为代码多数依附于已有的软件架构进行编写，代码的语义通常是在特定的软件框架环境中的表达。

```
def multilayer_perceptron(x):
    # Hidden fully connected layer with 256 neurons
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
    # Hidden fully connected layer with 256 neurons
    layer_2 = tf.add(tf.matmul(layer_1, weights['h2']), biases['b2'])
    # Output fully connected layer with a neuron for each class
    out_layer = tf.matmul(layer_2, weights['out']) + biases['out']
    return out_layer

# Construct model
logits = multilayer_perceptron(X)

# Define loss and optimizer
loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)

# Initializing the variables
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
```

```
tf.add(tf.matmul(x, weights['h1']), biases['b1'])
```

?

```
tf.matmul(layer_2, weights['out']) + biases['out']
```

?

```
loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
```

?

```
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)
```

?

GPT-3.5

GPT-3.5 models can understand and generate natural language or code. Our most capable and cost effective model in the GPT-3.5 family is `gpt-3.5-turbo` which has been optimized for chat but works well for traditional completions tasks as well.

LATEST MODEL	DESCRIPTION	MAX TOKENS	TRAINING DATA
<code>gpt-3.5-turbo</code>	Most capable GPT-3.5 model and optimized for chat at 1/10th the cost of <code>text-davinci-003</code> . Will be updated with our latest model iteration.	4,096 tokens	Up to Sep 2021
<code>gpt-3.5-turbo-0301</code>	Snapshot of <code>gpt-3.5-turbo</code> from March 1st 2023. Unlike <code>gpt-3.5-turbo</code> , this model will not receive updates, and will be deprecated 3 months after a new version is released.	4,096 tokens	Up to Sep 2021
<code>text-davinci-003</code>	Can do any language task with better quality, longer output, and consistent instruction-following than the <code>curie</code> , <code>babbage</code> , or <code>ada</code> models. Also supports inserting completions within text.	4,097 tokens	Up to Jun 2021
<code>text-davinci-002</code>	Similar capabilities to <code>text-davinci-003</code> but trained with supervised fine-tuning instead of reinforcement learning	4,097 tokens	Up to Jun 2021
<code>code-davinci-002</code>	Optimized for code-completion tasks	8,001 tokens	Up to Jun 2021



问题5：代码是否也应该用语言模型建模？



Program can be

a Token Sequence.

a Tree Structure.

a Graph.

Data flow Graph

Control flow Graph

Call Graph

.....

an Operation sequence.

an Production sequence.

问题5+：现有程序表示技术能否用于大模型？

Can we make use of different representations of programs to build **BIG MODELS**?

—— Our research has confirmed that these approaches are effective in small models.

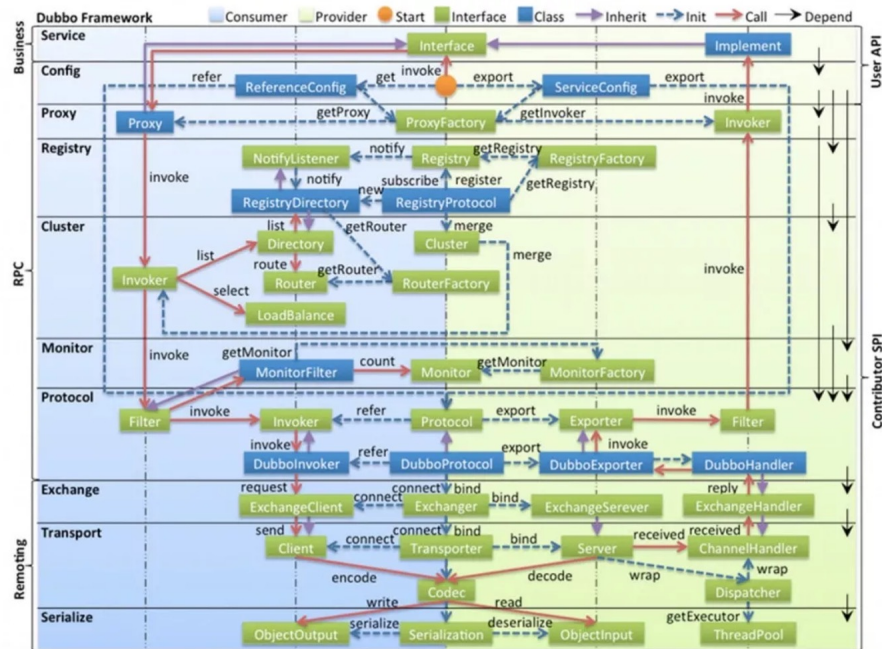


问题6：当前大模型在软件开发中的应用 存在哪些问题？

(1) 大模型与领域知识如何结合？



Big Model +



—— Software development usually depends on personalized private software development framework.
How can we let the big model learn domain knowledge?

(2) 复杂上下文如何与大模型交互？



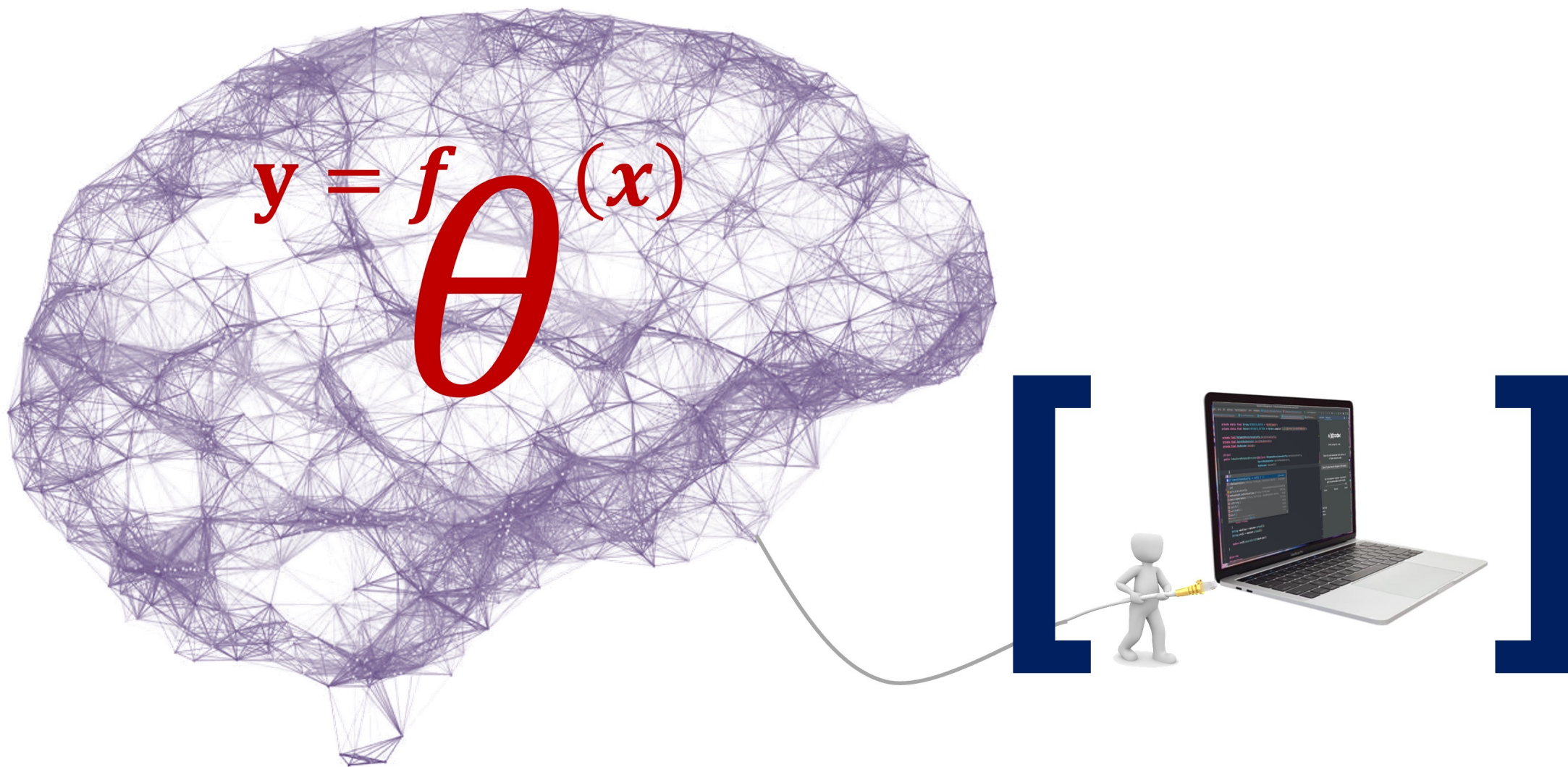
Big Model +

```
public static Calendar strToCalendar(String pubTime, String format) {
    try {
        SimpleDateFormat dateFormat = new SimpleDateFormat(format);
        Date date = dateFormat.parse(pubTime);
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(date);
        return calendar;
    } catch (Exception e) {
        return null;
    }
}

// 半角转全角
public static String toSBC(String input, Set<Character> notConvertSet) {
    char[] c = input.toCharArray();
    for (int i = 0; i < c.length; i++) {
        if (null != notConvertSet && notConvertSet.contains(c[i]))
            continue;
        if (c[i] == ' ') {
            c[i] = '\u3000';
        } else if (c[i] < '\u4ff7') {
            c[i] = (char)(c[i] + 65248);
        }
    }
}
```

- Software development happens in a complex environment, and it is difficult to transfer all the things to a big model.
- What we can do is pull the big model into this complex environment to play its role.

(3) 如何保护代码的知识产权？



问题7：大模型对软件开发有何影响？



编程 ≠ 软件开发

```
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]

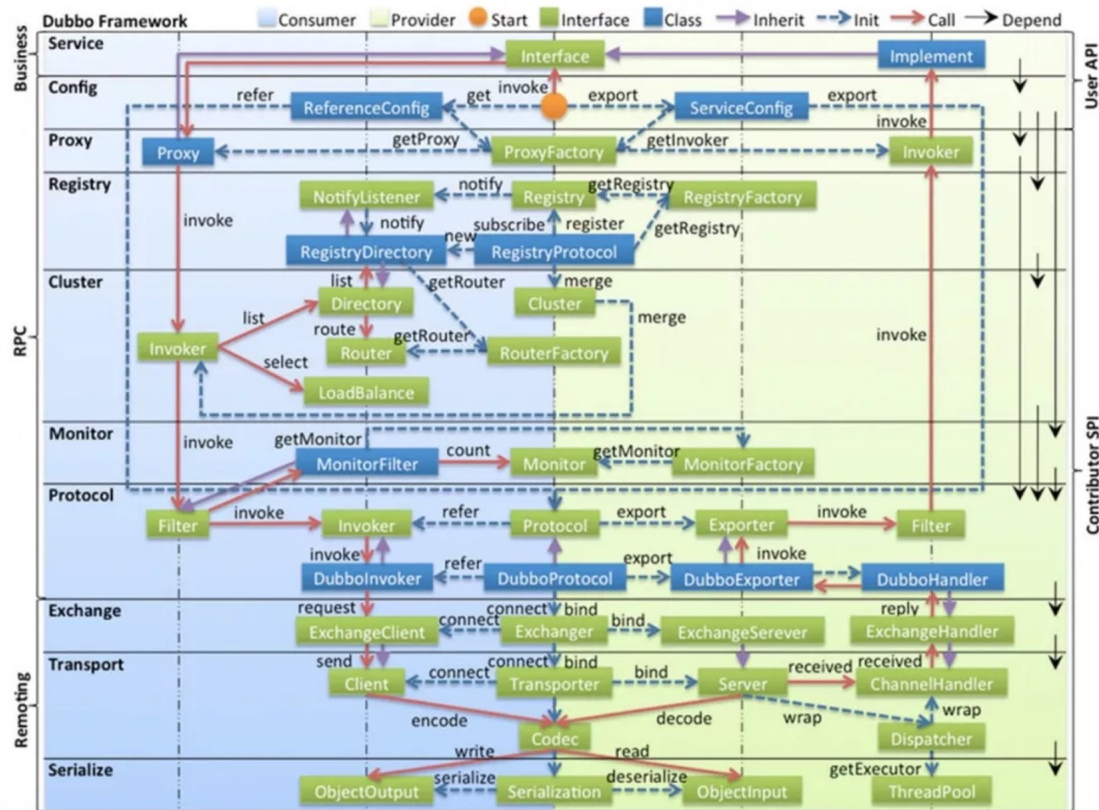
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) =>12
    solution([3, 3, 3, 3, 3]) =>9
    solution([30, 13, 24, 321]) =>0
    """
    return sum(lst[i] for i in range(0, len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)

def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

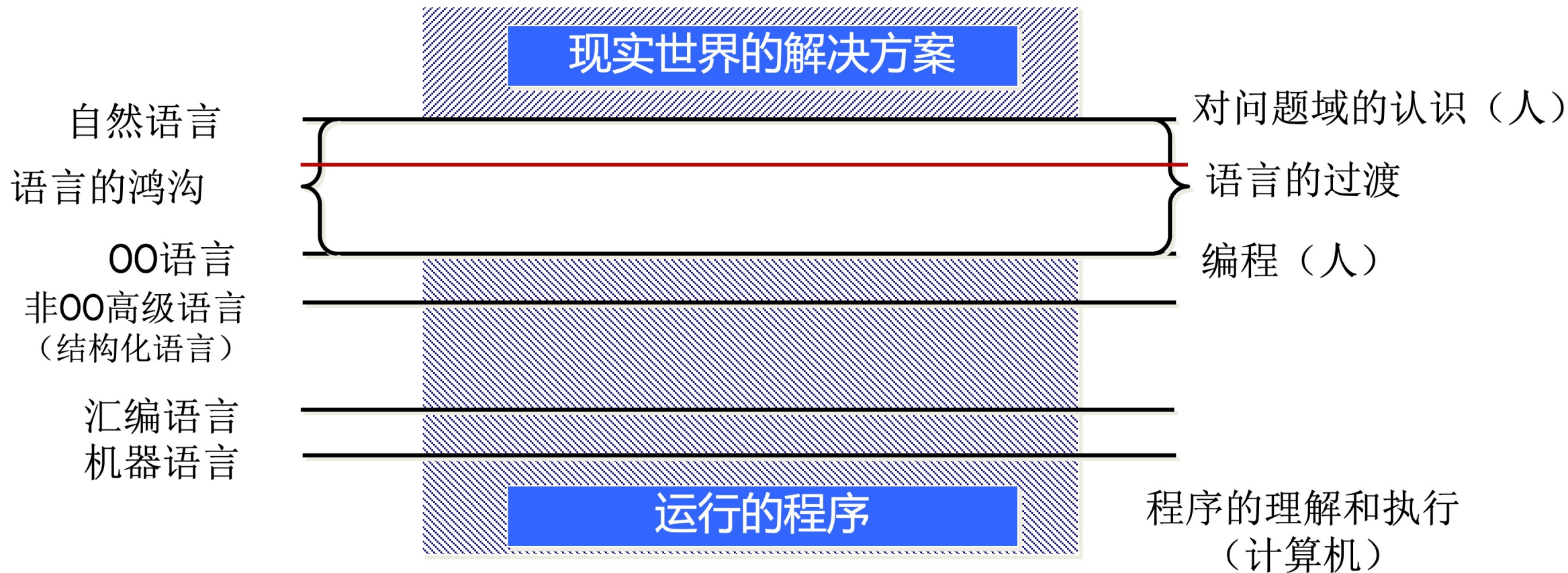
VS.



问题8：大模型对软件开发有何影响？



Maybe, the advanced programming language will become today's assembly language.





And, what will happen
if programs can be generated automatically?

—
Automated requirement analysis will be required.

Automated testing will be required.

Automatic repair will be required.

Automated defect detection will be required.

.....

in the end



The big model has shown us the dawn of software development automation, but we are also convinced that there is still a long way to go in the future.

THANKS