LARGE-MODEL TRAINING METHOD, APPARATUS AND DEVICE, AND STORAGE MEDIUM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to Chinese Patent Application No. 202410252065.9, filed on March 5, 2024, with the State Intellectual Property Office of the People's Republic of China, entitled "Large-Model Training Method, Apparatus and Device, and Storage Medium", the entire content of which is incorporated herein by reference.

FIELD OF THE DISCLOSURE

[0002] The application relates to the field of big data technology and, more particularly, to a large-model training method, apparatus, device and storage medium.

BACKGROUND

[0003] With the further development of big model technology and big data, paying attention to data privacy and security has become a global trend. How to ensure data and model privacy while enabling both parties to collaborate on large-model training and inference is a key focus.

[0004] Currently, a trained large model typically contains billions or tens of billions of parameters, requiring computing power and private data for fine-tuning. Small- and medium-sized enterprises often have localized data that can be used to fine-tune large models. Model and data owners want to collaborate on a large model, but neither party is willing to disclose its content to the other. However, small- and medium-sized enterprises have limited computing power on their local client terminals.

[0005] Therefore, in the relevant technologies, during the training of large models, there are technical problems such as low data security and insufficient utilization of computing power.

SUMMARY

[0006] The present disclosure provides a large-model training method, apparatus, device, and storage medium, thereby solving the technical problems of low data security and insufficient computing power utilization during large-model training.

[0007] In a first aspect, the present disclosure provides a method for training a large model, the large model includes an adaptor header, a model main body and an adaptor tail, the adaptor header includes a first cloud part and a first local part, the adaptor tail includes a second cloud part and a second local part, in a process of training the large model, the first cloud part, the model main body and the second cloud part are deployed on a cloud terminal, and the first local part and the second local part are deployed on a local client terminal,

[0008] and the method includes:

[0009] the local client terminal obtaining training data and performing sharding processing on the training data to obtain first sharded training data and second sharded training data;

[0010] the local client terminal sending the first sharded training data to the cloud terminal;

[0011] the cloud terminal inputting the first sharded training data to the first cloud part for computation, and the local client terminal inputting the second sharded training data to the first local part for computation so as to output a first eigenvector and send the first eigenvector to the cloud terminal;

[0012] the cloud terminal inputting the first eigenvector to the model main body so as to

output a second eigenvector;

[0013] the cloud terminal performing sharding processing on the second eigenvector to obtain a first sharded eigenvector and a second sharded eigenvector, and sending the second sharded eigenvector to the local client terminal;

[0014] the cloud terminal inputting the first sharded eigenvector to the second cloud part for computation, and the local client terminal inputting the second sharded eigenvector to the second local part for computation so as to output a prediction result of the training data; and

[0015] the cloud terminal and the local client terminal iteratively adjusting the large model according to prediction results of a preset amount of training data until the large model converges.

[0016] In the process of training the large model, the present disclosure splits the large model into three parts, the middle model main body M0 and two adaptor parts, the adaptor header and the adaptor tail. The adaptor header includes a first cloud part AH1 and a first local part AH2, and the adaptor tail includes a second cloud part AT1 and a second local part AT2. Each adaptor part is divided into two parts, half of which is located in the cloud terminal and half is located in the local client terminal. The training data for the large model is the industry privatized data from the client terminal. After the training data is sharded and entered, it enters the cloud terminal and the local client terminal respectively for the first joint training. After the first training, a first eigenvector in the form of an eigenvector is obtained, which is then input into the cloud terminal for plaintext training of all the training data. The output of the cloud terminal is sharded again and processed for a second joint training by the cloud terminal and the local client terminal respectively to obtain the prediction result of the training data. According to the prediction result of the training data, a trained large model can be obtained. Based on the

above training method, during the collaborative reasoning operation of the large model and the small model, neither the model nor the data leaves the local machine. The cloud terminal can merely obtain a portion of the data after sharding, and it is difficult to restore the eigenvector of the original input, which protects privacy and improves data security. In addition, merely the adaptor header and adaptor tail are distributed to the cloud terminal and the client terminal, and the encrypted client data is used for the adaptor part of model training. The main structure of the model is still located in the cloud terminal, and the cloud computing power is used for training and reasoning, which ensures the computing power performance. At the same time, the adaptor fine-tuning mechanism is effectively utilized. The number of adaptor parameters merely accounts for a few percent or a few thousandths of the entire original large model, which can achieve almost the same effect as the fine-tuning of all parameters of the full model, realizing the effective training of large models under the premise of securely and fully utilizing the computing power.

[0017] Optionally, the cloud terminal inputting the first sharded training data into the first cloud part for computation, and the local client terminal inputting the second sharded training data into the first local part for computation to output the first eigenvector and send the first eigenvector to the cloud terminal includes:

[0018] the cloud terminal inputting the first sharded training data to the first cloud part for computation, the local client terminal inputting the second sharded training data to the first local part for computation, and in a computation process by the first cloud part and the first local part, carrying out an encryption computation method of multi-party secure computing, and outputting the first eigenvector and sending the first eigenvector to the cloud terminal after the computation process of the first cloud part and the first local part.

[0019] Here, the present disclosure can share computing parameters based on multi-party secure computing, thereby realizing joint training by the first cloud part and the first local part based on a secret sharing protocol, and converting the original data into a difficult-to-restore eigenvector through multi-party secure computing, further improving the security and feasibility of large-model training.

[0020] Optionally, the cloud terminal inputting the first sharded eigenvector to the second cloud part for computation, and the local client terminal inputting the second sharded eigenvector to the second local part for computation to output the prediction result of the training data includes:

[0021] the cloud terminal inputting the first sharded eigenvector to the second cloud part for computation, the local client terminal inputting the second sharded eigenvector to the second local part for computation, and in a computation process by the second cloud part and the second local part, carrying out an encryption computation method of multi-party secure computing, and outputting the prediction result of the training data after the computation process of the second cloud part and the second local part.

[0022] Here, the computation of the adaptor tail in the present disclosure is also based on multi-party secure computing to share operation parameters, thereby realizing joint training by the second cloud part and the second local part based on the secret sharing protocol, further improving the security and feasibility of large-model training.

[0023] Optionally, the multi-party secure computing includes an ABY hybrid protocol or a secret sharing protocol.

[0024] Optionally, the cloud terminal and the local client terminal iteratively adjusting the large model according to the prediction results of the preset amount of training data until the

large model converges includes:

[0025] the cloud terminal obtaining loss parameters according to the prediction result and preset labels corresponding to the training data; the cloud terminal adjusting model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters; the local client terminal adjusting model parameters of the first local part and the second local part of the large model according to the loss parameters; and if the local client terminal determines that a prediction error of the large model is larger than a preset threshold, re-executing a step of "the local client terminal obtaining the training data" so as to continuously train the large model, and if the prediction error of the large model is smaller than or equal to the preset threshold, stopping iterative training.

[0026] Optionally, the cloud terminal obtaining the loss parameters according to the prediction result and the preset labels corresponding to the training data includes:

[0027] the cloud terminal inputting the prediction result and the preset labels into a loss function so as to output the loss parameters.

[0028] Optionally, a calculation formula for the loss function is:

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

[0029] Here, L denotes the loss function, M denotes a quantity of categories, N denotes a quantity of samples, y_{ic} is a sign function, indicating a probability that an i-th sample belongs to a c-th category, p_{ic} denotes a probability predicted by the large model that the i-th sample belongs to the c-th category, and i and N are arbitrary positive integers.

[0030] Here, after obtaining the prediction result of the training data, the present disclosure calculates the loss with the predicted labels, and feeds back the entire model network

for model convergence computation, thereby adjusting the model parameters of the large model to adjust the large model until the requirements are met, thereby realizing the training of the large model.

[0031] Optionally, before the cloud terminal obtaining the loss parameters according to the prediction result and the preset labels corresponding to the training data and adjusting the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters, the method further includes:

[0032] the local client terminal encrypting the preset labels corresponding to the training data according to a preset encryption algorithm to obtain encrypted labels; and the local client terminal sending the encrypted labels to the cloud terminal.

[0033] Here, in order to further ensure data security during the large-model training process, when the local client terminal of the present disclosure sends the preset labels corresponding to the training data to the cloud terminal, the preset labels are first encrypted. The cloud terminal cannot obtain the original preset labels, which further improves the security of large-model training.

[0034] Optionally, the cloud terminal obtaining the loss parameters according to the prediction result and the preset labels corresponding to the training data and adjusting the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters includes:

[0035] the cloud terminal encrypting the prediction result according to the preset encryption algorithm to obtain an encrypted prediction result; and the cloud terminal obtaining the loss parameters according to the encrypted prediction result and the encrypted labels and adjusting the model parameters of the first cloud part and the second cloud part of the large

model according to the loss parameters.

[0036] Optionally, the preset encryption algorithm is a homomorphic encryption algorithm.

[0037] Here, the cloud terminal encrypts the prediction result of the training data according to the preset encryption algorithm of the local client terminal for the preset labels, so that the encrypted prediction result can be processed with the encrypted labels to obtain the loss parameters, and then the model parameters can be adjusted under the premise of security and reliability.

[0038] Optionally, performing sharding processing on the training data to obtain the first sharded training data and the second sharded training data includes:

[0039] carrying out random sharding processing on the training data to obtain the first sharded training data and the second sharded training data.

[0040] Here, when sharding the training data, the present disclosure can adopt a random sharding method. Through this method, the cloud terminal cannot restore the original training data, thereby fully protecting the data security of the local client terminal.

[0041] Optionally, performing sharding processing on the training data to obtain the first sharded training data and the second sharded training data includes:

[0042] performing sharding processing on the training data according to a first preset secret sharding algorithm to obtain the first sharded training data and the second sharded training data.

[0043] Here, when sharding the training data, the present disclosure can adopt a first preset secret sharding algorithm. Through this method, the cloud terminal cannot restore the original training data, thereby fully protecting the data security of the local client terminal.

[0044] Optionally, the cloud terminal performing sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector includes:

[0045] the cloud terminal performing random sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector.

[0046] Optionally, the cloud terminal performing sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector includes:

[0047] the cloud terminal performing sharding processing on the second eigenvector according to a second preset secret sharding algorithm to obtain the first sharded eigenvector and the second sharded eigenvector.

[0048] Here, the cloud terminal of the present disclosure can shard the second eigenvector through random or preset algorithms, and further split the data to achieve secure and reliable joint training by the cloud terminal and local client terminal, further improving the data security and data privacy of large-model training.

[0049] Optionally, the adaptor header and adaptor tail of the trained large model are deployed on the local client terminal, and the model main body is deployed on the cloud. terminal

[0050] Here, the adaptor header and adaptor tail of the trained large model in the present disclosure are deployed locally, and the model main body is deployed in the cloud. The data inference and prediction stage also requires cooperation between both parties to complete. No one party can complete the inference independently. The data rights and interests of the local client terminal are guaranteed, and the model and computing power rights and interests of the cloud terminal are guaranteed, which improves the security of large-model predictions and the utilization of computing resources.

[0051] Optionally, the above method further includes:

[0052] the local client terminal obtaining a piece of to-be-predicted data; the local client terminal inputting the to-be-predicted data to the adaptor header to output a first prediction vector; the local client terminal sending the first prediction vector to the cloud terminal; the cloud terminal inputting the first prediction vector to the model main body so as to output a second prediction vector; the cloud terminal sending the second prediction vector to the local client terminal; and the local client terminal inputting the second prediction vector to the adaptor tail so as to output a prediction result of the to-be-predicted data

[0053] Here, the present disclosure can perform inference and prediction based on the trained large model. For the to-be-predicted data on the local client terminal, the adaptor header of the local client terminal is first used to convert the to-be-predicted data into a first prediction vector encrypted relative to the cloud terminal, and then the computing power of the cloud terminal is used to fully infer and predict the to-be-predicted data, and then the prediction result of the to-be-predicted data is output through the adaptor tail of the local client terminal, thereby realizing secure and efficient data inference and prediction.

[0054] Optionally, the data format of the first prediction vector is an encrypted format.

[0055] Here, the encrypted format of the first prediction vector further ensures the security of the local client terminal data.

[0056] In a second aspect, the present disclosure provides a large-model training apparatus, the large model including an adaptor header, a model main body, and an adaptor tail; where the adaptor header includes a first cloud part and a first local part, and the adaptor tail includes a second cloud part and a second local part; during a training process of the large model, the first cloud part, the model main body, and the second cloud part are deployed in a cloud

terminal, and the first local part and the second local part are deployed on a local client terminal;

[0057] and the apparatus includes:

[0058] a first processing module, configured to obtain training data from a local client terminal and perform sharding processing on the training data to obtain first sharded training data and second sharded training data;

[0059] a first transmission module, configured for the local client terminal to send the first sharded training data to the cloud terminal;

[0060] a second processing module, configured to input the first sharded training data into the first cloud part for computation on the cloud terminal, and to input the second sharded training data into the first local part for computation on the local client terminal, so as to output a first eigenvector and send it to the cloud terminal;

[0061] a first input module, configured to input the first eigenvector into the model main body on the cloud terminal to output a second eigenvector;

[0062] a third processing module, configured to perform sharding processing on the second eigenvector in the cloud terminal to obtain a first sharded eigenvector and a second sharded eigenvector, and send the second sharded eigenvector to the client terminal;

[0063] a fourth processing module, configured to input the first sharded eigenvector to the second cloud part for computation on the cloud terminal, and to input the second sharded eigenvector to the second local part for computation on the local client terminal, so as to output a prediction result of the training data; and

[0064] a fifth processing module, configured for the cloud terminal and the local client terminal to iteratively adjust the large model according to the prediction results of a preset amount of training data until the model converges.

[0065] Optionally, the second processing module is specifically configured such that:

[0066] the cloud terminal inputs the first sharded training data into the first cloud part for computation, and the local client terminal inputs the second sharded training data into the first local part for computation, and during the computation process by the first cloud part and the first local part, an encrypted computation method of multi-party secure computing is used to output the first eigenvector after the computation process of the first cloud part and the first local part and send the first eigenvector to the cloud terminal.

[0067] Optionally, the fourth processing module is specifically configured such that:

[0068] the cloud terminal inputs the first sharded eigenvector into the second cloud part for computation, and the local client terminal inputs the second sharded eigenvector into the second local part for computation, and during the computation process by the second cloud part and the second local part, an encrypted computation method of multi-party secure computing is used to output the prediction result of the training data through the computation of the second cloud part and the second local part.

[0069] Optionally, the multi-party secure computing includes an ABY hybrid protocol or a secret sharing protocol.

[0070] Optionally, the fifth processing module is specifically configured such that:

[0071] the cloud terminal obtains loss parameters based on the prediction result and the preset labels corresponding to the training data; the cloud terminal adjusts the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters; the local client terminal adjusts the model parameters of the first local part and the second local part of the large model according to the loss parameters; if the local client terminal determines that the prediction error of the large model is greater than a preset threshold, the step

of “the local client terminal obtaining training data” is re-executed to continue training the large model; if it is determined that the prediction error of the large model is less than or equal to the preset threshold, the iterative training is stopped.

[0072] Optionally, the fifth processing module is further specifically configured to: input the prediction result and the preset labels into a loss function on the cloud terminal to output the loss parameters.

[0073] Optionally, a calculation formula for the loss function is:

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

[0074] Here, L denotes the loss function, M denotes a quantity of categories, N denotes a quantity of samples, y_{ic} is a sign function, indicating a probability that an i-th sample belongs to a c-th category, p_{ic} denotes a probability predicted by the large model that the i-th sample belongs to the c-th category, and i and N are arbitrary positive integers.

[0075] Optionally, before the fifth processing module is used in the cloud terminal to obtain the loss parameters based on the prediction result and the preset labels corresponding to the training data and adjust the model parameters of the first cloud part and the second cloud part of the large model based on the loss parameters, the apparatus further includes an encryption processing module configured such that:

[0076] the local client terminal encrypts the preset labels corresponding to the training data according to a preset encryption algorithm to obtain encrypted labels; and the local client terminal sends the encrypted labels to the cloud terminal.

[0077] Optionally, the fifth processing module is specifically configured such that:

[0078] the cloud terminal encrypts the prediction result according to the preset

encryption algorithm to obtain an encrypted preset result; the cloud terminal obtains the loss parameters based on the encrypted prediction result and the encrypted labels, and adjusts the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters.

[0079] Optionally, the preset encryption algorithm is a homomorphic encryption algorithm.

[0080] Optionally, the first processing module is specifically configured such that:

[0081] the training data is randomly sharded to obtain the first sharded training data and the second sharded training data.

[0082] Optionally, the first processing module is specifically configured such that:

[0083] the training data is sharded according to a first preset secret sharding algorithm to obtain the first sharded training data and the second sharded training data.

[0084] Optionally, the third processing module is specifically configured such that:

[0085] the cloud terminal performs random sharding processing on the second eigenvector to obtain a first sharded eigenvector and a second sharded eigenvector.

[0086] Optionally, the third processing module is specifically configured such that:

[0087] the cloud terminal performs sharding processing on the second eigenvector according to a second preset secret sharding algorithm to obtain a first sharded eigenvector and a second sharded eigenvector.

[0088] Optionally, the adaptor header and adaptor tail of the trained large model are deployed on the local client terminal, and the model main body is deployed on the cloud terminal.

[0089] Optionally, the above apparatus further includes an inference prediction module,

configured such that:

[0090] the local client terminal obtains a piece of to-be-predicted data; the local client terminal inputs the to-be-predicted data into the adaptor header to output a first prediction vector; the local client terminal sends the first prediction vector to the cloud terminal; the cloud terminal inputs the first prediction vector into the model main body to output a second prediction vector; the cloud terminal sends the second prediction vector to the local client terminal; the local client terminal inputs the second prediction vector into the adaptor tail to output a prediction result of the to-be-predicted data.

[0091] Optionally, the data format of the first prediction vector is an encrypted format.

[0092] In a third aspect, the present disclosure provides a large-model training device, including: at least one processor and a memory.

[0093] The memory stores computer-executable instructions.

[0094] The at least one processor executes the computer-executable instructions stored in the memory, so that the at least one processor performs the large-model training method described in the first aspect and various possible designs of the first aspect.

[0095] In a fourth aspect, the present disclosure provides a computer-readable storage medium, which stores computer-executable instructions. When a processor executes the computer-executable instructions, it implements the large-model training method described in the first aspect and various possible designs of the first aspect.

[0096] In a fifth aspect, the present disclosure provides a computer program product, including a computer program that, when executed by a processor, implements the large-model training method described in the first aspect and various possible designs of the first aspect.

[0097] The present disclosure provides a large-model training method, apparatus, device

and storage medium, where the method uses industry privatized data from the client terminal as training data for the large model. After the data is sharded and entered, it is entered into the cloud terminal and the local client terminal for the first joint training. After the first training, the first eigenvector in the form of an eigenvector is obtained, which is then input into the cloud terminal for plaintext training of all the training data. The output of the cloud terminal is then sharded and processed for a second joint training by the cloud terminal and the local client terminal to obtain the prediction result of the training data. Based on the prediction result of the training data, a trained large model can be obtained. Based on the above training method, during the collaborative reasoning operation of the large model and the small model, neither the model nor the data leaves the local machine. The cloud terminal can merely obtain a portion of the data after sharding, and it is difficult to restore the eigenvector of the original input, which protects privacy and improves data security. In addition, merely the adaptor header and adaptor tail are distributed to the cloud terminal and the client terminal, and the encrypted client data is used for the adaptor part of model training. The main structure of the model is still located in the cloud terminal, and the cloud computing power is used for training and reasoning, which ensures the computing power performance. At the same time, the adaptor fine-tuning mechanism is effectively utilized. The number of adaptor parameters merely accounts for a few percent or a few thousandths of the entire original large model, which can achieve almost the same effect as the fine-tuning of all parameters of the full model, realizing the effective training of large models under the premise of secure and full utilization of computing power.

BRIEF DESCRIPTION OF THE DRAWINGS

[0098] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments consistent with the present disclosure and, together

with the description, serve to explain the principles of the present disclosure.

[0099] FIG. 1 is a schematic structural diagram of a joint model of a cloud terminal and local client terminal according to an embodiment of the present disclosure;

[00100] FIG. 2 is a schematic structural diagram of a large model based on an adaptor method according to an embodiment of the present disclosure;

[00101] FIG. 3 is a schematic structural diagram of a model during a training process of a large model according to an embodiment of the present disclosure;

[00102] FIG. 4 is a flowchart of a large-model training method according to an embodiment of the present disclosure;

[00103] FIG. 5 is a schematic diagram of the basic operation logic of an ABY protocol according to an embodiment of the present disclosure;

[00104] FIG. 6 is a schematic diagram of a parameter training process according to an embodiment of the present disclosure;

[00105] FIG. 7 is a schematic structural diagram of a trained large model according to an embodiment of the present disclosure;

[00106] FIG. 8 is a flowchart of another large-model training method according to an embodiment of the present disclosure; and

[00107] FIG. 9 is a schematic structural diagram of a large-model training device according to an embodiment of the present disclosure.

[00108] The above drawings illustrate specific embodiments of the present disclosure, which will be described in more detail below. These drawings and the textual descriptions are not intended to limit the scope of the present disclosure by any means, but rather to illustrate the concepts of the present disclosure to those skilled in the art by reference to specific

embodiments.

DETAILED DESCRIPTION

[00109] Exemplary embodiments will be described in detail hereinafter, with examples illustrated in the accompanying drawings. In the following descriptions, when referring to the drawings, identical numerals in different figures represent identical or similar elements, unless otherwise indicated. The implementations described in the following exemplary embodiments are not intended to represent all implementations consistent with the present disclosure. Rather, they are merely examples of apparatus and methods consistent with certain aspects of the present disclosure, as detailed in the appended claims.

[00110] It should be noted that the user information (including but not limited to user device information, user personal information, etc.) and data (including but not limited to data used for analysis, stored data, displayed data, etc.) involved in this disclosure are all information and data authorized by the users or fully authorized by all parties, and the collection, use and processing of relevant data should comply with relevant laws, regulations and standards, and provide corresponding operational access for users to choose to authorize or refuse.

[00111] A general-purpose, pre-trained large model typically contains billions or tens of billions of parameters, requiring computing power and private data for fine-tuning. Small- and medium-sized enterprises often have localized data that can be used to fine-tune large models, resulting in customized models. However, the current situation or difficulty lies in the fact that model and data owners want to collaborate on a large model, but are reluctant to disclose their respective content (model and data) to each other.

[00112] Currently, there are three approaches to training large models: Approach 1 proposes a privacy-preserving computing method and system based on distributed collaboration.

This technical solution primarily includes a software development kit (SDK), a controller, and multiple privacy-preserving computing appliances. The controller is deployed in the edge cloud, and the multiple privacy-preserving computing appliances are deployed in a private cloud. In the collaborative privacy-preserving computing process, the SDK first requests access to a dataset from n privacy-preserving computing appliances through the controller. Data element holders can register their datasets with the controller through the privacy-preserving computing appliances. This information includes, but is not limited to, data characteristics, data sources, data quantity, data usage rights pricing, and data privacy rules. After the SDK receives data from multiple privacy-preserving computing appliances, developers can develop privacy-preserving computing applications within the SDK. Upon completion, these applications are sent to the controller, which decomposes the privacy-preserving computing applications and sends the resulting computational tasks to each of the n privacy-preserving computing appliances. Each privacy-preserving computing appliance then computes the computational task based on its stored dataset, and the controller transmits the results to the SDK. Approach 2 proposes a federated learning method and system, a first privacy-preserving computing platform, and a second privacy-preserving computing platform. This technical solution enables collaborative optimization of a federated learning model by exchanging model-related information through encryption, without sharing protected private data or transmitting proprietary data externally. The two privacy-preserving computing platforms adhere to different protocols. Specifically, a first interactive operation is performed with a first data party based on a first parameter ciphertext held by the first data party, resulting in first to-be-aggregated data. The first parameter ciphertext is obtained by the first data party training locally based on its own dataset and encrypted according to a first encryption protocol. The first to-be-aggregated data conforms to a second

encryption protocol supported by the second privacy-preserving computing platform. The first to-be-aggregated data is forwarded to the second privacy-preserving computing platform via a routing node, so that the second privacy-preserving computing platform performs a first aggregation computation using the first to-be-aggregated data and second to-be-aggregated data to obtain aggregation parameters. Here, the second to-be-aggregated data is calculated based on the second parameter ciphertext of each second data party on the second privacy-preserving computing platform, and the second to-be-aggregated data conforms to the second encryption protocol. The aggregation parameters are returned by the second privacy-preserving computing platform and forwarded by the routing node, and the next round of iterative computation is performed using the aggregation parameters. Apporch 3 proposes a skyline query method and system for privacy protection of sensitive data. This technical solution uploads sensitive data information to two non-colluding first and second servers through additive secret sharing. The two servers generate a skyline graph through joint privacy computation, and each has a corresponding share graph. At the user terminal, user information is uploaded through additive secret sharing. Each server jointly retrieves the similar information share that the user expects to query on its own share graph through a private information retrieval method, and sends the similar information share to the user terminal. The user terminal recovers the complete query information to assist in decision-making.

[00113] However, all of the above approaches have drawbacks. The technical solution proposed in Approach 1 suffers from certain shortcomings, primarily due to factors such as data characteristics, data sources, and data volume being transferred out of the local private cloud during the collaborative privacy computing task, posing risks to data security. Furthermore, regarding computing power, the final privacy computing task is sent back to n privacy computing

machines via the SDK controller for local execution. For the decentralized participants in privacy computing in federated scenarios, insufficient computing power is often a thorny issue. How to effectively leverage the abundant computing power of major cloud providers to efficiently collaborate on privacy computing tasks remains a research topic. Approach 2 primarily focuses solely on protocol conversion. While the private data of both parties can be kept locally, it fails to consider the impact on computational efficiency and the significant computational cost of transferring full parameters. Approach 3 primarily requires the decision service provider to upload its full historical data to two servers using additive secret sharing. This results in data being transferred out of the local, making the "data available but not visible" mode impossible. Furthermore, uploading the full historical data to two servers, along with the continuous generation of massive amounts of user data, places significant strain on the communication link. Two servers perform grid-based information retrieval. After retrieving similar information that the user wants to query, their respective shares of similar information are sent to the user terminal. The user terminal restores complete similar information to facilitate the user's query and provide assistance to decision-making. Although this approach achieves the isolation of query users and data and has a certain degree of privacy protection, it is essentially a traditional database query and is not close to the specific vertical field's actual application demands for large models (large language models) to process massive data and realize model training and inference.

[00114] From the above, it can be seen that the relevant technologies have technical problems such as low data security and insufficient utilization of computing power during the training of large models.

[00115] In order to solve the above problems, embodiments of the present disclosure

provide a large-model training method, apparatus, device and medium. In this method, the cloud terminal and the local client terminal collaborate to jointly build a customized large model to achieve privacy protection and rights protection. The privacy includes the cloud model and the client's private data. During the cooperation process, the privacy is not exposed to the other party, and the formed model can still reflect the respective rights and interests during use.

[00116] Optionally, the embodiments of the present disclosure can achieve the following functions.

[00117] (1) Protect data and model privacy through secret sharing: Various secret sharing protocols are used in the collaborative reasoning and computing process between large and small models, so that neither the model nor the data leaves the local. Heterogeneous interoperable operators can also be used. Secure operators emphasize basic computing tools based on cryptographic construction and implemented using secure computing protocols, such as multi-party matrix multiplication, Euclidean distance, and size comparison computations.

[00118] (2) Rationally utilize computing power to ensure overall performance: Merely the adaptor header and the adaptor tail are distributed to the cloud terminal and the local client terminal for adaptor model training. The majority of the model structure remains in the cloud terminal, using cloud computing power for training and inference to ensure performance.

[00119] (3) Effectively utilize the adaptor fine-tuning mechanism: The adaptor is generally a partially effective layer in the original large model, such as a low-rank matrix of parameters, or a parameter layer with a very small proportion embedded in each layer of the large model. The number of adaptor parameters accounts for merely a few percent or a few thousandths of the entire original large model, which can achieve almost the same effect as the fine-tuning of all parameters of the entire model. This feature is used to distribute some

parameters of a very small part of the model (i.e., the adaptor) to the client terminal, which jointly completes secure training/inference based on privacy protection/secret sharing protocols with the cloud terminal.

[00120] Optionally, FIG. 1 is a schematic structural diagram of a joint model of the cloud terminal and local client terminal provided in an embodiment of the present disclosure. As shown in FIG. 1, the local client terminal here is also called a client terminal. The embodiments of the present disclosure can obtain a trained large model by jointly training the original large model through the cloud terminal and the local client terminal.

[00121] Optionally, FIG. 2 is a schematic structural diagram of a large model based on the adaptor method provided in an embodiment of the present disclosure. As shown in FIG. 2, the model structure: the model includes three parts, the middle model M0 and two adaptor parts at the head and tail, including an adaptor header (AH) and an adaptor tail (AT). The adaptor is a small effective layer of the head and tail of the original large model, usually accounting for merely a few percent or even a few thousandths of the parameters, and is specifically used to fine-tune the original model. Fine-tuning method: The training data is fed to the large model for fine-tuning training. During the training process, merely the parameters of the adaptor part are updated, and the parameters of the M0 part remain unchanged. Method features: This fine-tuning method can merely fine-tune the parameters of the original large model by a few percent or even a few thousandths, and can achieve a model inference accuracy that is almost the same as fine-tuning the entire original model.

[00122] As shown in FIG. 2, during the model pre-training, the large model M = AH + M0 + AT. After fine-tuning, that is, training is complete, the trained large model M^* = AH^* + M0 + AT^*. Merely the parameters of the adaptor are updated, while the parameters of the M0

component remain unchanged. It should be understood that the "+" here does not represent a mathematical computation, but rather represents the combination of the model M0 and the two adaptor parts at the head and tail.

[00123] It is understood that the structure illustrated in the embodiments of the present disclosure do not constitute a specific limitation on the training system architecture of the large model. In other feasible implementations of the present disclosure, the above architecture may include more or fewer components than those shown in the figure, or combine certain components, or split certain components, or arrange the components differently. The specific configuration can be determined according to the actual application scenario and is not limited herein. The components shown in FIGS. 1 and 2 can be implemented in hardware, software, or a combination of software and hardware.

[00124] In a specific implementation process, the cloud terminal and local client terminal include processors or processing devices.

[00125] It should be understood that the above-described processing device can be implemented by a processor reading instructions in a memory and executing the instructions, or can be implemented by a chip circuit.

[00126] In addition, the network architecture and service scenarios described in the embodiments of the present disclosure are intended to more clearly illustrate the technical solutions of the embodiments of the present disclosure, and do not constitute a limitation on the technical solutions provided by the embodiments of the present disclosure. A person skilled in the art will appreciate that with the evolution of network architecture and the emergence of new service scenarios, the technical solutions provided in the embodiments of the present disclosure are also applicable to similar technical problems.

[00127] The technical solutions of the present disclosure are described in detail hereinafter with reference to specific embodiments.

[00128] Optionally, FIG. 3 is a schematic structural diagram of a model during the training process of a large model provided in an embodiment of the present disclosure. As shown in FIG. 3, the large model in the training process includes an adaptor header (the first cloud part and the first local part constitute the adaptor header), a model main body M0 and an adaptor tail (the second cloud part and the second local part constitute the adaptor tail). The adaptor header includes the first cloud part AH1 and the first local part AH2, and the adaptor tail includes the second cloud part AT1 and the second local part AT2. In the process of training the large model, the first cloud part, the model main body and the second cloud part are deployed on the cloud terminal, and the first local part and the second local part are deployed on the local client terminal.

[00129] During the training process, the data is split into two parts D1 and D2.

[00130] Here, the adaptor is generally composed of some effective layers from the original large model, such as a low-rank matrix of parameters, or a parameter layer with a very small percentage additionally embedded in each layer of the large model. The adaptor includes two parts: the adaptor header and the adaptor tail, denoted as AH and AT, respectively. They are deployed in a distributed manner, with part of each located on the cloud terminal and the other part on the client terminal. The remaining parameters of the large model remain on the cloud terminal.

[00131] As shown in FIG. 3, the cloud part and the local part can achieve joint training through multi-party computing (MPC) operator operations.

[00132] As shown in FIG. 3, the collaborative training process is as follows.

[00133] Step 1: The original training data is the industry privatized data from the client terminal, which is sharded and entered into the next step of the training process based on the secret sharing computation.

[00134] Through step 1, the client's input data is protected.

[00135] Step 2: The adaptor header is divided into two halves, one half located in the cloud terminal and the other half located in the client terminal. The two parties conduct the first joint training based on the secret sharing protocol.

[00136] Through step 2, some parameters of the adaptor are held in a distributed manner.

[00137] Step 3: The trained output of the adaptor header is in plaintext.

[00138] At this point, the output data of the middle layers is already an eigenvector, and it is difficult to restore the original input.

[00139] Step 4: Perform plaintext training on all training data on the cloud terminal.

[00140] The M0 computing part on the cloud terminal can prevent the large model from being leaked while making full use of the computing power on the cloud terminal.

[00141] Step 5: The output of M0 is again secret-shared and split into two halves. The adaptor tail is also split into two halves, one half located in the cloud terminal and the other half located in the client terminal. Both parties conduct a second joint training based on the secret sharing protocol.

[00142] Some parameters of the adaptor continue to be held in a distributed manner.

[00143] Optionally, FIG. 4 is a flowchart of a large-model training method provided in an embodiment of the present disclosure, which is used to train the large model in FIG. 3. The execution entity of the embodiments of the present disclosure can be the joint model of the cloud terminal and local client terminal in FIG. 1, and the specific execution entity can be determined

according to the actual application scenario. As shown in FIG. 4, the method includes the following steps.

[00144] S401: The local client terminal obtains training data and performs sharding processing on the training data to obtain first sharded training data and second sharded training data.

[00145] Optionally, any training data may be obtained, and sharding processing may be performed on the training data to obtain the first sharded training data and second sharded training data.

[00146] Optionally, performing sharding processing on the training data to obtain first sharded training data and second sharded training data includes:

[00147] randomly sharding the training data to obtain the first sharded training data and the second sharded training data.

[00148] Here, the embodiments of the present disclosure can adopt a random sharding method when sharding the training data. Through this method, the cloud terminal cannot restore the original training data, thereby fully protecting the data security of the local client terminal.

[00149] Optionally, performing sharding processing on the training data to obtain the first sharded training data and the second sharded training data includes:

[00150] sharding the training data according to a first preset secret sharding algorithm to obtain the first sharded training data and the second sharded training data.

[00151] It is understood that the first preset secret sharding algorithm can be determined according to actual conditions, and the embodiments of the present disclosure do not impose any specific restrictions on this.

[00152] Here, when performing sharding processing on the training data, the embodiments

of the present disclosure can adopt a first preset secret sharding algorithm. Through this method, the cloud terminal cannot restore the original training data, thereby fully protecting the data security of the local client terminal.

[00153] S402: The local client terminal sends the first sharded training data to the cloud terminal.

[00154] S403: The cloud terminal inputs the first sharded training data into the first cloud part for computation, and the local client terminal inputs the second sharded training data into the first local part for computation, so as to output a first eigenvector and send it to the cloud terminal.

[00155] Optionally, when the cloud terminal inputs the first sharded training data into the first cloud part for computation, and the local client terminal inputs the second sharded training data into the first local part for computation, so as to output the first eigenvector and send it to the cloud terminal, it includes that:

[00156] the cloud inputs the first sharded training data into the first cloud part for computation, and the local client terminal inputs the second sharded training data into the first local part for computation, and during the computation process by the first cloud part and the first local part, an encrypted computation method of multi-party secure computing is used to output the first eigenvector through the computation of the first cloud part and the first local part and send it to the cloud terminal.

[00157] Here, the embodiments of the present disclosure can share computing parameters based on multi-party secure computing, thereby realizing joint training by the first cloud part and the first local part based on a secret sharing protocol, and converting the original data into a difficult-to-restore eigenvector through multi-party secure computing, further improving the

security and feasibility of large-model training.

[00158] Optionally, the multi-party secure computing includes an ABY hybrid protocol or a secret sharing (SS) protocol. The ABY protocol is a two-party dedicated secure computation framework that is a hybrid of three protocols: A stands for arithmetic sharing, B stands for Boolean sharing, and Y stands for Yao's garbled circuits.

[00159] Here, the client's training data is secretly sharded and distributed to the cloud terminal and client terminal. At the same time, the network parameters of the AH part are trained based on secret sharing protocols (such as ABY, SS, etc.).

[00160] Taking the ABY protocol as an example, the training process is explained in detail below.

[00161] The basic operation logic of the ABY protocol is shown in FIG. 5. For example, the client has a fine-tuned dataset for a certain industry:

[00162] "If the price level (i.e., CPI) rises, then the ability of the same amount of money to purchase the same goods and services will decrease, that is, the purchasing power of money will decrease. Conversely, if the price level falls, then the ability of the same amount of money to purchase the same goods and services will increase, that is, the purchasing power of money will increase."

[00163] At this point, the result of the dataset token is [If, the price, level, (i.e., C, P, I) rises, then, the ability of, the same, amount, of, money, to purchase, the goods, and services, will, decrease, conversely, it, increases], N=27 is a list of all words. Next, the word list is encoded and vector embedded. Taking the embedding of some tokens into a 768-dimensional vector as an example, FIG. 6 is a schematic diagram of a parameter training process provided by an embodiment of the present disclosure. As shown in FIG. 6, in this example, the input data we

obtain is a 27*768 matrix, recorded as the input data matrix D. Let AH and AT be m*n dimensional matrices (m and n are any positive integers). In the sharding, we put m/2*n dimensional matrices on the cloud terminal and the client terminal respectively. Next, the matrix D is cut according to the ABY protocol, and random triples are generated for both parties. The ABY secret sharing protocol blinds the computation data on both sides through triples, and the two parties cannot reversely obtain the other party's data during the computation process.

[00164] S404: The cloud terminal inputs the first eigenvector into the model main body to output a second eigenvector.

[00165] Here, the first eigenvector is the plaintext data in a secret-shared form.

[00166] After obtaining the decrypted plaintext data in a secret-shared form, it is ready to enter the training for some other parameters of the large model on the cloud terminal. After the AH part of the operation, we get the sharded multiplication form D1*D2 of the original input data. All the secrets are placed on the cloud terminal and enter the fixed part M0 for training.

[00167] The data is trained on other parameters of the large model on the cloud terminal, making full use of the cloud computing power to obtain the plaintext output of the secret sharing data, which is ready to be sharded again and enter AT to participate in another round of secret sharing training.

[00168] S405: The cloud terminal performs sharding processing on the second eigenvector to obtain a first sharded eigenvector and a second sharded eigenvector, and sends the second sharded eigenvector to the client terminal.

[00169] Optionally, when the cloud terminal performs sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector, it includes that:

[00170] the cloud terminal performs random sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector.

[00171] Optionally, when the cloud terminal performs sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector, it includes that:

[00172] the cloud terminal performs sharding processing on the second eigenvector according to a second preset secret sharding algorithm to obtain the first sharded eigenvector and the second sharded eigenvector.

[00173] Here, in the embodiments of the present disclosure, the cloud terminal can shard the second eigenvector through random or preset algorithms, and further split the data to achieve secure and reliable joint training by the cloud terminal and local client terminal, further improving the data security and data privacy of large-model training.

[00174] S406: The cloud terminal inputs the first sharded eigenvector to the second cloud part for computation, and the local client terminal inputs the second sharded eigenvector to the second local part for computation to output a prediction result of the training data.

[00175] Optionally, the data is distributed to the cloud terminal and client terminal, and the network parameters of the AT part are trained based on a secret sharing protocol (such as ABY, SS, etc.), and the computation logic of the AT part is the same as that in step S403.

[00176] Optionally, when the cloud terminal inputs the first sharded eigenvector to the second cloud part for computation, and the local client terminal inputs the second sharded eigenvector to the second local part for computation, to output the prediction result of the training data, it includes that:

[00177] the cloud terminal inputs the first sharded eigenvector to the second cloud part for

computation, and the local client terminal inputs the second sharded eigenvector to the second local part for computation, and during the computation process by the second cloud part and the second local part, the encryption computation method of multi-party secure computing is used to output the prediction result of the training data through the computation of the second cloud part and the second local part.

[00178] Here, the operation of the adaptor tail in the embodiments of the present disclosure is also based on multi-party secure computing to share operation parameters, thereby realizing joint training by the second cloud part and the second local part based on the secret sharing protocol, further improving the security and feasibility of large-model training.

[00179] S407: The cloud terminal and the local client terminal iteratively adjust the large model according to the prediction results of a preset amount of training data until the model converges.

[00180] Optionally, when the cloud terminal and the local client terminal iteratively adjust the large model according to the prediction results of a preset amount of training data until the model converges, it includes that:

[00181] the cloud terminal obtains the loss parameters based on the prediction result and preset labels corresponding to the training data; the cloud terminal adjusts the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters; the local client terminal adjusts the model parameters of the first local part and the second local part of the large model according to the loss parameters; if the local client terminal determines that the prediction error of the large model is greater than a preset threshold, the step of “the local client terminal obtaining training data” is re-executed to continue training the large model. If it is determined that the prediction error of the large model is less than or equal to the

preset threshold, the iterative training is stopped.

[00182] It is understood that the preset threshold here can be determined according to actual conditions, and the embodiments of the present disclosure do not impose specific limitations on this.

[00183] Optionally, the training output is obtained, and the loss is calculated with the labels, which is then fed back to the entire model network for model convergence computation. Specifically, the masks of the training data tokens are predicted and the loss function is calculated. Here, the general multi-class cross-entropy loss function is used for illustration. The output of the AT is the logits value calculated by the classification layer, which is used to calculate the loss with the true label values. Based on the above steps, the model is iteratively trained until the model converges to a certain accuracy.

[00184] Optionally, when the cloud terminal obtains the loss parameters based on the prediction result and preset labels corresponding to the training data, it includes that:

[00185] the cloud terminal inputs the prediction result and preset labels into the loss function to output the loss parameters.

[00186] Optionally, the calculation formula for the loss function is:

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

[00187] Here, L denotes the loss function, M denotes the number of categories, N denotes the number of samples, y_{ic} is a sign function (takes 0 or 1, takes 1 if the true category of sample i is equal to c, otherwise takes 0), indicating the probability that the i-th sample belongs to the c-th category, p_{ic} denotes the probability predicted by the model that the i-th sample belongs to the c-th category, and i and N are arbitrary positive integers.

[00188] Here, in the embodiments of the present disclosure, after obtaining the prediction result of the training data, it calculates the loss with the preset labels and feeds back the entire model network to perform model convergence computation, thereby adjusting the model parameters of the large model to adjust the large model until the requirements are met, thereby realizing the training of the large model.

[00189] Optionally, before obtaining the loss parameters on the cloud terminal according to the prediction result and the preset labels corresponding to the training data, and adjusting the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters, the method further includes that:

[00190] the local client terminal encrypts the preset labels corresponding to the training data according to a preset encryption algorithm to obtain encrypted labels; and the local client terminal sends the encrypted labels to the cloud terminal.

[00191] Here, in order to further ensure data security during the large-model training process, when the local client terminal of the embodiments of the present disclosure sends the preset labels corresponding to the training data to the cloud terminal, the preset labels are first encrypted. The cloud terminal cannot obtain the original preset labels, which further improves the security of large-model training.

[00192] Optionally, when the cloud terminal obtains the loss parameters based on the prediction result and the preset labels corresponding to the training data, and adjusts the model parameters of the first cloud part and the second cloud part of the large model based on the loss parameters, it includes that:

[00193] the cloud terminal encrypts the prediction result according to a preset encryption algorithm to obtain an encrypted preset result; and the cloud terminal obtains the loss parameters

based on the encrypted prediction result and the encrypted labels, and adjusts the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters.

[00194] Optionally, the preset encryption algorithm is a homomorphic encryption algorithm.

[00195] Here, the cloud terminal encrypts the prediction result of the training data according to the preset encryption algorithm of the local client terminal for the preset labels, so that the encrypted prediction result can be processed with the encrypted labels to obtain the loss parameters, and then the model parameters can be adjusted under the premise of security and reliability.

[00196] Optionally, the trained AH and AT in the above steps are merged and placed on the client terminal to obtain the final trained large model. The adaptor header and adaptor tail of the trained large model are deployed on the local client terminal, and the model main body is deployed on the cloud terminal.

[00197] Exemplarily, FIG. 7 is a structural diagram of a trained large model provided in an embodiment of the present disclosure. Through the trained large model in FIG. 7, data inference and prediction can be realized.

[00198] In the process of training the large model, the embodiments of the present disclosure splits the large model into three parts, the middle model main body M0 and two adaptor parts, the adaptor header and the adaptor tail. The adaptor header includes a first cloud part AH1 and a first local part AH2, and the adaptor tail includes a second cloud part AT1 and a second local part AT2. Each adaptor part is divided into two parts, half of which is located in the cloud terminal and the other half is located in the local client terminal. The training data for the

large model is the industry privatized data from the client terminal. After the data is sharded and entered, it enters the cloud terminal and the local client terminal respectively for the first joint training. After the first training, a first eigenvector in the form of an eigenvector is obtained, which is then input into the cloud terminal for plaintext training of all the training data. The output of the cloud terminal is sharded again and processed for the second joint training by the cloud terminal and the local client terminal respectively to obtain the prediction result of the training data. According to the prediction result of the training data, a trained large model can be obtained. Based on the above training method, during the collaborative reasoning operation of the large model and the small model, neither the model nor the data leaves the local machine. The cloud terminal can merely obtain a portion of the data after sharding, and it is difficult to restore the eigenvector of the original input, which protects privacy and improves data security. In addition, merely the adaptor header and adaptor tail are distributed to the cloud terminal and the client terminal, and the encrypted client data is used for the adaptor part of model training. The main structure of the model is still located in the cloud terminal, and the cloud computing power is used for training and reasoning, which ensures the computing power performance. At the same time, the adaptor fine-tuning mechanism is effectively utilized. The number of adaptor parameters merely accounts for a few percent or a few thousandths of the entire original large model, which can achieve almost the same effect as the fine-tuning of all parameters of the full model, thereby realizing the effective training of large models under the premise of secure and full utilization of computing power.

[00199] Optionally, the embodiments of the present disclosure can implement data inference training through a trained large model. Accordingly, FIG. 8 is a flowchart of another large-model training method provided by an embodiment of the present disclosure. As shown in

FIG. 8, the method includes the following.

[00200] S801: The local client terminal obtains training data and performs sharding processing on the training data to obtain first sharded training data and second sharded training data.

[00201] S802: The local client terminal sends the first sharded training data to the cloud terminal.

[00202] S803: The cloud terminal inputs the first sharded training data into the first cloud part for computation, and the local client terminal inputs the second sharded training data into the first local part for computation, so as to output a first eigenvector and send it to the cloud terminal.

[00203] S804: The cloud terminal inputs the first eigenvector into the model main body to output a second eigenvector.

[00204] S805: The cloud terminal performs sharding processing on the second eigenvector to obtain a first sharded eigenvector and a second sharded eigenvector, and sends the second sharded eigenvector to the client terminal.

[00205] S806: The cloud terminal inputs the first sharded eigenvector to the second cloud part for computation, and the local client terminal inputs the second sharded eigenvector to the second local part for computation to output a prediction result of the training data.

[00206] S807: The cloud terminal and the local client terminal iteratively adjust the large model according to the prediction results of a preset amount of training data until the model converges.

[00207] The implementation of steps S801-S807 is the same as that of steps S401-S407, and will not be described in detail here.

[00208] S808: The local client terminal obtains a piece of to-be-predicted data.

[00209] S809: The local client terminal inputs the to-be-predicted data into the header adaptor to output a first prediction vector.

[00210] The data format of the first prediction vector is an encrypted format.

[00211] Here, the first prediction vector is in an encrypted format for the cloud terminal, and the cloud terminal cannot obtain specific data, which improves the security of data inference and prediction.

[00212] First, the inference data passes through the AH part on the client terminal and is output as a scrambled form of the original plaintext inference data, which cannot be reversed, thereby ensuring the privacy of the inference data. Next, this form of inference data is passed to the remaining parameters of the large model on the cloud terminal to participate in inference.

[00213] S810: The local client terminal sends the first prediction vector to the cloud terminal.

[00214] S811: The cloud terminal inputs the first prediction vector to the model main body to output a second prediction vector.

[00215] The data in the previous step is inferred and computed using the remaining parameters of the model to obtain the computation output, which is the second prediction vector.

[00216] S812: The cloud terminal sends the second prediction vector to the local client terminal.

[00217] S813: The local client terminal inputs the second prediction vector to the adaptor tail to output a prediction result of the to-be-predicted data.

[00218] The computation output of the previous step is passed to the client terminal, enters the AT part for inference computation, and obtains the model inference result for output.

[00219] The embodiments of the present disclosure achieve privacy protection. Neither the large model and training data before collaboration nor the incremental model and inference data after collaboration are exposed to the other party. This also protects rights and interests. Both parties collaborate to form a joint large model, and the inference phase also requires cooperation, as neither party can complete inference independently. This ensures the data rights and interests of the client terminal, and the model and computing power rights of the cloud terminal are protected.

[00220] The embodiments of the present disclosure can perform inference and prediction based on a trained large model. For the to-be-predicted data on the local client terminal, the adaptor header of the local client terminal is first used to convert the to-be-predicted data into a first prediction vector encrypted relative to the cloud terminal. Then, the computing power of the cloud terminal is used to fully infer and predict the to-be-predicted data, and then the prediction result of the to-be-predicted data is output through the adaptor tail of the local client terminal, thereby realizing secure and efficient data inference and prediction.

[00221] The embodiments of the present disclosure also provide a large-model training apparatus, and the apparatus of the embodiments of the present disclosure includes: a first processing module, a first transmission module, a second processing module, a first input module, a third processing module, a fourth processing module, and a fifth processing module. The large-model training apparatus here can be the processing apparatus of the cloud terminal or local client terminal, the processor itself, or a chip or integrated circuit that implements the functions of the processor. It should be noted here that the division of the first processing module, the first transmission module, the second processing module, the first input module, the third processing module, the fourth processing module, and the fifth processing module is just a

division of logical functions. Physically, the modules can be integrated or independent.

[00222] The first processing module is configured for the local client terminal to obtain training data and perform sharding processing on the training data to obtain first sharded training data and second sharded training data.

[00223] The first transmission module is configured for the local client terminal to send the first sharded training data to the cloud terminal.

[00224] The second processing module is configured to input the first sharded training data into the first cloud part for computation on the cloud terminal, and to input the second sharded training data into the first local part for computation on the local client terminal, so as to output a first eigenvector and send it to the cloud terminal.

[00225] The first input module is configured for the cloud terminal to input the first eigenvector into the model main body to output a second eigenvector.

[00226] The third processing module is configured to perform sharding processing on the second eigenvector in the cloud terminal to obtain a first sharded eigenvector and a second sharded eigenvector, and send the second sharded eigenvector to the client terminal.

[00227] The fourth processing module is configured to input the first sharded eigenvector to the second cloud part for computation on the cloud terminal, and to input the second sharded eigenvector to the second local part for computation on the local client terminal, so as to output a prediction result of the training data.

[00228] The fifth processing module is configured for the cloud terminal and the local client terminal to iteratively adjust the large model according to the prediction results of a preset amount of training data until the model converges.

[00229] Optionally, the second processing module is specifically configured such that:

[00230] the cloud terminal inputs the first sharded training data into the first cloud part for computation, and the local client terminal inputs the second sharded training data into the first local part for computation, and during the computation process by the first cloud part and the first local part, an encrypted computation method of multi-party secure computing is used to output the first eigenvector through the computation of the first cloud part and the first local part and send it to the cloud terminal.

[00231] Optionally, the fourth processing module is specifically configured such that:

[00232] the cloud terminal inputs the first sharded eigenvector to the second cloud part for computation, and the local client terminal inputs the second sharded eigenvector to the second local part for computation, and during the computation process by the second cloud part and the second local part, the encryption computation method of multi-party secure computing is used to output the prediction result of the training data through the computation of the second cloud part and the second local part.

[00233] Optionally, the multi-party secure computing includes an ABY hybrid protocol or a secret sharing protocol.

[00234] Optionally, the fifth processing module is specifically configured such that:

[00235] the cloud terminal obtains the loss parameters based on the prediction result and the preset labels corresponding to the training data; the cloud terminal adjusts the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters; the local client terminal adjusts the model parameters of the first local part and the second local part of the large model according to the loss parameters; if the local client terminal determines that the prediction error of the large model is greater than a preset threshold, the step of “the local client terminal obtaining training data” is re-executed to continue training

the large model. If it is determined that the prediction error of the large model is less than or equal to the preset threshold, the iterative training is stopped.

[00236] Optionally, the fifth processing module is further specifically configured to: input the prediction result and the preset labels into the loss function on the cloud terminal to output the loss parameters.

[00237] Optionally, the calculation formula for the loss function is:

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

[00238] Here, L denotes the loss function, M denotes the number of categories, N denotes the number of samples, y_{ic} is a sign function, indicating the probability that the i-th sample belongs to the c-th category, p_{ic} denotes the probability predicted by the model that the i-th sample belongs to the c-th category, and i and N are arbitrary positive integers.

[00239] Optionally, before the fifth processing module is used in the cloud terminal to obtain the loss parameters based on the prediction result and the preset labels corresponding to the training data, and to adjust the model parameters of the first cloud part and the second cloud part of the large model based on the loss parameters, the apparatus further includes an encryption processing module configured such that:

[00240] the local client terminal encrypts the preset labels corresponding to the training data according to a preset encryption algorithm to obtain encrypted labels; and the local client terminal sends the encrypted labels to the cloud terminal.

[00241] Optionally, the fifth processing module is specifically configured such that:

[00242] the cloud terminal encrypts the prediction result according to a preset encryption algorithm to obtain an encrypted preset result; the cloud terminal obtains the loss parameters

based on the encrypted prediction result and the encrypted labels, and adjusts the model parameters of the first cloud part and the second cloud part of the large model according to the loss parameters.

[00243] Optionally, the preset encryption algorithm is a homomorphic encryption algorithm.

[00244] Optionally, the first processing module is specifically configured such that:

[00245] the training data is randomly sharded to obtain the first sharded training data and the second sharded training data.

[00246] Optionally, the first processing module is specifically configured such that:

[00247] the training data is sharded according to a first preset secret sharding algorithm to obtain the first sharded training data and the second sharded training data.

[00248] Optionally, the third processing module is specifically configured such that:

[00249] the cloud terminal performs random sharding processing on the second eigenvector to obtain the first sharded eigenvector and the second sharded eigenvector.

[00250] Optionally, the third processing module is specifically configured such that:

[00251] the cloud terminal performs sharding processing on the second eigenvector according to a second preset secret sharding algorithm to obtain the first sharded eigenvector and the second sharded eigenvector.

[00252] Optionally, the adaptor header and adaptor tail of the trained large model are deployed on the local client terminal, and the model main body is deployed on the cloud terminal.

[00253] Optionally, the above apparatus further includes an inference prediction module, configured such that:

[00254] the local client terminal obtains a piece of to-be-predicted data; the local client terminal inputs the to-be-predicted data into the adaptor header to output a first prediction vector; the local client terminal sends the first prediction vector to the cloud terminal; the cloud terminal inputs the first prediction vector into the model main body to output a second prediction vector; the cloud terminal sends the second prediction vector to the local client terminal; and the local client terminal inputs the second prediction vector into the adaptor tail to output the prediction result of the to-be-predicted data.

[00255] Optionally, the data format of the first prediction vector is an encrypted format.

[00256] Referring to FIG. 9, which shows a schematic structural diagram of a large-model training device suitable for implementing the embodiments of the present disclosure. The large-model training device 900 can be a terminal device or a server. The terminal device can include, but is not limited to, mobile terminals such as mobile phones, laptop computers, digital broadcast receivers, personal digital assistants (PDAs), tablet computers (portable Android® devices, PADs), portable multimedia players (PMPs), vehicle-mounted terminals (such as vehicle-mounted navigation terminals), etc., as well as fixed terminals such as digital TVs, desktop computers, etc. The large-model training device shown in FIG. 9 is merely an example and should not impose any limitations on the functions and scope of use of the embodiment of the present disclosure.

[00257] As shown in FIG. 9, the large-model training device 900 may include a processing apparatus (e.g., a central processing unit, a graphics processing unit, etc.) 901, which can perform various appropriate actions and processes according to a program stored in a read-only memory (ROM) 902 or a program loaded from a storage apparatus 908 into a random access memory (RAM) 903. Various programs and data required for the operation of the large-

model training device 900 are also stored in the RAM 903. The processing apparatus 901, the ROM 902, and the RAM 903 are connected to each other via a bus 904. An input/output (I/O) interface 905 is also connected to the bus 904.

[00258] Typically, the following apparatuses can be connected to the I/O interface 905: input apparatus 906 including, for example, a touch screen, touchpad, keyboard, mouse, camera, microphone, accelerometer, gyroscope, etc.; output apparatus 907 including, for example, a liquid crystal display (LCD), speaker, vibrator, etc.; storage apparatus 908 including, for example, a magnetic tape, hard disk, etc.; and communication apparatus 909. The communication apparatus 909 can allow the large-model training device 900 to communicate with other devices wirelessly or by wire to exchange data. Although FIG. 9 shows a large-model training device 900 with various apparatuses, it should be understood that it is not required to implement or have all of the apparatuses shown. More or fewer apparatuses may be implemented or included instead.

[00259] In particular, according to an embodiment of the present disclosure, the processes described above with reference to the flowcharts can be implemented as a computer software program. For example, an embodiment of the present disclosure includes a computer program product, which includes a computer program carried on a computer-readable medium, and the computer program includes program code for executing the methods shown in the flowcharts. In such an embodiment, the computer program can be downloaded and installed from the network through the communication apparatus 909, or installed from the storage apparatus 908, or installed from the ROM 902. When the computer program is executed by the processing apparatus 901, the above-described functions defined in the methods of the embodiments of the present disclosure are performed.

[00260] It should be noted that the computer-readable medium described above in the present disclosure may be a computer-readable signal medium or a computer-readable storage medium, or any combination of the two. A computer-readable storage medium may be, for example, but not limited to, an electrical, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or component, or any combination of the above. More specific examples of computer-readable storage media may include, but are not limited to: an electrical connection with one or more wires, a portable computer disk, a hard disk, a RAM, a ROM, an erasable programmable read-only memory (EPROM or flash memory), an optical fiber, a portable compact disk read-only memory (CD-ROM), an optical storage apparatus, a magnetic storage apparatus, or any suitable combination of the above. In the present disclosure, a computer-readable storage medium may be any tangible medium that contains or stores a program that can be used by or in conjunction with an instruction execution system, apparatus, or component. In the present disclosure, a computer-readable signal medium may include a data signal propagated in baseband or as part of a carrier wave, which carries computer-readable program code. Such a propagated data signal may take a variety of forms, including but not limited to electromagnetic signals, optical signals, or any suitable combination of the above. A computer-readable signal medium may also be any computer-readable medium other than a computer-readable storage medium that can transmit, propagate, or transport a program for use by or in conjunction with an instruction execution system, device, or apparatus. The program code contained on the computer-readable medium may be transmitted using any suitable medium, including but not limited to wires, optical cables, radio frequency (RF), etc., or any suitable combination thereof.

[00261] The computer-readable medium may be included in the large-model training

device; or it may exist independently without being assembled into the large-model training device.

[00262] The computer-readable medium carries one or more programs. When the one or more programs are executed by the large-model training device, the large-model training device executes the methods shown in the above embodiments.

[00263] Computer program code for performing the operations of the present disclosure may be written in one or more programming languages, or a combination thereof, including object-oriented programming languages such as Java, Smalltalk, C++, and conventional procedural programming languages such as “C” or similar programming languages. The program code may be executed entirely on the user’s computer, partially on the user’s computer, as a stand-alone software package, partially on the user’s computer and partially on a remote computer, or entirely on a remote computer or server. In cases involving a remote computer, the remote computer may be connected to the user’s computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or may be connected to an external computer (e.g., through the Internet using an Internet service provider).

[00264] The flowcharts and block diagrams in the accompanying drawings illustrate the possible implementation architecture, functions and operations of the systems, methods and computer program products according to various embodiments of the present disclosure. In this regard, each box in the flowchart or block diagram can represent a module, program segment, or a part of code, and the module, program segment, or a part of code contains one or more executable instructions for realizing the specified logical functions. It should also be noted that in some alternative implementations, the functions marked in the box can also occur in a different order than that marked in the accompanying drawings. For example, two boxes represented in

succession can actually be executed substantially in parallel, and they can sometimes be executed in the opposite order, depending on the functions involved. It should also be noted that each box in the block diagrams and/or flowcharts, and the combination of the boxes in the block diagrams and/or flowcharts, can be implemented with a dedicated hardware-based system that performs the specified functions or operations, or can be implemented with a combination of dedicated hardware and computer instructions.

[00265] The units involved in the embodiments described in this disclosure may be implemented in software or hardware. In some cases, the name of a unit does not limit the unit itself. For example, the first acquisition unit may also be described as a "unit for acquiring at least two Internet Protocol addresses."

[00266] The functions described above in the disclosure may be performed, at least in part, by one or more hardware logic components. For example, and without limitation, exemplary types of hardware logic components that may be used include: field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), application specific standard products (ASSPs), systems on chip (SOCs), complex programmable logic devices (CPLDs), and the like.

[00267] In the context of the present disclosure, a machine-readable medium can be a tangible medium that can contain or store a program for use by or in conjunction with an instruction execution system, apparatus or device. A machine-readable medium can be a machine-readable signal medium or a machine-readable storage medium. A machine-readable medium can include, but is not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus or device, or any suitable combination of the foregoing. A more specific example of a machine-readable storage medium can include an

electrical connection based on one or more lines, a portable computer disk, a hard disk, a RAM, a ROM, an EPROM or flash memory, an optical fiber, a portable CD-ROM, an optical storage device, a magnetic storage device, or any suitable combination of the foregoing.

[00268] The large-model training device of the embodiments of the present disclosure can be used to execute the technical solutions in the above-described method embodiments of the present disclosure. Its implementation principles and technical effects are similar and will not be repeated here.

[00269] An embodiment of the present disclosure also provides a computer-readable storage medium, which stores computer execution instructions. When the computer execution instructions are executed by a processor, they are used to implement any of the above-described large-model training methods.

[00270] An embodiment of the present disclosure also provides a computer program product, including a computer program, which, when executed by a processor, is used to implement any of the above-described large-model training methods.

[00271] In the several embodiments provided in the present disclosure, it should be understood that the disclosed systems, apparatuses and methods can be implemented in other ways. For example, the apparatus embodiments described above are merely schematic. For example, the division of units is merely a logical function division. In actual implementation, there may be other division methods, such as multiple units or components can be combined or integrated into another system, or some features can be ignored or not executed. Another point is that the mutual coupling or direct coupling or communication connection shown or discussed can be an indirect coupling or communication connection through some interface, apparatus or unit, which can be electrical, mechanical or other forms.

[00272] In addition, the functional units in the various embodiments of the present disclosure may be integrated into a single processing unit, or each unit may exist physically separately, or two or more units may be integrated into a single unit. The above-described integrated units may be implemented in the form of hardware or software functional units.

[00273] Those skilled in the art will readily appreciate other embodiments of the present disclosure after considering the specification and practicing the disclosure disclosed herein. This disclosure is intended to cover any variations, uses, or adaptations of the present disclosure that follow the general principles of the present disclosure and include common knowledge or customary techniques in the art not disclosed herein. The descriptions and examples are to be considered as exemplary only, and the true scope and spirit of the present disclosure are indicated by the following claims.

[00274] It should be understood that the present disclosure is not limited to the exact structure described above and shown in the drawings, and that various modifications and changes may be made without departing from the scope thereof. The scope of the present disclosure is limited merely by the appended claims.