TASK PROCESSING METHOD, CODE COMPLETION METHOD, CODE PROCESSING MODEL TRAINING METHOD, INFORMATION PROCESSING METHOD BASED ON CODE PROCESSING MODEL, AND MODEL TRAINING PLATFORM

The present disclosure claims priority to Chinese Patent Application No. 202410331805.8, filed with the China National Intellectual Property Administration on March 21, 2024 and entitled "TASK PROCESSING METHOD, CODE COMPLETION METHOD, CODE PROCESSING MODEL TRAINING METHOD, INFORMATION PROCESSING METHOD BASED ON CODE PROCESSING MODEL, AND MODEL TRAINING PLATFORM", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

Embodiments of the present disclosure relate to the field of computer technologies, and in particular, to a task processing method, a code completion method, a code processing model training method, an information processing method based on a code processing model, and a model training platform.

BACKGROUND

With the development of computer technologies, large models have begun to exhibit prominent advantages, showing remarkable capabilities in language understanding, generation, interaction, and reasoning, and are widely applied to the natural language processing fields such as dialogue, translation, and code processing. Using the code processing field as an example, intelligent capabilities such as code completion and code commenting can be provided for developers by using the large models. Therefore, the large models have gradually become a research focus.

Currently, code text of a target task is usually directly processed by using the large model, to obtain a code processing result. However, in the foregoing method, the large model needs to predict overly long content at a time, resulting in excessively long prediction time and low prediction accuracy. Therefore, there is an urgent need for an efficient and accurate task processing solution.

SUMMARY

In view of this, embodiments of the present disclosure provide a task processing method. One or more embodiments of the present disclosure also relate to a code completion method, a

code processing model training method, an information processing method based on a code processing model, a task processing apparatus, a code completion apparatus, a code processing model training apparatus, an information processing apparatus based on a code processing model, a model training platform, a computing device, a computer-readable storage medium, and a computer program product, to resolve a technical disadvantage in the related art in which "a large model needs to predict overly long content at a time, resulting in excessively long prediction time and low prediction accuracy".

According to a first aspect of the embodiments of the present disclosure, a task processing method is provided, including:

obtaining to-be-processed code data of a target task; and

inputting the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

According to a second aspect of the embodiments of the present disclosure, a code completion method is provided, including:

obtaining to-be-completed code data of a code completion task; and

inputting the to-be-completed code data into a code completion model, to obtain a code completion result of the code completion task, where the code completion model is obtained by performing training based on a plurality of sample completion results and sample code block data respectively corresponding to the plurality of sample completion results, and the sample completion results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

According to a third aspect of the embodiments of the present disclosure, a code processing model training method is provided, including:

obtaining a sample set, where the sample set includes a plurality of pieces of sample code data;

performing hierarchical sampling on the plurality of pieces of sample code data, to obtain a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results;

inputting the plurality of pieces of sample code block data into a code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of

sample code block data; and

training the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

According to a fourth aspect of the embodiments of the present disclosure, an information processing method based on a code processing model is provided, including:

receiving a task generation request, where the task generation request includes request information;

obtaining the code processing model based on the request information, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data; and

generating task information based on the code processing model, where the task information is used for executing a target task.

According to a fifth aspect of the embodiments of the present disclosure, a task processing apparatus is provided, including:

a first obtaining module, configured to obtain to-be-processed code data of a target task; and

a first input module, configured to input the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

According to a sixth aspect of the embodiments of the present disclosure, a code completion apparatus is provided, including:

a second obtaining module, configured to obtain to-be-completed code data of a code completion task; and

a second input module, configured to input the to-be-completed code data into a code completion model, to obtain a code completion result of the code completion task, where the code completion model is obtained by performing training based on a plurality of sample completion results and sample code block data respectively corresponding to the plurality of sample completion results, and the sample completion results and the sample code block data

are obtained by performing hierarchical sampling on sample code data.

According to a seventh aspect of the embodiments of the present disclosure, a code processing model training apparatus is provided, including:

a third obtaining module, configured to obtain a sample set, where the sample set includes a plurality of pieces of sample code data;

a sampling module, configured to perform hierarchical sampling on the plurality of pieces of sample code data, to obtain a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results;

a third input module, configured to input the plurality of pieces of sample code block data into a code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and

a training module, configured to train the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

According to an eighth aspect of the embodiments of the present disclosure, an information processing apparatus based on a code processing model is provided, including:

a first receiving module, configured to receive a task generation request, where the task generation request includes request information;

a fourth obtaining module, configured to obtain the code processing model based on the request information, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data; and

a generation module, configured to generate task information based on the code processing model, where the task information is used for executing a target task.

According to a ninth aspect of the embodiments of the present disclosure, a model training platform is provided, including a request interface unit and a model unit, where

the request interface unit is configured to receive a model training request, where the model training request includes request information; and

the model unit is configured to obtain a code processing model; and train the code processing model based on the request information, to obtain a code processing model on which the training is completed, where the request information includes a plurality of sample processing results and sample code block data respectively corresponding to the plurality of

sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

According to a tenth aspect of the embodiments of the present disclosure, a computing device is provided, including:

a memory and a processor, where

the memory is configured to store a computer program or computer instructions, the processor is configured to execute the computer program or the computer instructions, and the computer program or the computer instructions, when executed by the processor, implement steps of the method according to the first aspect, the second aspect, the third aspect, or the fourth aspect.

According to an eleventh aspect of the embodiments of the present disclosure, a computer-readable storage medium is provided. The computer-readable storage medium stores a computer program or computer instructions, and the computer program or the computer instructions, when executed by a processor, implement steps of the method according to the first aspect, the second aspect, the third aspect, or the fourth aspect.

According to a twelfth aspect of the embodiments of the present disclosure, a computer program product is provided. The computer program product includes a computer program or computer instructions, and the computer program or the computer instructions, when executed by a processor, implement steps of the method according to the first aspect, the second aspect, the third aspect, or the fourth aspect.

A task processing method provided in an embodiment of the present disclosure includes: obtaining to-be-processed code data of a target task; and inputting the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data. Because the code processing model is obtained by performing training based on the sample code block data, prediction is performed by using a code block as a unit, thereby shortening predicted content. In addition, through hierarchical sampling, the code processing model can be adapted to various code processing scenarios, thereby accurately predicting various types of code blocks, and improving task processing efficiency and accuracy.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 is an architectural diagram of a task processing system according to an embodiment of the present disclosure.

FIG. 2 is an architectural diagram of another task processing system according to an embodiment of the present disclosure.

FIG. 3 is a flowchart of a task processing method according to an embodiment of the present disclosure.

FIG. 4 is a flowchart of a code completion method according to an embodiment of the present disclosure.

FIG. 5 is a flowchart of a code processing model training method according to an embodiment of the present disclosure.

FIG. 6 is a flowchart of a processing process of a code processing model training method according to an embodiment of the present disclosure.

FIG. 7 is a flowchart of an information processing method based on a code processing model according to an embodiment of the present disclosure.

FIG. 8 is a schematic structural diagram of a model training platform according to an embodiment of the present disclosure.

FIG. 9 is a schematic structural diagram of a task processing apparatus according to an embodiment of the present disclosure.

FIG. 10 is a schematic structural diagram of a code completion apparatus according to an embodiment of the present disclosure.

FIG. 11 is a schematic structural diagram of a code processing model training apparatus according to an embodiment of the present disclosure.

FIG. 12 is a schematic structural diagram of an information processing apparatus based on a code processing model according to an embodiment of the present disclosure.

FIG. 13 is a structural block diagram of a computing device according to an embodiment of the present disclosure.

DESCRIPTION OF EMBODIMENTS

In the following descriptions, numerous specific details are described to provide a thorough understanding of the present disclosure. However, the present disclosure can be implemented in many other manners different from those described herein, and a person skilled in the art can make similar promotion without violating the connotation of the present disclosure. Therefore, the present disclosure is not limited to specific implementations

disclosed below.

The terms used in one or more embodiments of the present disclosure are for the purpose of describing specific embodiments only and are not intended to limit one or more embodiments of the present disclosure. Singular forms "a/an" and "the" used in one or more embodiments and the appended claims of the present disclosure are also intended to include plural forms, unless the context clearly indicates otherwise. It should be further understood that the term "and/or" used in one or more embodiments of the present disclosure indicates and includes any or all possible combinations of one or more associated listed items.

It should be understood that although the terms "first", "second", and the like may be used for describing various types of information in one or more embodiments of the present disclosure, such information should not be limited to these terms. These terms are merely used for distinguishing between information of the same type. For example, without departing from the scope of one or more embodiments of the present disclosure, "first" may also be referred to as "second". Similarly, "second" may also be referred to as "first". Depending on the context, for example, the term "if" used herein may be explained as "when", "upon", or "in response to determining".

In addition, it should be noted that, user information (including but not limited to user equipment information, user personal information, and the like) and data (including but not limited to data for analysis, stored data, displayed data, and the like) involved in one or more embodiments of the present disclosure are all information and data that are authorized by a user or that are fully authorized by all parties, and related data needs to be collected, used, and processed by complying with relevant laws, regulations, and standards of relevant countries and regions. In addition, a corresponding operation entry is provided for the user to select to authorize or reject.

In one or more embodiments of the present disclosure, a large model refers to a deep learning model having large-scale model parameters, and typically includes hundreds of millions, tens of billions, hundreds of billions, trillions, or even more than ten trillion model parameters. The large model may also be referred to as a foundation model (Foundation Model), and is pre-trained by using a large-scale corpus of unannotated data to produce a pre-trained model with hundreds of millions or more parameters. Such a model can adapt to a wide range of downstream tasks and exhibit strong generalization capabilities. Examples of such a model include a large language model (LLM, Large Language Model) and a multi-modal pre-training model (multi-modal pre-training model).

During actual application, only a small quantity of samples are required to fine-tune the

pre-trained model, so that the large model may be applied to different tasks. The large model may be widely applied to fields such as natural language processing (NLP, Natural Language Processing) and computer vision, and specifically may be applied to tasks in, for example, computer vision fields such as visual query answering (VQA, Visual Question Answering), image caption (IC, Image Caption), and image generation, and in natural language processing fields such as text-based sentiment classification, text abstract generation, and machine translation. Main application scenarios of the large model include a digital assistant, an intelligent robot, search, online education, office software, e-commerce, intelligent design, or the like.

First, the terms involved in one or more embodiments of the present disclosure are explained.

Intelligent coding assistant: the intelligent coding assistant enhances programming tasks in a software development procedure and improves coding efficiency of developers by using large model technologies. Main capabilities of the intelligent coding assistant include automatic code completion, code question answering, automatic unit test generation, automatic code optimization, and the like.

Automatic code completion: the automatic code completion may also be referred to as code completion, and is to predict and recommend possible code completion items in real time in a code writing process of a developer, so as to increase a coding speed and improve accuracy of the developer.

Abstract syntax tree (AST, Abstract Syntax Tree): the abstract syntax tree is an intermediate representation form of source code, and is used for abstracting and parsing a structure of the source code in a tree-like structure.

Fill-In-Middle (Fill-In-Middle): the Fill-In-Middle is a data mode for training large code models, and is a data organization form constructed to enable the model to predict code content to be completed while understanding the code context.

Code metadata (Code Metadata): the code metadata is structured data of code, and includes additional information for describing code components (such as a class, a method, an attribute, an interface, or a file name). The information does not directly constitute execution logic of the code, but is crucial to understanding, managing, and using the code.

Child node (Child Node): in an abstract syntax tree (AST, Abstract Syntax Tree), all directly included sub-nodes under a code block node are referred to as child nodes of the code block. For example, in Java code, each statement or expression in a code block is a child node of the code block node.

Sibling node (Sibling Node): in the AST, any two nodes having a same parent node are referred to as sibling nodes.

In the field of software development, writing and maintaining code is a time-consuming task that requires a high degree of concentration. Through functions such as automatic code completion, code syntax checking, and code generation, the intelligent coding assistant can significantly reduce manual input time of developers, and accelerate development. In addition, the intelligent coding assistant can learn programming specifications and better practices based on a large amount of training data, help developers avoid common errors, and provide suggestions of high-quality code that conforms to a standard.

Using automatic code completion as an example, currently, automatic code completion can be usually performed in the following two manners. First, an automatic completion capability of a large model can be trained by using a Fill-In-Middle method. However, in a model training process, an end position of a code prefix (prefix) and a start position of a code suffix (suffix) are randomly selected from specified source code in a simple random sampling manner, and then all code in the middle (middle) of the two positions is used as a prediction result, to construct data of <prefix,suffix,middle> for training. However, in the foregoing solution, the model does not process code of a predicted middle part, so that the model predicts more content at a time, for example, predicts five functions. This not only results in longer prediction time, but also interferes with a coding pace of a user, which severely affects use experience of the intelligent coding assistant. Second, the model mostly performs prediction in a single-line completion form. However, in some cases, for example, when a developer intends to complete development of an entire function after writing function comments, automatic code completion needs to be triggered a plurality of times, which significantly reduces efficiency and experience of the user.

To resolve the foregoing problem, the embodiments of the present disclosure provide a method for training a code processing model of an intelligent coding assistant. Syntax parsing and hierarchical sampling are performed on original sample code data, and training data that can be used for training a model is constructed in a Fill-In-Middle mode, so that the model can process a code task based on to-be-processed code data written by a user. Specifically, to-be-processed code data of a target task may be obtained. The to-be-processed code data is input into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data

are obtained by performing hierarchical sampling on sample code data.

It should be noted that, according to the solution provided in the embodiments of the present disclosure, the intelligent coding assistant can implement, based on advanced artificial intelligence technologies, functions such as automatic code completion, code snippet recommendation, and error detection and repair through in-depth understanding of programming intentions of developers and context information, thereby significantly improving coding efficiency of developers. Because the code processing model is trained based on the sample code block data, prediction is performed by using a code block as a unit, thereby shortening predicted content. In addition, through hierarchical sampling, the code processing model can be adapted to various code processing scenarios (such as a code completion scenario, a code recommendation scenario, and a code comment generation scenario), thereby accurately predicting various types of code blocks (such as a class, a function, a logic block, a comment, and a statement), and improving task processing efficiency and accuracy.

In the present disclosure, a task processing method is provided. The present disclosure also relates to a code completion method, a code processing model training method, an information processing method based on a code processing model, a task processing apparatus, a code completion apparatus, a code processing model training apparatus, an information processing apparatus based on a code processing model, a model training platform, a computing device, a computer-readable storage medium, and a computer program product, which are described in detail one by one in the following embodiments.

FIG. 1 is an architectural diagram of a task processing system according to an embodiment of the present disclosure. The task processing system may include a client 100 and a server 200.

The client 100 is configured to send to-be-processed code data of a target task to the server 200.

The server 200 is configured to: input the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data; and send the code processing result to the client 100.

The client 100 is further configured to receive the code processing result sent by the server 200.

According to the solution in the embodiments of the present disclosure, because the code

processing model is obtained by performing training based on the sample code block data, prediction is performed by using a code block as a unit, thereby shortening predicted content. In addition, through hierarchical sampling, the code processing model can be adapted to various code processing scenarios, thereby accurately predicting various types of code blocks, and improving task processing efficiency and accuracy.

FIG. 2 is an architectural diagram of another task processing system according to an embodiment of the present disclosure. The task processing system may include a plurality of clients 100 and a server 200. The client 100 may include an end-side device. The server 200 may include a cloud-side device. A communication connection may be established among the plurality of clients 100 through the server 200. In a task processing scenario, the server 200 is configured to provide a code processing service among the plurality of clients 100. The plurality of clients 100 may be respectively used as senders or receivers, and communication is implemented through the server 200.

For example, a user may interact with the server 200 through the client 100 to receive data sent by another client 100, or send data to another client 100. In the task processing scenario, the user may publish a data stream to the server 200 through the client 100. The server 200 generates a code processing result based on the data stream, and pushes the code processing result to another client that establishes communication.

A connection is established between the client 100 and the server 200 via a network. The network provides a medium for a communication link between the client 100 and the server 200. The network may include various connection types, for example, a wired communication link, a wireless communication link, or an optical cable. Data transmitted by the client 100 may be published to the server 200 after processing such as encoding, transcoding, and compression.

The client 100 may be a browser, an APP (Application, application), or a web application such as H5 (HyperText Markup Language5, hypertext markup language5) application, a light application (also referred to as a mini program, a lightweight application), or a cloud application. The client 100 may be obtained through development based on a software development kit (SDK, Software Development Kit) of a corresponding service provided by the server 200, for example, may be obtained through development based on a real time communication (RTC, Real Time Communication) SDK. The client 100 may be deployed in an electronic device, and needs to run depending on a device, some APPs in the device, or the like. For example, the electronic device may have a display screen and support information browsing, and may be a personal mobile terminal such as a mobile phone, a tablet computer, or a personal computer. Various other types of applications, such as a human-machine

conversational application, a model training application, a text processing application, a web browser application, a shopping application, a search application, an instant messaging tool, an email client, and social platform software, may usually be configured in the electronic device.

The server 200 may include a server that provides various services, for example, a server that provides communication services for a plurality of clients, for another example, a server that provides support for a model used on a client for background training, and for another example, a server that processes data sent by a client. It should be noted that the server 200 may be implemented as a distributed server cluster including a plurality of servers, or may be implemented as a single server. The server may alternatively be a server of a distributed system, or a server combined with a blockchain. The server may alternatively be a cloud server of basic cloud computing services such as a cloud service, a cloud database, cloud computing, a cloud function, cloud storage, a network service, cloud communication, a middleware service, a domain name service, a security service, a content delivery network (CDN, Content Delivery Network), big data, and an artificial intelligence platform, or may be an intelligent cloud computing server or an intelligent cloud host with an artificial intelligence technology.

It should be noted that, the task processing method provided in the embodiments of the present disclosure is generally performed by the server. However, in other embodiments of the present disclosure, the client may also have a function similar to that of the server, to perform the task processing method provided in the embodiments of the present disclosure. In other embodiments, the task processing method provided in the embodiments of the present disclosure may alternatively be performed by the client and the server together.

FIG. 3 is a flowchart of a task processing method according to an embodiment of the present disclosure. The method specifically includes the following steps.

Step 302: obtain to-be-processed code data of a target task.

In one or more embodiments of the present disclosure, the to-be-processed code data of the target task may be obtained, so that based on the target task, the to-be-processed code data is processed by using a task processing method based on syntax parsing and hierarchical sampling, to obtain a code processing result of the target task.

Specifically, the target task refers to a code processing task, and the target task includes, but is not limited to, a code completion task, a code recommendation task, and a code comment generation task. The to-be-processed code data may be multilingual code data, such as Python or Java. The to-be-processed code data may be code text data, or may be code block data, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. The code block data may be referred to as code block (Code Block) data. A

code block refers to a block with complete code semantic logic, and may be a single line or a plurality of lines in length. Some code blocks have block headers, for example, a condition statement of an if block, and a method definition of a method definition block. The code block includes a coordinate range, to facilitate capturing block code from original code. Some code blocks are defined with unique names, to facilitate association between a referencing party and a definition via the unique name during reference restoration or test case association.

During actual application, there are a plurality of manners of obtaining the to-be-processed code data of the target task, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, the to-be-processed code data of the target task sent by a user may be received. In another possible implementation of the present disclosure, the to-be-processed code data of the target task may be read from another data obtaining device or database.

Step 304: input the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

In one or more embodiments of the present disclosure, after the to-be-processed code data of the target task is obtained, the to-be-processed code data may be further input into the code processing model, to obtain the code processing result of the target task.

Specifically, the code processing model may be a pre-trained large code model, or may be a machine learning model obtained by training a large model based on the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results. The code processing model is, for example, a bidirectional encoder representation-based deep self-attention model (BERT, Bidirectional Encoder Representations from Transformers) or a text-to-text deep self-attention model (T5, Text-to-Text Transfer Transformer). A task processing result corresponds to the target task. If the target task is the code completion task, the task processing result is a code completion result. If the target task is the code comment generation task, the task processing result is a code commenting result.

It should be noted that the code processing model includes an encoding unit and a decoding unit. The to-be-processed code data is input into the code processing model. First, the encoding unit captures a complex dependency between elements inside the to-be-processed

code data through a self-attention mechanism, and converts the to-be-processed code data layer by layer into a high-level semantic embedding representation, to obtain an encoded vector. Next, the decoding unit progressively decodes, through the self-attention mechanism and a mask mechanism, the encoded vector provided by the encoding unit, to obtain the code processing result.

In an optional embodiment of the present disclosure, before the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, syntax parsing may be further performed on the to-be-processed code data, to obtain a plurality of pieces of to-be-processed code block data. Further, the plurality of pieces of to-be-processed code block data may be input into the code processing model, to obtain the code processing result of the target task.

According to the solution in the embodiments of the present disclosure, because the code processing model is obtained by performing training based on the sample code block data, prediction is performed by using a code block as a unit, thereby shortening predicted content. In addition, through hierarchical sampling, the code processing model can be adapted to various code processing scenarios, thereby accurately predicting various types of code blocks, and improving task processing efficiency and accuracy.

During actual application, when the to-be-processed code data is input into the code processing model, the to-be-processed code data may be directly input into the code processing model, to obtain the code processing result of the target task. Further, to ensure that the code processing model can accurately process the to-be-processed code data, processing prompt information of the code processing model may be further obtained, and the to-be-processed code data is standardized by using the processing prompt information.

In an optional embodiment of the present disclosure, the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task may include the following steps:

obtaining the processing prompt information of the code processing model; and

inputting the processing prompt information and the to-be-processed code data into the code processing model, to obtain the code processing result of the target task.

Specifically, the processing prompt information is used for prompting and guiding the code processing model to process the to-be-processed code data. The processing prompt information may include code metadata, such as a code file name. The processing prompt information is specifically set based on an actual situation. This is not limited in the embodiments of the present disclosure. The code processing model can learn a language type

of the to-be-processed code data via the code file name, to accurately process the to-be-processed code data based on syntax information corresponding to the language type.

During actual application, there are a plurality of manners of obtaining the processing prompt information of the code processing model, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, the processing prompt information of the code processing model sent by the user may be received. In another possible implementation of the present disclosure, the processing prompt information of the code processing model may be read from another data obtaining device or database.

According to the solution in the embodiments of the present disclosure, the processing prompt information of the code processing model is obtained; and the processing prompt information and the to-be-processed code data are input into the code processing model, to obtain the code processing result of the target task. The processing prompt information is input into the code processing model, so that the code processing model learns more about the to-be-processed code data, thereby ensuring accuracy of the code processing result.

In an optional embodiment of the present disclosure, after the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method may further include the following steps:

sending the code processing result to a front-end user;

receiving result feedback information sent by the front-end user, where the result feedback information is information for feeding back the code processing result based on task information of the target task;

constructing model optimization data based on the result feedback information; and

performing parameter adjustment on the code processing model by using the model optimization data.

Specifically, the result feedback information may be information for feeding back content, quality, and completeness of the code processing result, which reflects actual feelings and expectations of the front-end user on the code processing result. The result feedback information includes, but is not limited to, result quality evaluation information, an accurate code processing result after correction, and a model optimization field. The model optimization data refers to accurate optimization sample data for optimizing the code processing model.

During actual application, there are a plurality of manners of constructing the model optimization data based on the result feedback information, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a

possible implementation of the present disclosure, the model optimization data may be constructed directly based on the result feedback information. In another possible implementation of the present disclosure, optimization prompt information may be generated based on the result feedback information, and the model optimization data sent by the front-end user based on the optimization prompt information is received.

It should be noted that, an example in which the model optimization data is constructed directly based on the result feedback information is used. If the result feedback information is the accurate accurate code processing result after correction, the model optimization data may be constructed based on the to-be-processed code data of the target task and the accurate code processing result after correction. If the result feedback information is the model optimization field, for example, the XXX field, sample code data in the XXX field may be obtained, and the sample code data in the XXX field is determined as the model optimization data. A process of performing parameter adjustment on the code processing model by using the model optimization data is the same as a process of training the code processing model. Details are not described again in the embodiments of the present disclosure.

According to the solution in the embodiments of the present disclosure, the code processing result is sent to the front-end user; the result feedback information sent by the front-end user is received, where the result feedback information is the information for feeding back the code processing result based on the task information of the target task; the model optimization data is constructed based on the result feedback information; and parameter adjustment is performed on the code processing model by using the model optimization data. The result feedback information is collected and used, to continuously optimize performance of the code processing model, thereby more accurately meeting an actual requirement of the front-end user and improving quality and accuracy of a final code processing result.

In an optional embodiment of the present disclosure, the constructing the model optimization data based on the result feedback information may include the following steps:

generating optimization prompt information based on the result feedback information, where the optimization prompt information is used for guiding the front-end user to send the model optimization data for optimizing the code processing model; and

sending the optimization prompt information to the front-end user, and receiving the model optimization data sent by the front-end user based on the optimization prompt information.

It should be noted that, there are a plurality of manners of generating the optimization prompt information based on the result feedback information, and a specific choice depends on

an actual situation. This is not limited in the embodiments of the present disclosure.

In a possible implementation of the present disclosure, preset prompt information may be directly obtained, and the result feedback information is added to the preset prompt information, to obtain the optimization prompt information. For example, the preset prompt information is "I sincerely apologize for any inaccurate information I have provided. Please point out specific inaccuracies or provide the correct answers to the relevant questions, and I will revise and refine my response as soon as possible to better serve you". If the result feedback information is "inaccurate result", the optimization prompt information is "As for your feedback on the inaccurate result, I sincerely apologize for any inaccurate information I have provided. Please point out specific inaccuracies or provide the correct answers to the relevant questions, and I will revise and refine my response as soon as possible to better serve you".

In another possible implementation of the present disclosure, type recognition may be performed on the result feedback information, to determine an information type of the result feedback information, the information type is further matched with a prompt type of each piece of prompt information in a prompt information library, and prompt information whose prompt type is the same as the information type is determined as the optimization prompt information.

According to the solution in the embodiments of the present disclosure, the optimization prompt information is generated based on the result feedback information, where the optimization prompt information is used for guiding the front-end user to send the model optimization data for optimizing the code processing model; and the optimization prompt information is sent to the front-end user, and the model optimization data sent by the front-end user based on the optimization prompt information is received. The model optimization data is obtained in an interactive guiding manner, thereby improving interactivity with the user, and improving user satisfaction.

In an optional embodiment of the present disclosure, after the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method may further include the following steps:

annotating key information in the code processing result, to obtain an updated code processing result; and

sending the updated code processing result to a front-end user.

Specifically, the key information refers to information in the code processing result that helps the user understand, debug, and maintain the code, such as a function, a class, or an object.

During actual application, before the annotating the key information in the code processing result, to obtain the updated code processing result, key information recognition

may be performed on the code processing result, to determine the key information in the code processing result. There are a plurality of manners of performing key information recognition on the code processing result, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, the code processing result may be matched with preset key information, and information that is the same as the preset key information and that appears in the code processing result is determined as the key information. In another possible implementation of the present disclosure, the code processing result may be input into a key information recognition model, to obtain the key information, where the key information recognition model is obtained by performing training based on a plurality of pieces of training code data and a key information tag of each piece of training code data.

It should be noted that, when the key information in the code processing result is annotated, operations such as bolding, highlighting, italicizing, and adding a border may be performed on key text information. A manner of annotating the key information is specifically selected based on an actual situation. This is not limited in the embodiments of the present disclosure.

According to the solution in the embodiments of the present disclosure, the key information in the code processing result is annotated, to obtain the updated code processing result; and the updated code processing result is sent to the front-end user, so that the front-end user can view the result, thereby improving use experience.

In an optional embodiment of the present disclosure, after the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method may further include the following steps:

sending the code processing result to a front-end user;

receiving modification information sent by the front-end user, where the modification information is used for modifying the code processing result; and

modifying the code processing result based on the modification information, to obtain a modified code processing result.

Specifically, the modification information is used for describing a modification requirement of the front-end user on the code processing result. The modification requirement includes, but is not limited to, a requirement for adding a comment and a requirement for generating test information. Further, the modification information may be modification requirement information for the key information in the code processing result.

During actual application, there are a plurality of manners of modifying the code processing result based on the modification information, and a specific choice depends on an

actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, a code modification tool (for example, a text editor) may be invoked to modify the code processing result based on the modification information. In another possible implementation of the present disclosure, the task processing result and the modification information may be input into a modification content generation model, to obtain modification content corresponding to the modification information, and the code processing result may be updated by using the modification content, to obtain the modified code processing result, where the modification content generation model is obtained by performing training based on a plurality of pieces of sample code, sample modification information of each piece of sample code, and an actual modification result.

According to the solution in the embodiments of the present disclosure, the code processing result is sent to the front-end user; the modification information sent by the front-end user is received, where the modification information is used for modifying the code processing result; and the code processing result is modified based on the modification information, to obtain the modified code processing result. The code processing result is modified based on the modification information sent by the front-end user, so that human-computer interactivity is improved, and task processing adaptability and flexibility are also improved.

In an optional embodiment of the present disclosure, a training manner of the code processing model is described. In other words, before the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method may further include the following steps:

obtaining a sample set, where the sample set includes a plurality of pieces of sample code data;

performing hierarchical sampling on the plurality of pieces of sample code data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results;

inputting the plurality of pieces of sample code block data into the code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and

training the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

Specifically, the training manner of the code processing model is supervised training. In

other words, the sample code block data carries a real processing tag (a sample processing result), and the sample processing result is a generation target of the code processing model and is used for guiding a training process of the code processing model. The sample code block data may include a code block prefix and a code block suffix of the sample processing result. The model may predict the sample processing result by using the sample code block data. The code processing model may be a pre-trained large model, or may be an untrained neural network model of a natural language. The sample code data may be referred to as source code data. The sample code data relates to various programming languages and fields. In this way, the model can learn from abundant source code data, so that the data exposed to the model in the training process is close to real development scenarios of developers. In addition, a knowledge density and a generalization capability of the model are improved, thereby enabling the model to provide valuable code processing suggestions for any developer in any development scenario.

It should be noted that in a real code processing scenario, there is a mutual nesting relationship between code blocks. For example, a SortAlgorithm class includes two functions, and the first function further includes content such as two statements and one logic block. Data features of nested code blocks belong to different levels. The data features of different levels may be considered as a type of variable. Various combinations may be needed for each value case of these variables. During sampling, these variables and combination cases thereof may be considered as much as possible, thereby simulating more diversified context scenarios and improving generalization of sampled model training data.

Therefore, to enable the model to provide valuable code processing suggestions in various development scenarios, the embodiments of the present disclosure provide a hierarchical sampling manner. In other words, hierarchical sampling is performed based on data features of different levels of code, to construct model training data (sample code block data and a sample processing result) in various languages and various context scenarios. During hierarchical sampling, the data features of different levels may be randomly sampled, or may be intentionally sampled based on an actual requirement of the user. The model training data obtained through sampling may be output in a form of <prefix+suffix,middle>, for continuous training of the model. For example, the data features of different levels include a processing topic and a context scenario. When hierarchical sampling is performed on a plurality of pieces of sample code data, a code processing topic may be first randomly sampled, and then, regardless of a code processing topic, one piece of code block data needs to be found as a model generation target (a sample processing result). In other words, under a code processing topic

obtained through sampling, a model generation target may be determined by randomly sampling a context scenario.

Further, after the performing hierarchical sampling on the plurality of pieces of sample code data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results, the plurality of pieces of sample code block data may be directly input into the code processing model, to obtain the prediction processing results respectively corresponding to the plurality of pieces of sample code block data. The sample processing result and the sample code block data corresponding to the sample processing result may be further organized into model training data based on a preset format, and the code processing model is continuously trained by using the model training data. Then, a code processing model on which the training is completed is deployed in an intelligent coding assistant. The preset format is, for example, "<filename> code file name <fim_prefix> code block prefix <fim_suffix> code block suffix <fim_middle> target code block", the code block prefix and the code block suffix are the sample code block data, and the target code block is the sample processing result.

It should be noted that, there are a plurality of manners of obtaining the sample set, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, a large amount of sample code data may be read from another data obtaining device or database (for example, The Stack dataset including source code data of projects in various languages). In another possible implementation of the present disclosure, a large amount of sample code data may be crawled from a network.

During actual application, when the code processing model is trained based on the sample processing result and the prediction processing result, a loss value may be calculated based on the sample processing result and the prediction processing result, and model parameters of the code processing model are adjusted based on the loss value until the training process meets a preset stop condition, to obtain a code processing model on which the training is completed. There are many functions for calculating the loss value, such as a cross-entropy loss function, an L1 norm loss function, a maximum loss function, a mean square error loss function, and a logarithm loss function, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. The preset stop condition includes, but is not limited to, the loss value being less than or equal to a preset threshold, and a quantity of iterations reaching a preset quantity of iterations. The preset threshold and the preset quantity of iterations are specifically selected based on an actual situation. This is not limited in the

embodiments of the present disclosure.

In a possible implementation of the present disclosure, after the loss value is calculated, the loss value is compared with the preset threshold. Specifically, if the loss value is greater than the preset threshold, it indicates that a difference between the sample processing result and the prediction processing result is great, and a prediction capability of the code processing model for the sample code block data is poor. In this case, the model parameters of the code processing model may be adjusted, and the step of inputting the plurality of pieces of sample code block data into the code processing model, to obtain the prediction processing results respectively corresponding to the plurality of pieces of sample code block data is performed again. The code processing model continues to be trained until the loss value is less than or equal to the preset threshold, indicating that the difference between the sample processing result and the prediction processing result is small, the preset stop condition is reached, and a code processing model on which the training is completed is obtained.

In another possible implementation of the present disclosure, in addition to comparing a magnitude relationship between the loss value and the preset threshold, whether the training on the current code processing model is completed may be further determined with reference to the quantity of iterations. Specifically, if the loss value is greater than the preset threshold, the model parameters of the code processing model are adjusted, and the step of inputting the plurality of pieces of sample code block data into the code processing model, to obtain the prediction processing results respectively corresponding to the plurality of pieces of sample code block data is performed again. The code processing model continues to be trained, and when the preset quantity of iterations is reached, the iteration is stopped, to obtain a code processing model on which the training is completed.

According to the solution in the embodiments of the present disclosure, the code processing model is trained based on the sample processing result and the prediction processing result, and if a preset stop condition is not met, the code processing model continues to be trained, until the preset stop condition is reached, to obtain a code processing model on which the training is completed. The model parameters of the code processing model are continuously adjusted, so that a finally obtained code processing model is more accurate.

During actual application, there are a plurality of manners of performing hierarchical sampling on the plurality of pieces of sample code data, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, hierarchical sampling may be directly performed on the sample code data, to obtain the plurality of sample processing results and the sample code

block data respectively corresponding to the plurality of sample processing results. In another possible implementation of the present disclosure, to ensure that hierarchical sampling can be performed by using a code block as a unit, hierarchical sampling may be performed on a plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results. The plurality of pieces of candidate code block data are obtained by performing syntax parsing on the sample code data.

In an optional embodiment of the present disclosure, the performing hierarchical sampling on the plurality of pieces of sample code data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results may include the following steps:

performing, for first sample code data, syntax parsing on the first sample code data, to obtain a plurality of pieces of candidate code block data, where the first sample code data is any one of the plurality of pieces of sample code data; and

performing hierarchical sampling on the plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results.

Specifically, syntax parsing is used for determining a plurality of code blocks of code data. Through the code blocks, the model can be enabled to identify which code segments in the code data are classes, which code segments are functions, which code segments are logic blocks, and the like. Further, the model can be enabled to have syntax tree awareness of generated results, so as to determine a required granularity of content to be generated.

During actual application, there are a plurality of manners of performing syntax parsing on the first sample code data, to obtain the plurality of pieces of candidate code block data, and a specific choice depends on an actual situation. This is not limited in the embodiments of the present disclosure. In a possible implementation of the present disclosure, the first sample code data may be matched with preset code block data, and data that is the same as the preset code block data and that appears in the first sample code data is determined as the candidate code block data. In another possible implementation of the present disclosure, syntax parsing may be performed on the first sample code data by using an AST parsing tool, to obtain the plurality of pieces of candidate code block data. An abstract syntax tree is a structured representation form of programming language source code.

It should be noted that, when hierarchical sampling is performed on the plurality of pieces of candidate code block data, a context-aware hierarchical sampling method may be used for

obtaining, through sampling, the sample processing results and the sample code block data corresponding to the sample processing results from the plurality of pieces of candidate code block data related to different topics, context scenarios, completeness levels, and completion granularities.

According to the solution in the embodiments of the present disclosure, for first sample code data, syntax parsing is performed on the first sample code data, to obtain the plurality of pieces of candidate code block data, where the first sample code data is any one of the plurality of pieces of sample code data; and hierarchical sampling is performed on the plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results. Through a method based on syntax parsing and hierarchical sampling of diversified scenarios, model training data in various languages and various context scenarios is constructed, so that the model training data better conforms to real development scenarios.

In an optional embodiment of the present disclosure, using a hierarchical sampling manner divided into a code processing topic and a code sampling strategy as an example, a specific implementation process of hierarchical sampling is described. In other words, the performing hierarchical sampling on the plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results may include the following steps:

obtaining a target code processing topic by sampling from code processing topics, where the code processing topics include at least one of a code block processing topic, a code block content processing topic, a code block partial content processing topic, and a code block empty content processing topic; and

obtaining the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the code sampling strategy under the target code processing topic.

Specifically, the code sampling strategy refers to a strategy for performing sampling based on a code context scenario. The code context scenario refers to a scenario including code context content. The code processing topic refers to a processing topic of code data sampled each time. The code processing topic may be used for describing a target processing range of the model for the code data. Using a code completion topic as an example, the code completion topic includes at least one of a code block completion topic, a code block content completion topic, a code block partial content completion topic, and a code block empty content

completion topic.

The code block processing topic represents that under the topic, the target processing range of the model is a single code block. Using the code block completion topic as an example, under the code block completion topic, an objective of the model is to complete a complete code block, for example, complete a complete function, or complete a complete logic block.

The code block content processing topic represents that under the topic, the target processing range of the model is entire content inside a single code block. Using the code block content completion topic as an example, under the code block content completion topic, an objective of the model is to fully complete content inside a code block, that is, complete entire content, for example, complete statements in which a middle part is removed, to finally complete the entire content of the code block.

The code block partial content processing topic represents that under the topic, the target processing range of the model is partial content inside a single code block. Using the code block partial content completion topic as an example, under the code block partial content completion topic, an objective of the model is to complete partial content inside a code block. For example, if a current incomplete statement is "arr[]", only content within the square brackets [] is completed, and content outside the square brackets [] does not need to be completed.

The code block empty content processing topic represents that under the topic, the target processing range of the model is empty content, in other words, no content needs to be processed. Using the code block empty content completion topic as an example, under the code block empty content completion topic, an objective of the model is not to complete any content in a complete context. For example, if a current statement is "result = sort.bubble_sort(arr)", when a cursor is positioned after "arr" and before ")", the model predicts that the completion content is empty.

According to the solution in the embodiments of the present disclosure, the target code processing topic is obtained by sampling from the code processing topics; and the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results are obtained by sampling from the plurality of pieces of candidate code block data according to the code sampling strategy under the target code processing topic. Through the method based on hierarchical sampling of diversified scenarios, model training data under different languages (such as a Python language and a Java language) in different context scenarios is constructed, so that the model training data better conforms to real development scenarios.

During actual application, under the target code processing topic, when the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results are obtained by sampling from the plurality of pieces of candidate code block data according to the code sampling strategy, at least one strategy may be randomly selected from a plurality of code sampling strategies, and the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results are obtained by sampling from the plurality of pieces of candidate code block data by using the selected strategy. Further, to ensure accuracy of the sample processing result and the sample code block data obtained through sampling, the code sampling strategies may be divided into result sampling strategies for obtaining the sample processing result through sampling and code block sampling strategies for obtaining the sample code block data through sampling, so that at least one strategy is randomly selected from the result sampling strategies and at least one strategy is randomly selected from the code block sampling strategies, and the sample processing result and the sample code block data are respectively obtained through sampling by using the selected strategies.

In an optional embodiment of the present disclosure, the code sampling strategy includes a result sampling strategy and a code block sampling strategy, the result sampling strategy includes at least one of a category sampling strategy, a prefix category sampling strategy, a suffix category sampling strategy, a child node sampling strategy, and a completeness sampling strategy, and the code block sampling strategy includes at least one of a prefix content sampling strategy and a suffix content sampling strategy; and the obtaining the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the code sampling strategy under the target code processing topic may include the following steps:

under the target code processing topic, obtaining the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the result sampling strategy, and obtaining the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the code block sampling strategy.

The category sampling strategy refers to a strategy for sampling a sample processing result based on a category variable. The category includes a class, a function, a logic block, a comment, a statement, and the like. Through the category sampling strategy, a code block of a category may be selected from the plurality of pieces of candidate code block data as the sample

processing result.

The prefix category sampling strategy refers to a strategy for sampling a sample processing result based on a category variable of a preceding code block. Through the prefix category sampling strategy, a code block whose preceding code block belongs to a category may be selected from the plurality of pieces of candidate code block data as the sample processing result. For example, when the preceding code block is a comment and the sample processing result is a function, it represents that a coding scenario of the user is as follows: the user has written a function comment, and then needs to write a corresponding complete function.

The suffix category sampling strategy refers to a strategy for sampling a sample processing result based on a category variable of a next code block. Through the suffix category sampling strategy, a code block whose next code block belongs to a category may be selected from the plurality of pieces of candidate code block data as the sample processing result. For example, when the sample processing result is a comment and the next code block is a class, it represents that a coding scenario of the user is as follows: the user has written a class, and intends to add a documentation comment to the class.

The child node sampling strategy refers to a strategy for sampling a sample processing result depending on whether child nodes are included or based on a quantity of included child nodes. Through the child node sampling strategy, a code block that includes child nodes, does not include child nodes, or has a quantity of child nodes greater than a preset quantity may be selected from the plurality of pieces of candidate code block data as the sample processing result. For example, some code blocks include child nodes, and some code blocks do not include child nodes. During sampling, the two cases may be distinguished, so that model training data is available for each case. Further, quantities of child nodes included in code blocks may be different. A larger quantity of child nodes included in a code block indicates greater difficulty, and a smaller quantity of child nodes included in a code block indicates lower difficulty. For example, some functions may include only one return statement, and some functions may include dozens of child nodes such as various logic blocks, statements, and comments. Difficulties thereof are different. During sampling, both a simple code block and a complex code block can be sampled.

The completeness sampling strategy refers to a strategy for sampling a sample processing result based on a completeness level of a code block. Through the completeness sampling strategy, code blocks with different completeness levels may be selected from the plurality of pieces of candidate code block data as the sample processing results. For example, when the

completed sample processing result is a function, the following three cases may be encountered. Case 1: the user does not input any content of a to-be-completed function. Case 2: the user has input a tab (tab) of a to-be-completed function. Case 3: the user has input some content of a to-be-completed function. During coding of developers, any case may be encountered. Therefore, the code blocks with different completeness levels may be sampled as the sample processing results.

The prefix content sampling strategy refers to a strategy for sampling sample code block data based on prefix content of the sample processing result. The prefix content of the sample processing result may include the following: 1. original prefix content of the sample code data; 2. prefix content of the last K child nodes of the sample processing result; 3. prefix content of the last K sibling nodes of the sample processing result; and 4. empty prefix content. Based on different prefix content, different sample code block data corresponding to a same sample processing result may be obtained through sampling. K is a positive integer.

The suffix content sampling strategy refers to a strategy for sampling sample code block data based on suffix content of the sample processing result. The suffix content of the sample processing result may include the following: 1. original suffix content of the sample code data; 2. suffix content of the last K child nodes of the sample processing result; 3. suffix content of the last K sibling nodes of the sample processing result; and 4. empty suffix content. Based on different suffix content, different sample code block data corresponding to a same sample processing result may be obtained through sampling.

According to the solution in the embodiments of the present disclosure, under the target code processing topic, the plurality of sample processing results are obtained by sampling from the plurality of pieces of candidate code block data according to the result sampling strategy, and the sample code block data respectively corresponding to the plurality of sample processing results is obtained by sampling from the plurality of pieces of candidate code block data according to the code block sampling strategy. The category sampling strategy refers to a strategy for sampling a sample processing result based on a category variable. The category includes a class, a function, a logic block, a comment, a statement, and the like. Through the category sampling strategy, a code block of a category may be selected from the plurality of pieces of candidate code block data as the sample processing result. Through the target code processing topic, different processing ranges in code processing scenarios can be simulated; through the result sampling strategy, more diversified sample processing results that better conform to real development scenarios can be obtained through sampling; and through the code block sampling strategy, more diversified sample code block data that better conforms to real

development scenarios can be obtained through sampling, so that model training data better conforms to real development scenarios.

With reference to FIG. 4, the task processing method provided in the present disclosure is further described below by using an example in which the task processing method is applied to an automatic code completion scenario. FIG. 4 is a flowchart of a code completion method according to an embodiment of the present disclosure. The method specifically includes the following steps.

Step 402: obtain to-be-completed code data of a code completion task.

Step 404: input the to-be-completed code data into a code completion model, to obtain a code completion result of the code completion task, where the code completion model is obtained by performing training based on a plurality of sample completion results and sample code block data respectively corresponding to the plurality of sample completion results, and the sample completion results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

It should be noted that, an implementation of step 402 and step 404 is the same as an implementation of step 302 and step 304. For related descriptions of the to-be-completed code data and the code completion model, reference may made to the related descriptions of the to-be-processed code data and the code processing model. Details are not described again in the embodiments of the present disclosure.

According to the solution in the embodiments of the present disclosure, for capabilities required for the code completion model, model training data is constructed through a method based on code syntax analysis and hierarchical sampling of diversified scenarios, and a code completion model is obtained by training with the model training data. In this way, in various real development scenarios, the code completion model accurately predicts various types of code blocks that a user intends to input next (which may be a class, a function, a logic block, a comment, a statement, or the like) based on a code prefix, a code suffix, and some code metadata written by the user, so as to improve coding efficiency and experience of the user.

In an optional embodiment of the present disclosure, before the obtaining the to-be-completed code data of the code completion task, the method may further include the following steps:

sending completion prompt information to a front-end user, where the completion prompt information is used for guiding the front-end user to send the to-be-completed code data for processing of the code completion task; and

receiving the to-be-completed code data of the code completion task sent by the front-end

user based on the completion prompt information.

It should be noted that when the code completion task is started, the completion prompt information may be first sent to the front-end user, and the front-end user may learn about a processing procedure or a processing requirement of the code completion task based on the completion prompt information, to input to-be-completed code data that more conforms to an actual requirement. For example, code completion information may be "Hello, I'm your intelligent coding assistant, ready to help you complete all types of code. You can directly input to-be-completed code data you would like to complete".

Further, after the completion prompt information is sent to the front-end user, the to-be-completed code data of the code completion task sent by the front-end user based on the completion prompt information may be received, and the to-be-completed code data is input into the code completion model, to obtain the code completion result of the code completion task.

According to the solution in the embodiments of the present disclosure, the completion prompt information is sent to the front-end user, where the completion prompt information is used for guiding the front-end user to send the to-be-completed code data for processing of the code completion task; and the to-be-completed code data of the code completion task sent by the front-end user based on the completion prompt information is received. The front-end user is guided, by using the completion prompt information, to input to-be-completed code data that needs to be completed, thereby improving human-computer interactivity, and enabling the to-be-completed code data to accurately reflect an actual requirement of the user.

In an optional embodiment of the present disclosure, after the inputting the to-be-completed code data into the code completion model, to obtain the code completion result of the code completion task, the method may further include the following steps:

sending the code completion result to a front-end user;

receiving result feedback information sent by the front-end user, where the result feedback information is information for feeding back the code completion result based on task information of the code completion task; and

sending the result feedback information to a model training platform, where the model training platform is configured to perform parameter adjustment on the code completion model by using the result feedback information.

Specifically, the model training platform may be deployed in a terminal device, or may be deployed in a cloud device. A specific setting depends on an actual situation. This is not limited in the embodiments of the present disclosure. The model training platform provides

large-scale data processing and high-performance computing resources, to train, optimize, and deploy various machine learning models, especially deep learning models. On the model training platform, the user may upload data, select or customize an algorithm model, and efficiently perform model training and verification with a distributed computing capability. The model training platform may receive a task generation request from the terminal device, obtain a corresponding code processing model based on request information, and generate task information based on the code processing model. The model training platform may further quickly respond to requirements of different tasks, and invoke an appropriate code processing model to perform task processing, to finally generate a high-quality code processing result.

During actual application, after the result feedback information sent by the front-end user is received, the result feedback information may be sent to the model training platform, and the model training platform may perform parameter adjustment on the code completion model based on the result feedback information. For a manner in which the model training platform performs parameter adjustment on the code completion model based on the result feedback information, reference may made to the foregoing implementations of "constructing the model optimization data based on the result feedback information; and performing parameter adjustment on the code processing model by using the model optimization data". Details are not described again in the embodiments of the present disclosure.

According to the solution in the embodiments of the present disclosure, the model training platform performs parameter adjustment on the code completion model based on the result feedback information, so that accuracy and training efficiency of the code completion model can be ensured, model training costs can be reduced, and a convenient and efficient model training service can be provided for the user.

FIG. 5 is a flowchart of a code processing model training method according to an embodiment of the present disclosure. The method specifically includes the following steps.

Step 502: obtain a sample set, where the sample set includes a plurality of pieces of sample code data.

Step 504: perform hierarchical sampling on the plurality of pieces of sample code data, to obtain a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results.

Step 506: input the plurality of pieces of sample code block data into a code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data.

Step 508: train the code processing model based on the sample processing result and the

prediction processing result, to obtain a code processing model on which the training is completed.

It should be noted that, for implementations of step 502 to step 508, reference may made to the foregoing code processing model training manner. Details are not described again in the embodiments of the present disclosure.

During actual application, after the code processing model on which the training is completed is obtained, model parameters of the code processing model on which the training is completed may be sent to a terminal device, and a front-end user may locally construct the code processing model based on the model parameters, and use the code processing model to perform tasks such as code completion.

According to the solution in the embodiments of the present disclosure, the code processing model is trained based on the sample processing result and the prediction processing result, and if a preset stop condition is not met, the code processing model continues to be trained, until the preset stop condition is reached, to obtain a code processing model on which the training is completed. The model parameters of the code processing model are continuously adjusted, so that a finally obtained code processing model is more accurate.

FIG. 6 is a flowchart of a processing process of a code processing model training method according to an embodiment of the present disclosure. The method specifically includes the following steps.

Source code data is collected: source code data related to various programming languages and fields is collected as sample code data, so that a model can learn from abundant source code data, data exposed to a model training process can be ensured to be close to a real development scenario of a user, a knowledge density and a generalization capability of the model can be improved, and valuable completion suggestions can be provided for any developer in any development scenario.

AST syntax parsing is performed on the source code data: for first sample code data, syntax parsing is performed on the first sample code data, to obtain a plurality of pieces of candidate code block data, where the first sample code data is any one of the plurality of pieces of sample code data. Syntax parsing is performed on the source code data, so that the model can be enabled to identify which code segments in the code data are classes, which code segments are functions, which code segments are logic blocks, and the like. In addition, the model can be enabled to have syntax tree awareness of generated results, so as to determine a required granularity of content to be generated, to predict only content of a next code block (a function, a logic block, or a statement). In this way, the predicted content is greatly shortened

and accurately meets a requirement of the user, which provides more valuable code processing suggestions for the user while not interfering with a coding pace of the user.

Training data is constructed through multi-scenario hierarchical sampling: a target code processing topic is obtained by sampling from code processing topics, where the code processing topics include at least one of a code block processing topic, a code block content processing topic, a code block partial content processing topic, and a code block empty content processing topic. Under the target code processing topic, a plurality of sample processing results are obtained by sampling from the plurality of pieces of candidate code block data according to a result sampling strategy, and sample code block data respectively corresponding to the plurality of sample processing results are obtained by sampling from the plurality of pieces of candidate code block data according to a code block sampling strategy, where the result sampling strategy includes at least one of a category sampling strategy, a prefix category sampling strategy, a suffix category sampling strategy, a child node sampling strategy, and a completeness sampling strategy, and the code block sampling strategy includes at least one of a prefix content sampling strategy and a suffix content sampling strategy. Through the hierarchical sampling method, the plurality of pieces of candidate code block data obtained through syntax parsing are sampled, to construct model training data in various languages and various context scenarios.

The model is trained: a plurality of pieces of sample code block data are input into a code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and the code processing model is trained based on the sample processing results and the prediction processing results, to obtain a code processing model on which the training is completed.

According to the solution in the embodiments of the present disclosure, through the AST syntax parsing and the hierarchical sampling method, a logic block type that the user intends to process is accurately predicted under different topics and in different context scenarios, so that the model can predict code content of granularities such as a complete class, function, logic block, comment, and statement.

FIG. 7 is a flowchart of an information processing method based on a code processing model according to an embodiment of the present disclosure. The method specifically includes the following steps.

Step 702: receive a task generation request, where the task generation request includes request information.

Specifically, the information processing method based on the code processing model may

be applied to a terminal device, or may be applied to a model training platform. The task generation request is used for requesting generation of task information of a target task. The task generation request usually includes a task type, an expected output format, and the request information. For example, when selecting a "code completion" function on a front-end interface of the model training platform and uploading code data, a user may construct a task generation request. The task generation request includes information such as code data, a task type (that is, code completion), and a language type of the code data. The request information refers to parameters or description information that is related to the target task and that is carried in the task generation request. The request information includes, but is not limited to, a task scenario identifier and a task model identifier of the target task, or model training data of the target task.

Step 704: obtain the code processing model based on the request information, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

In an optional embodiment of the present disclosure, the request information includes the task scenario identifier or the task model identifier of the target task. The obtaining the code processing model based on the request information may include the following steps:

determining a target scenario-based template from a plurality of preset scenario-based templates based on the task scenario identifier, and searching a model library for the code processing model based on the target scenario-based template, where the model library stores a plurality of processing models;

or

searching the model library for the code processing model based on the task model identifier.

Specifically, the task scenario identifier refers to a unique or specific tag for distinguishing between different task application scenarios. In the embodiments of the present disclosure, the task scenario identifier is a part of the request information. The target scenario-based template matching the request information may be selected from a series of preset scenario-based templates based on the task scenario identifier, to generate the task information. For example, the task scenario identifier is "code completion", which means that a front-end user expects to perform code completion on the uploaded code data, and the front-end user may select a code completion scenario-based template from the plurality of preset scenario-based templates based on the task scenario identifier.

The preset scenario-based templates are standard configuration scenario-based templates predefined for different task application scenarios, and each template includes information such as model information matching the task application scenario and a task processing procedure. Task generation requests in different scenarios can be quickly responded to through the series of preset scenario-based templates. Different preset scenario-based templates correspond to different task types, model information, and processing procedures. For example, there may be a scenario-based template specifically for code completion in the preset scenario-based templates, and the scenario-based template includes model information and a processing procedure of a trained code completion model.

The target scenario-based template refers to a scenario-based template matching the task scenario identifier. When the task generation request is parsed, a corresponding target scenario-based template may be positioned based on the task scenario identifier, and a corresponding code processing model and other related configuration information are selected from the model library based on the model information included in the target scenario-based template. For example, when the task scenario identifier is "code completion", the target scenario-based template is a template including the model information of the code completion model and related configuration parameters.

The model library is a resource library in which deep learning models are stored in a centralized manner, and these models are trained and optimized to resolve different processing tasks. The processing models stored in the model library include, but are not limited to, a code completion model, a code comment generation model, and a code recommendation model. In addition, the processing models in the model library may be divided into different versions based on different application scenarios. For example, the model library may store code completion models of a plurality of versions, such as a code completion model for a Python language and a code completion model for a Java language.

The task model identifier refers to a unique or specific tag for distinguishing between models applicable to different tasks. For example, the task model identifier may be "code completion". Based on the task model identifier, the model library may be searched for the code completion model suitable for processing the code completion task.

According to the solution in the embodiments of the present disclosure, the target scenario-based template is determined from the plurality of preset scenario-based templates based on the task scenario identifier, and the model library is searched for the code processing model based on the target scenario-based template, where the model library stores a plurality of processing models. Alternatively, the model library is searched for the code processing

model based on the task model identifier. A task scenario-based template, a task model identifier, and model library resources are predefined, so that a process of obtaining the code processing model is more flexible, efficient, and standard.

In another optional embodiment of the present disclosure, in addition to selecting a pre-trained code processing model from the model library, the code processing model may be further obtained through targeted training based on the model training data in the request information. In other words, the request information includes a plurality of pieces of sample code data of the target task. The obtaining the code processing model based on the request information may include the following steps:

training the code processing model corresponding to the target task based on the plurality of pieces of sample code data, to obtain a code processing model on which the training is completed.

It should be noted that an implementation of "training the code processing model corresponding to the target task based on the plurality of pieces of sample code data, to obtain the code processing model on which the training is completed" is the same as the foregoing code processing model training manner. Details are not described again in the embodiments of the present disclosure.

According to the solution in the embodiments of the present disclosure, the code processing model corresponding to the target task is trained based on the plurality of pieces of sample code data, to obtain the code processing model on which the training is completed, thereby ensuring that the code processing model better conforms to a user requirement and accuracy of the code processing model.

Step 706: generate the task information based on the code processing model, where the task information is used for executing the target task.

Specifically, the task information includes a model configuration and a processing procedure required for executing the target task. The terminal device or another server component may correctly process, based on the task information, the target task by using the code processing model.

It should be noted that when the task information is generated based on the code processing model, the code processing model may be directly packaged to obtain the task information. Model information of the code processing model may be further obtained, and the task information is constructed based on the model information. The model information may be, for example, a model parameter configuration, a processing manner of input data, a specification of an expected output result, and intermediate steps and other auxiliary

information that may be involved.

For example, in the code completion task, the task information may include parameters such as address information of a selected code completion model, a storage location of input code data, a target path for output completion results, and other environment configurations required for the operation of the code completion model. Such information enables correct loading of the code completion model and execution of the code completion task on a local or remote server.

According to the solution in the embodiments of the present disclosure, the task information of the target task is generated, so that processing quality and efficiency of the target task can be ensured, system deployment and operation and maintenance costs can be reduced, and a convenient and efficient task processing service can be provided for the user.

FIG. 8 is a schematic structural diagram of a model training platform according to an embodiment of the present disclosure. The model training platform includes a request interface unit 802 and a model unit 804.

The request interface unit 802 is configured to receive a model training request, where the model training request includes request information; and

the model unit 804 is configured to obtain a code processing model; and train the code processing model based on the request information, to obtain a code processing model on which the training is completed, where the request information includes a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

It should be noted that a processing manner of the request interface unit and the model unit is the same as the foregoing code processing model training manner. Details are not described again in the embodiments of the present disclosure.

In an optional embodiment of the present disclosure, the model training platform further includes a model library; and

the model unit is further configured to store the code processing model on which the training is completed in the model library.

It should be noted that, after training the code processing model based on the request information, to obtain the code processing model on which the training is completed, the model unit may further store the code processing model on which the training is completed in the model library, so that subsequently the code processing model on which the training is completed may be directly obtained from the model library, and the training does not need to

be repeated. For a manner of obtaining the code processing model on which the training is completed from the model library, reference may be made to the foregoing implementations of "determining the target scenario-based template from the plurality of preset scenario-based templates based on the task scenario identifier, and searching the model library for the code processing model based on the target scenario-based template, where the model library stores the plurality of processing models; or searching the model library for the code processing model based on the task model identifier". Details are not described again in the embodiments of the present disclosure.

In correspondence to the foregoing task processing method embodiment, the present disclosure further provides a task processing apparatus embodiment. FIG. 9 is a schematic structural diagram of a task processing apparatus according to an embodiment of the present disclosure. As shown in FIG. 9, the apparatus includes:

a first obtaining module 902, configured to obtain to-be-processed code data of a target task; and

a first input module 904, configured to input the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

Optionally, the first input module 904 is further configured to: obtain processing prompt information of the code processing model; and input the processing prompt information and the to-be-processed code data into the code processing model, to obtain the code processing result of the target task.

Optionally, the apparatus further includes: a first sending module, configured to: send the code processing result to a front-end user; receive result feedback information sent by the front-end user, where the result feedback information is information for feeding back the code processing result based on task information of the target task; construct model optimization data based on the result feedback information; and perform parameter adjustment on the code processing model by using the model optimization data.

Optionally, the first sending module is further configured to: generate optimization prompt information based on the result feedback information, where the optimization prompt information is used for guiding the front-end user to send the model optimization data for optimizing the code processing model; and send the optimization prompt information to the

front-end user, and receive the model optimization data sent by the front-end user based on the optimization prompt information.

Optionally, the apparatus further includes: an annotation module, configured to: annotate key information in the code processing result, to obtain an updated code processing result; and send the updated code processing result to a front-end user.

Optionally, the apparatus further includes: a second sending module, configured to: send the code processing result to a front-end user; receive modification information sent by the front-end user, where the modification information is used for modifying the code processing result; and modify the code processing result based on the modification information, to obtain a modified code processing result.

Optionally, the apparatus further includes: a fifth obtaining module, configured to: obtain a sample set, where the sample set includes a plurality of pieces of sample code data; perform hierarchical sampling on the plurality of pieces of sample code data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results; input the plurality of pieces of sample code block data into the code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and train the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

Optionally, the fifth obtaining module is further configured to: perform, for first sample code data, syntax parsing on the first sample code data, to obtain a plurality of pieces of candidate code block data, where the first sample code data is any one of the plurality of pieces of sample code data; and perform hierarchical sampling on the plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results.

Optionally, the fifth obtaining module is further configured to: obtain a target code processing topic by sampling from code processing topics, where the code processing topics include at least one of a code block processing topic, a code block content processing topic, a code block partial content processing topic, and a code block empty content processing topic; and obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to a code sampling strategy under the target code processing topic.

Optionally, the code sampling strategy includes a result sampling strategy and a code

block sampling strategy, where the result sampling strategy includes at least one of a category sampling strategy, a prefix category sampling strategy, a suffix category sampling strategy, a child node sampling strategy, and a completeness sampling strategy, and the code block sampling strategy includes at least one of a prefix content sampling strategy and a suffix content sampling strategy; and the fifth obtaining module is further configured to: under the target code processing topic, obtain the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the result sampling strategy, and obtain the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the code block sampling strategy.

According to the solution in the embodiments of the present disclosure, because the code processing model is obtained by performing training based on the sample code block data, prediction is performed by using a code block as a unit, thereby shortening predicted content. In addition, through hierarchical sampling, the code processing model can be adapted to various code processing scenarios, thereby accurately predicting various types of code blocks, and improving task processing efficiency and accuracy.

The foregoing is a schematic solution of a task processing apparatus according to this embodiment. It should be noted that the technical solution of the task processing apparatus is based on the same concept as the technical solution of the foregoing task processing method. For details not described in detail in the technical solution of the task processing apparatus, reference may made to the descriptions of the technical solution of the foregoing task processing method.

In correspondence to the foregoing code completion method embodiment, the present disclosure further provides a code completion apparatus embodiment. FIG. 10 is a schematic structural diagram of a code completion apparatus according to an embodiment of the present disclosure. As shown in FIG. 10, the apparatus includes:

a second obtaining module 1002, configured to obtain to-be-completed code data of a code completion task; and

a second input module 1004, configured to input the to-be-completed code data into a code completion model, to obtain a code completion result of the code completion task, where the code completion model is obtained by performing training based on a plurality of sample completion results and sample code block data respectively corresponding to the plurality of sample completion results, and the sample completion results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

Optionally, the apparatus further includes: a second receiving module, configured to: send completion prompt information to a front-end user, where the completion prompt information is used for guiding the front-end user to send the to-be-completed code data for processing of the code completion task; and receive the to-be-completed code data of the code completion task sent by the front-end user based on the completion prompt information.

Optionally, the apparatus further includes: a third receiving module, configured to: send the code completion result to a front-end user; receive result feedback information sent by the front-end user, where the result feedback information is information for feeding back the code completion result based on task information of the code completion task; and send the result feedback information to a model training platform, where the model training platform is configured to perform parameter adjustment on the code completion model by using the result feedback information.

According to the solution in the embodiments of the present disclosure, for capabilities required for the code completion model, model training data is constructed through a method based on code syntax analysis and hierarchical sampling of diversified scenarios, and a code completion model is obtained by training with the model training data. In this way, in various real development scenarios, the code completion model accurately predicts various types of code blocks that a user intends to input next (which may be a class, a function, a logic block, a comment, a statement, or the like) based on a code prefix, a code suffix, and some code metadata written by the user, so as to improve coding efficiency and experience of the user.

The foregoing is a schematic solution of a code completion apparatus according to this embodiment. It should be noted that the technical solution of the code completion apparatus is based on the same concept as the technical solution of the foregoing code completion method. For details not described in detail in the technical solution of the code completion apparatus, reference may made to the descriptions of the technical solution of the foregoing code completion method.

In correspondence to the foregoing code processing model training method embodiment, the present disclosure further provides a code processing model training apparatus embodiment. FIG. 11 is a schematic structural diagram of a code processing model training apparatus according to an embodiment of the present disclosure. As shown in FIG. 11, the apparatus includes:

a third obtaining module 1102, configured to: obtain a sample set, where the sample set includes a plurality of pieces of sample code data;

a sampling module 1104, configured to perform hierarchical sampling on the plurality of

pieces of sample code data, to obtain a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results;

a third input module 1106, configured to input the plurality of pieces of sample code block data into a code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and

a training module 1108, configured to train the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

According to the solution in the embodiments of the present disclosure, the code processing model is trained based on the sample processing result and the prediction processing result, and if a preset stop condition is not met, the code processing model continues to be trained, until the preset stop condition is reached, to obtain a code processing model on which the training is completed. The model parameters of the code processing model are continuously adjusted, so that a finally obtained code processing model is more accurate.

The foregoing is a schematic solution of a code processing model training apparatus according to this embodiment. It should be noted that the technical solution of the code processing model training apparatus is based on the same concept as the technical solution of the foregoing code processing model training method. For details not described in detail in the technical solution of the code processing model training apparatus, reference may made to the descriptions of the technical solution of the foregoing code processing model training method.

In correspondence to the foregoing method embodiment, the present disclosure further provides an embodiment of an information processing apparatus based on a code processing model. FIG. 12 is a schematic structural diagram of an information processing apparatus based on a code processing model according to an embodiment of the present disclosure. As shown in FIG. 12, the apparatus includes:

a first receiving module 1202, configured to receive a task generation request, where the task generation request includes request information;

a fourth obtaining module 1204, configured to obtain the code processing model based on the request information, where the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data; and

a generation module 1206, configured to generate task information based on the code

processing model, where the task information is used for executing a target task.

Optionally, the request information includes a task scenario identifier or a task model identifier of the target task. The fourth obtaining module 1204 is further configured to: determine a target scenario-based template from a plurality of preset scenario-based templates based on the task scenario identifier, and search a model library for the code processing model based on the target scenario-based template, where the model library stores a plurality of processing models; or search the model library for the code processing model based on the task model identifier.

Optionally, the request information includes a plurality of pieces of sample code data of the target task. The fourth obtaining module 1204 is further configured to train the code processing model corresponding to the target task based on the plurality of pieces of sample code data, to obtain a code processing model on which the training is completed.

According to the solution in the embodiments of the present disclosure, the task information of the target task is generated, so that processing quality and efficiency of the target task can be ensured, system deployment and operation and maintenance costs can be reduced, and a convenient and efficient task processing service can be provided for the user.

The foregoing is a schematic solution of an information processing apparatus based on a code processing model according to this embodiment. It should be noted that the technical solution of the information processing apparatus based on the code processing model is based on the same concept as the technical solution of the foregoing information processing method based on the code processing model. For details not described in detail in the technical solution of the information processing apparatus based on the code processing model, reference may made to the descriptions of the technical solution of the foregoing information processing method based on the code processing model.

FIG. 13 is a structural block diagram of a computing device according to an embodiment of the present disclosure. Components of the computing device 1300 include, but are not limited to, a memory 1310 and a processor 1320. The processor 1320 is connected to the memory 1310 via a bus 1330, and a database 1350 is configured to save data.

The computing device 1300 further includes an access device 1340. The access device 1340 enables the computing device 1300 to perform communication via one or more networks 1360. Examples of these networks include a public switched telephone network (PSTN, Public Switched Telephone Network), a local area network (LAN, Local Area Network), a wide area network (WAN, Wide Area Network), a personal area network (PAN, Personal Area Network), or a combination of communication networks such as the Internet. The access device 1340 may

include one or more of any type of wired or wireless network interfaces (for example, a network interface card (NIC, Network Interface Card)), such as an IEEE802.11 wireless local area network (WLAN, Wireless Local Area Network) wireless interface, a worldwide interoperability for microwave access (Wi-MAX, World Interoperability for Microwave Access) interface, an Ethernet interface, a universal serial bus (USB, Universal Serial Bus) interface, a cellular network interface, a Bluetooth interface, and a near field communication (NFC, Near Field Communication) interface.

In an embodiment of the present disclosure, the foregoing components of the computing device 1300 and other components not shown in FIG. 13 may also be connected to each other, for example, via the bus. It should be understood that the structural block diagram of the computing device shown in FIG. 13 is merely used for exemplary purposes, but is not intended to limit the scope of the present disclosure. A person skilled in the art may add other components or replace the components with other components as required.

The computing device 1300 may be any type of stationary or mobile computing device, including a mobile computer or a mobile computing device (such as a tablet computer, a personal digital assistant, a laptop computer, a notebook computer, or a netbook), a mobile phone (such as a smartphone), a wearable computing device (such as a smartwatch or smart glasses), or another type of mobile device, or a stationary computing device such as a desktop computer or a personal computer (PC, Personal Computer). The computing device 1300 may alternatively be a mobile or stationary server.

The processor 1320 is configured to execute a computer program or computer instructions. The computer program or the computer instructions, when executed by the processor, implement steps of the task processing method, the code completion method, the code processing model training method, or the information processing method based on the code processing model.

The foregoing is a schematic solution of a computing device according to this embodiment. It should be noted that the technical solution of the computing device is based on the same concept as the technical solutions of the task processing method, the code completion method, the code processing model training method, and the information processing method based on the code processing model. For details that are not described in detail in the technical solution of the computing device, reference may made to the descriptions of the technical solutions of the task processing method, the code completion method, the code processing model training method, or the information processing method based on the code processing model.

An embodiment of the present disclosure further provides a computer-readable storage

medium, storing a computer program or computer instructions. The computer program or the computer instructions, when executed by a processor, implement steps of the task processing method, the code completion method, the code processing model training method, or the information processing method based on the code processing model.

The foregoing is a schematic solution of a computer-readable storage medium according to this embodiment. It should be noted that the technical solution of the storage medium is based on the same concept as the technical solutions of the task processing method, the code completion method, the code processing model training method, and the information processing method based on the code processing model. For details that are not described in detail in the technical solution of the storage medium, reference may made to the descriptions of the technical solutions of the task processing method, the code completion method, the code processing model training method, or the information processing method based on the code processing model.

An embodiment of the present disclosure further provides a computer program product, including a computer program or computer instructions. The computer program or the computer instructions, when executed by a processor, implement steps of the task processing method, the code completion method, the code processing model training method, or the information processing method based on the code processing model.

The foregoing is a schematic solution of a computer program product according to this embodiment. It should be noted that the technical solution of the computer program product is based on the same concept as the technical solutions of the task processing method, the code completion method, the code processing model training method, and the information processing method based on the code processing model. For details that are not described in detail in the technical solution of the computer program product, reference may made to the descriptions of the technical solutions of the task processing method, the code completion method, the code processing model training method, or the information processing method based on the code processing model.

Specific embodiments of the present disclosure are described above. Other embodiments fall within the scope of the appended claims. In some embodiments, actions or steps recorded in the claims can be performed in sequences different from those in the embodiments and may still achieve an expected result. In addition, the processes depicted in the accompanying drawings are not necessarily performed in the specific sequence or the successive sequence shown in the figures to achieve an expected result. In some implementations, multitasking and parallel processing may be feasible or beneficial.

The computer instructions include computer program code. The computer program code may be in the form of source code, in the form of object code, in the form of an executable file, in some intermediate forms, or the like. The computer-readable medium may include: any entity or apparatus that is capable of carrying the computer program code, a recording medium, a USB flash drive, a removable hard disk, a magnetic disk, an optical disc, a computer memory, a read-only memory (ROM, Read-Only Memory), a random access memory (RAM, Random Access Memory), an electric carrier signal, a telecommunication signal, and a software distribution medium. It should be noted that content included in the computer-readable medium may be properly increased or reduced based on requirements of patent practices. For example, in some regions, according to patent practices, the computer-readable medium does not include the electrical carrier signal or the telecommunication signal.

It should be noted that for the foregoing method embodiments, for ease of description, the method embodiments are described as a series of action combinations, but a person skilled in the art should learn that the embodiments of the present disclosure are not limited to a sequence of described actions. Because according to the embodiments of the present disclosure, some steps can be performed in another sequence or at the same time. A person skilled in the art should also be aware that the embodiments described in the specification are preferred embodiments, and that the actions and modules involved are not necessarily required for the embodiments of the present disclosure.

In the foregoing embodiments, the descriptions of the embodiments have respective emphasis. For a part that is not described in detail in an embodiment, reference may made to the related descriptions in other embodiments.

The preferred embodiments of the present disclosure disclosed above are only used for assisting in elaboration of the present disclosure. The optional embodiments do not describe all details in detail, and do not limit the present invention to being only the specific implementations. It is apparent that many modifications and variations can be made based on the content of the embodiments of the present disclosure. In the present disclosure, these embodiments are selected and described in detail to better explain the principle and the actual application of the embodiments of the present disclosure, so that a person skilled in the art can understand and use the present disclosure well. The present disclosure is limited only by the claims and their full scope and equivalents.

CLAIMS

1. A task processing method, comprising:

obtaining to-be-processed code data of a target task; and

inputting the to-be-processed code data into a code processing model, to obtain a code processing result of the target task, wherein the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

2. The method according to claim 1, wherein the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task comprises:

obtaining processing prompt information of the code processing model; and

inputting the processing prompt information and the to-be-processed code data into the code processing model, to obtain the code processing result of the target task.

3. The method according to claim 1 or 2, wherein after the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method further comprises:

sending the code processing result to a front-end user;

receiving result feedback information sent by the front-end user, wherein the result feedback information is information for feeding back the code processing result based on task information of the target task;

constructing model optimization data based on the result feedback information; and

performing parameter adjustment on the code processing model by using the model optimization data.

4. The method according to claim 3, wherein the constructing the model optimization data based on the result feedback information comprises:

generating optimization prompt information based on the result feedback information, wherein the optimization prompt information is used for guiding the front-end user to send the model optimization data for optimizing the code processing model; and

sending the optimization prompt information to the front-end user, and receiving the model optimization data sent by the front-end user based on the optimization prompt information.

5. The method according to any one of claims 1 to 4, wherein after the inputting the to-

be-processed code data into the code processing model, to obtain the code processing result of the target task, the method further comprises:

annotating key information in the code processing result, to obtain an updated code processing result; and

sending the updated code processing result to a front-end user.

6. The method according to any one of claims 1 to 5, wherein after the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method further comprises:

sending the code processing result to a front-end user;

receiving modification information sent by the front-end user, wherein the modification information is used for modifying the code processing result; and

modifying the code processing result based on the modification information, to obtain a modified code processing result.

7. The method according to any one of claims 1 to 6, wherein before the inputting the to-be-processed code data into the code processing model, to obtain the code processing result of the target task, the method further comprises:

obtaining a sample set, wherein the sample set comprises a plurality of pieces of sample code data;

performing hierarchical sampling on the plurality of pieces of sample code data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results;

inputting the plurality of pieces of sample code block data into the code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and

training the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

8. The method according to claim 7, wherein the performing hierarchical sampling on the plurality of pieces of sample code data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results comprises:

performing, for first sample code data, syntax parsing on the first sample code data, to obtain a plurality of pieces of candidate code block data, wherein the first sample code data is any one of the plurality of pieces of sample code data; and

performing hierarchical sampling on the plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results.

9. The method according to claim 8, wherein the performing hierarchical sampling on the plurality of pieces of candidate code block data, to obtain the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results comprises:

obtaining a target code processing topic by sampling from code processing topics, wherein the code processing topics comprise at least one of a code block processing topic, a code block content processing topic, a code block partial content processing topic, and a code block empty content processing topic; and

obtaining the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to a code sampling strategy under the target code processing topic.

10. The method according to claim 9, wherein the code sampling strategy comprises a result sampling strategy and a code block sampling strategy, the result sampling strategy comprises at least one of a category sampling strategy, a prefix category sampling strategy, a suffix category sampling strategy, a child node sampling strategy, and a completeness sampling strategy, and the code block sampling strategy comprises at least one of a prefix content sampling strategy and a suffix content sampling strategy; and

the obtaining the plurality of sample processing results and the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the code sampling strategy under the target code processing topic comprises:

under the target code processing topic, obtaining the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the result sampling strategy, and obtaining the sample code block data respectively corresponding to the plurality of sample processing results by sampling from the plurality of pieces of candidate code block data according to the code block sampling strategy.

11. A code completion method, comprising:

obtaining to-be-completed code data of a code completion task; and

inputting the to-be-completed code data into a code completion model, to obtain a code completion result of the code completion task, wherein the code completion model is obtained

by performing training based on a plurality of sample completion results and sample code block data respectively corresponding to the plurality of sample completion results, and the sample completion results and the sample code block data are obtained by performing hierarchical sampling on sample code data.

12. The method according to claim 11, wherein before the obtaining the to-be-completed code data of the code completion task, the method further comprises:

sending completion prompt information to a front-end user, wherein the completion prompt information is used for guiding the front-end user to send the to-be-completed code data for processing of the code completion task; and

receiving the to-be-completed code data of the code completion task sent by the front-end user based on the completion prompt information.

13. The method according to claim 11 or 12, wherein after the inputting the to-be-completed code data into the code completion model, to obtain the code completion result of the code completion task, the method further comprises:

sending the code completion result to a front-end user;

receiving result feedback information sent by the front-end user, wherein the result feedback information is information for feeding back the code completion result based on task information of the code completion task; and

sending the result feedback information to a model training platform, wherein the model training platform is configured to perform parameter adjustment on the code completion model by using the result feedback information.

14. A code processing model training method, comprising:

obtaining a sample set, wherein the sample set comprises a plurality of pieces of sample code data;

performing hierarchical sampling on the plurality of pieces of sample code data, to obtain a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results;

inputting the plurality of pieces of sample code block data into a code processing model, to obtain prediction processing results respectively corresponding to the plurality of pieces of sample code block data; and

training the code processing model based on the sample processing result and the prediction processing result, to obtain a code processing model on which the training is completed.

15. An information processing method based on a code processing model, comprising:

receiving a task generation request, wherein the task generation request comprises request information;

obtaining the code processing model based on the request information, wherein the code processing model is obtained by performing training based on a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data are obtained by performing hierarchical sampling on sample code data; and

generating task information based on the code processing model, wherein the task information is used for executing a target task.

16. The method according to claim 15, wherein the request information comprises a task scenario identifier or a task model identifier of the target task; and

the obtaining the code processing model based on the request information comprises:

determining a target scenario-based template from a plurality of preset scenario-based templates based on the task scenario identifier, and searching a model library for the code processing model based on the target scenario-based template, wherein the model library stores a plurality of processing models;

or

searching the model library for the code processing model based on the task model identifier.

17. The method according to claim 15, wherein the request information comprises a plurality of pieces of sample code data of the target task; and

the obtaining the code processing model based on the request information comprises:

training the code processing model corresponding to the target task based on the plurality of pieces of sample code data, to obtain a code processing model on which the training is completed.

18. A model training platform, comprising a request interface unit and a model unit, wherein

the request interface unit is configured to receive a model training request, wherein the model training request comprises request information; and

the model unit is configured to obtain a code processing model; and train the code processing model based on the request information, to obtain a code processing model on which the training is completed, wherein the request information comprises a plurality of sample processing results and sample code block data respectively corresponding to the plurality of sample processing results, and the sample processing results and the sample code block data

are obtained by performing hierarchical sampling on sample code data.

19. The model training platform according to claim 18, further comprising a model library, wherein

the model unit is further configured to store the code processing model on which the training is completed in the model library.

20. A computing device, comprising:

a memory and a processor, wherein

the memory is configured to store a computer program or computer instructions, the processor is configured to execute the computer program or the computer instructions, and the computer program or the computer instructions, when executed by the processor, implement steps of the method according to any one of claims 1 to 10, any one of claims 11 to 13, claim 14, or any one of claims 15 to 17.

21. A computer-readable storage medium, storing a computer program or computer instructions, wherein the computer program or the computer instructions, when executed by a processor, implement steps of the method according to any one of claims 1 to 10, any one of claims 11 to 13, claim 14, or any one of claims 15 to 17.

22. A computer program product, comprising a computer program or computer instructions, wherein the computer program or the computer instructions, when executed by a processor, implement steps of the method according to any one of claims 1 to 10, any one of claims 11 to 13, claim 14, or any one of claims 15 to 17.