

CLUSTER-BASED WORD VECTOR PROCESSING METHOD, DEVICE, AND APPARATUS

TECHNICAL FIELD

The present application relates to the field of computer software technologies, and in particular, to a cluster-based word vector processing method, apparatus, and device.

BACKGROUND

Most of today's natural language processing solutions use neural network-based architectures. An important underlying technology in such an architecture is word vectors. A word vector maps a word to a fixed dimension, and the vector represents semantic information of the word.

In the existing technologies, common algorithms for generating word vectors, such as Google's word vector algorithm and Microsoft's deep neural network algorithm, often run on a single computer.

Based on the existing technologies, an efficient large-scale word vector training solution is needed.

SUMMARY

Embodiments of the present application provide a cluster-based word vector processing method, apparatus, and device, to resolve the following technical problem: an efficient large-scale word vector training solution is needed.

To resolve the foregoing technical problem, the embodiments of the present application are implemented as follows:

An embodiment of the present application provides a cluster-based word vector processing method, where the cluster includes a plurality of worker computers, and the method includes operations performed by each of the worker computers:

- obtaining a word and context words of the word that are extracted from a corpus or corpuses;

- obtaining word vectors for the word and the context words of the word; and

- training the corresponding word vectors according to the word and the context words of the word.

The method further includes updating word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.

An embodiment of the present application provides a cluster-based word vector processing apparatus, where the cluster includes a plurality of worker computers. The apparatus is located in the cluster, and includes an integration and update module and a training module that is located in a worker computer.

The training module of each of the worker computers performs the following steps:

- obtaining a word and context words of the word that are extracted from a corpus or corpuses;

- obtaining word vectors for the word and the context words of the word; and

- training the corresponding word vectors according to the word and the context words of the word.

The integration and update module updates word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.

An embodiment of the present application provides a cluster-based word vector processing device, where the device belongs to the cluster and includes:

- at least one processor; and

- a memory in communication connection with the at least one processor, where

- the memory stores an instruction executable by the at least one processor to cause the at least one processor to:

- obtain a word and context words of the word that are extracted from a corpus or corpuses;

- obtain word vectors for the word and the context words of the word;

- train the corresponding word vectors according to the word and the context words of the word; and

- update word vectors of same words according to one or more respective training results of the word vectors of the same words.

The foregoing at least one technical solution used in the embodiments of the present application can achieve the following beneficial effect: a distributed parallel processing capability of a cluster enables the solution to be applicable to large-scale word vector training and efficiency is relatively high.

BRIEF DESCRIPTION OF THE DRAWINGS

To describe the technical solutions in the embodiments of this specification or in the prior art more clearly, the following briefly introduces the accompanying drawings required for describing the embodiments or the prior art. Apparently, the accompanying drawings in the following description show merely some embodiments of this specification, and a person of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

FIG. 1 is a schematic diagram of an overall architecture of a cluster, according to a solution of the present application;

FIG. 2 is a schematic flowchart of a cluster-based word vector processing method, according to an embodiment of the present application;

FIG. 3 is a schematic principle diagram of a cluster-based word vector processing method in an actual application scenario, according to an embodiment of the present application;

FIG. 4 is a detailed schematic flowchart of a cluster-based word vector processing method corresponding to FIG. 3, according to an embodiment of the present application; and

FIG. 5 is a schematic structural diagram of a cluster-based word vector processing apparatus corresponding to FIG. 2, according to an embodiment of the present application.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present application provide a cluster-based word vector processing method, apparatus, and device.

To enable a person skilled in the art to better understand the technical solutions in this specification, the technical solutions of the embodiments of this specification will be described clearly and thoroughly below with reference to the accompanying drawings of the embodiments of this specification. Apparently, the described embodiments are merely some rather than all of the embodiments of this application. All other embodiments obtained by a person of ordinary skill in the art based on the embodiments of this specification without creative efforts shall fall within the protection scope of the present invention.

The solutions in the present application are applicable to a cluster, and efficiency in processing large-scale word vectors in the cluster is higher. Specifically, training corpuses may be divided, and then a plurality of worker computers in the cluster separately trains word vectors in a distributed manner corresponding to a corpus or corpuses obtained through

division of the portion of a corpus or corpuses. Because some corpuses may include same words, respective training results of word vectors of the same words by all the worker computers are integrated, to help further update the word vectors of the same words that have been stored before the training.

The solutions may involve one or more clusters. For example, in FIG. 1, the solutions involve to two clusters.

FIG. 1 is a schematic diagram of an overall architecture of a cluster involved in an actual application scenario, according to a solution of the present application. In the overall architecture, three parts are mainly involved: a server cluster, a worker computer cluster, and a database. The database stores corpuses used for training, for the worker computer cluster to read. The server cluster stores original word vectors, and the worker computer cluster cooperates with the server cluster to train word vectors and update the word vectors in the server cluster according to training results.

The architecture in FIG. 1 is exemplary, not unique. For example, the solution may alternatively involve only one cluster, or the cluster includes at least one scheduler and a plurality of worker computers, and the scheduler completes the work of the foregoing server cluster; or the solution may alternatively involve one worker computer cluster and one server; and so on.

The following describes the solutions of the present application in detail.

FIG. 2 is a schematic flowchart of a cluster-based word vector processing method, according to an embodiment of the present application. The cluster includes a plurality of worker computers. Steps in FIG. 2 are performed by at least one computer (or a program on a computer) in the cluster, and different steps may be performed by different execution bodies. The procedure in FIG. 2 may be performed a plurality of rounds, and a different group of corpuses may be used in each round.

The procedure in FIG. 2 includes the following steps:

S202: Each of the worker computers included in the cluster separately obtains a word and context words of the word that are extracted from a corpus or corpuses.

S204: The worker computer obtains word vectors for the word and the context words of the word.

S206: The worker computer trains the corresponding word vectors according to the word and the context words of the word.

S208: The cluster updates word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.

In the present embodiment of the present application, each worker computer may perform steps S202 to S206 in parallel in a distributed manner. Each worker computer usually corresponds to some different corpuses. In this case, large-scale training corpuses can be efficiently used, and word vector training efficiency can also be improved. For example, corpuses currently used for training word vectors may be divided into a plurality of parts, each worker computer may read one part, and then perform steps S202 to S206 based on the part of the corpuses read by the worker computer.

For ease of description, steps S202 to S204 are described mainly from a perspective of a particular worker computer in the following embodiments.

In an embodiment of the present application, if a current round of procedure is the first round of procedure, the word vectors obtained in step S204 may be obtained through initialization. For example, a word vector of each word and a word vector of context words of the word may be initialized randomly or according to a specified probability distribution, and the specified probability distribution is, for example, a 0-1 distribution. If the current round of procedure is not the first round of procedure, the word vectors obtained in step S204 may be word vectors that are updated and stored after a previous round of procedure is performed.

In an embodiment of the present application, generally, step S208 may be performed by a server cluster other than a worker computer cluster, or performed by a scheduler or a server that belongs to a same cluster as the worker computer, to reduce load of the worker computer. Correspondingly, the updated word vectors may be stored in the server, for use in a next round of procedure.

The rest can be deduced by analogy. After a plurality of rounds of procedure is performed until all groups of training corpuses are used, word vectors finally obtained through updating may be written into a database, for use in various scenarios that require the word vectors, or may still be stored in the cluster.

Based on the method in FIG. 2, a distributed parallel processing capability of the cluster enables the method to be applicable to large-scale word vector training and efficiency is relatively high. In addition, large-scale training data can also be efficiently used.

Based on the method in FIG. 2, the present embodiment of the present application further provides some specific implementation solutions of the method and expanded solutions. The architecture in FIG. 1 is used as an example below for description.

In the present embodiment of the present application, based on the architecture in FIG. 1, the cluster in FIG. 2 includes a server cluster and a worker computer cluster, the worker computer cluster performs steps S202 to S206, and the server cluster performs step S208.

The architecture in FIG. 1 may alternatively be referred to as a parameter server, and the parameter server can be used to meet common parallel requirements: data parallel and model parallel. Data parallel means that each computer loads different training data and synchronizes model training, and global information may be synchronized at regular intervals. Model parallel means that each computer loads only some model parameters, and model parameters loaded on all computer are put together as a full amount of model parameters.

The server cluster mainly implements model parallel, that is, the server cluster maintains a full amount of model parameters in a memory, while the worker computer cluster reads different training data for training in parallel. An entire process is as follows: The server cluster distributes parameters to the worker computer cluster (each worker computer may read the full amount of model parameters, or may read only some of the model parameters). Each worker computer reads different training data, starts parallel training, and updates model parameters in the worker computer. The worker computer cluster transmits trained model parameters back to the server cluster. The server cluster aggregates all updated data, that is, updates a model, and then transmits new model parameters to the worker computer cluster. This process is repeated, until all training data is trained, or a maximum quantity of training times is reached. In terms of the specific scenario of the present application, the foregoing training data may be a corpus, and the model parameter may be a word vector.

In the present embodiment of the present application, extracting a word and context words of the word may be performed by the worker computer, or may be performed in advance by another device. For example, in the former manner, before step S202 of obtaining a word and context words of the word that are extracted from a corpus or corpuses, the following may further be performed: obtaining, by each of the worker computers, a corpus or corpuses through distributed reading. If the corpuses are stored in the database, the corpuses may be read from the database.

In the present embodiment of the present application, the obtaining a word and context words of the word that are extracted from a corpus or corpuses may specifically include: creating a corresponding word pair according to the corpus obtained through reading, where the word pair includes a current word and a context word of the current word. For example, a word in the corpus obtained through reading may be scanned, and a currently scanned word is the current word and denoted as w . A sliding window including w is determined according to a specified sliding window distance. Each other word in the sliding window is used as a context word of w and denoted as c . In this way, a word pair $\{w, c\}$ is formed.

Further, it is assumed that word vectors are stored in a plurality of servers included in the server cluster. In this case, step S204 of obtaining word vectors for the word and the context words of the word may specifically include: obtaining a current word set and a context word set through extraction according to the created word pair; and obtaining word vectors of words included in the current word set and the context word set from the server. Certainly, this is not a unique implementation. For example, a word vector of a currently scanned word may be obtained from the server when the corpus is scanned, and the created word pair may not need to be relied on.

In the present embodiment of the present application, the corresponding word vectors may be trained according to a designated loss function and the created word pair.

To obtain a better training effect and achieve faster convergence, training may further be performed in combination with a designated negative sample word. The negative sample word is considered as a word having a lower correlation with the corresponding current word than the context word, and generally, several negative sample words may be randomly selected from all words. In such a case, step S206 of training the corresponding word vectors according to the word and the context words of the word may specifically include: training the corresponding word vectors according to the designated loss function, the designated negative sample word, and the created word pair. The current word and each negative sample word may also constitute a word pair. Assuming that there are λ negative sample words, the corresponding words may be denoted as $\{w, c'_1\}$, $\{w, c'_2\}$, ..., and $\{w, c'_\lambda\}$. For ease of description, the negative sample word pairs and the foregoing context word pairs are collectively denoted as $\{w, c\}$, and y is used for distinguishing. For a context word pair, $y=1$, and for a negative sample word pair, $y=0$.

For ease of understanding, an example of the loss function is provided as follows:

$$L(w, c) = \log \sigma(\bar{w} \cdot \bar{c}) + \lambda E_{c' \sim P(D)} \left[-\log \sigma(\bar{w} \cdot \bar{c}') \right],$$

where $L(w, c)$ represents the loss function, c' represents the negative sample word, $\bar{w}$ represents a word vector of w , $\bar{c}$ represents a word vector of c , $\bar{c}'$ represents a word vector of c' , λ is a quantity of negative sample words of w , and σ is an activation function, such as a sigmoid function.

Certainly, in addition to the foregoing example, the loss function may alternatively be implemented in another form, and a training target is to make a similarity of $\bar{w}$ and $\bar{c}$ as

great as possible, and a similarity of $\vec{w}$ and $\vec{c}'$ as small as possible. In the foregoing example, the similarity is measured through vector point multiplication, and the similarity may alternatively be measured in another manner.

Further, if the word vectors are trained by using a gradient descent method, the training the corresponding word vectors according to a designated loss function, a designated negative sample word, and the created word pair may specifically include: traversing the corpus obtained through reading; and according to the designated loss function, the designated negative sample word, and the created word pair, calculating a gradient according to the designated loss function, the designated negative sample word, and the created word pair, and iteratively updating word vectors of traversed current words and context words of the current words according to the gradient.

In an actual application, one or more threads on each worker computer may iteratively update the word vectors of the traversed current words and context words of the current words in a manner of asynchronous calculation without locking. Therefore, the threads on the worker computer may also perform updating in parallel without interfering with each other, thereby further improving training efficiency.

In the present embodiment of the present application, the gradient and the training result may vary with different loss functions and different activation functions. The foregoing example of the loss function is still used to describe calculation during training.

The calculating a gradient according to the designated loss function, the designated negative sample word, and the created word pair, and iteratively updating word vectors of traversed current words and context words of the current words according to the gradient may specifically include:

iteratively updating the word vectors of the traversed current words and context words of the current words according to the following formulas:

$$\vec{w}_{i,t+1} = \vec{w}_{i,t} + g \vec{c}_{i,t}, w \in B_{i,k} \quad (\text{formula 1}) \text{ and}$$

$$\vec{c}_{i,t+1} = \vec{c}_{i,t} + g \vec{w}_{i,t}, c \in \Gamma(w) \quad (\text{formula 2}),$$

$$\text{where } g = \alpha(y - \sigma(\vec{w} \cdot \vec{c})), \quad y = \begin{cases} 1, \{w, c\} \\ 0, \{w, c'\} \end{cases}, \quad w \text{ represents a current word, } c$$

represents a context word of w , c' represents a negative sample word, $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,t}$ and $\vec{c}_{i,t}$ represent a t^{th}

update on an i^{th} worker computer, $B_{i,k}$ represents a k^{th} corpus on the i^{th} worker computer, $\Gamma(w)$ represents a context word set of w , α represents a learning rate, and for example, may be set to 0.025, and σ is a sigmoid function, namely, $\sigma = \frac{1}{1+e^{-x}}$.

Further, calculation of the gradient is described as follows:

$$\text{if } \nabla \sigma(z)|_z = \frac{1}{\sigma(z)} \cdot \sigma(z) \cdot (1 - \sigma(z)) = 1 - \sigma(z) \text{ and}$$

$$\nabla \sigma(-z)|_z = -(1 - \sigma(-z)) = -\sigma(z),$$

$$\nabla L(w, c)|_w = (y - \sigma(\vec{w} \cdot \vec{c})) \vec{c} \text{ and}$$

$$\nabla L(w, c)|_c = (y - \sigma(\vec{w} \cdot \vec{c})) \vec{w}.$$

In the present embodiment of the present application, when the server updates the word vectors according to the training results, step S208 of updating word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words may specifically include: obtaining, by the server, the training results of one or more of the worker computers with respect to the word vectors of the same words; and performing average calculation according to the training results and the word vectors of the same words that are stored before training, to obtain vector increment values, and updating the word vectors of the same words according to the vector increment values.

The update process is a model averaging process. The average calculation may be implemented in a plurality of manners, for example, a quantity of occurrences of a word on each worker computer is used as a weight to average the training results of the worker computer; for another example, the training results of the worker computer are averaged directly; and so on. The former manner is used as an example, and the foregoing vector increment values may be obtained through calculation according to the following formula:

$$\Delta(\vec{w}) = \frac{\sum_{i=0}^I \lambda_i(w) (\vec{w}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(w)} \quad (\text{formula 3) and}$$

$$\Delta(\vec{c}) = \frac{\sum_{i=0}^I \lambda_i(c) (\vec{c}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(c)} \quad (\text{formula 4),}$$

$\vec{w}_{i,T}$ and $\vec{c}_{i,T}$ represent an iterative update result on an i^{th} worker computer, $\lambda_i(w)$

represents a quantity of occurrences of w on the i^{th} worker computer, and $\vec{w}_{srv}$ represents $\vec{w}$ stored by a respective server before the training.

A word vector before the update and the calculated corresponding vector increment value are added to obtain an updated word vector.

According to the foregoing description, an embodiment of the present application further provides a schematic principle diagram of a cluster-based word vector processing method in an actual application scenario, as shown in FIG. 3. Further, an embodiment of the present application further provides a detailed schematic flowchart of a cluster-based word vector processing method corresponding to FIG. 3, as shown in FIG. 4.

FIG. 3 exemplarily shows worker computers 0 to 2 and servers 0 to 2. Description is provided mainly for the worker computer 0. The worker computers 1 and 2 are briefly represented, and worker modes of the worker computers 1 and 2 are consistent with a worker mode of the worker computer 0. "wid" and "cid" are identifiers and respectively represent the current word and the context word. "wid list" and "cid list" are lists of identifiers and respectively represent the current word set and the context word set. A brief worker procedure in FIG. 3 includes: each worker computer reads corpuses in a distributed manner, and creates a word pair; each worker computer obtains corresponding word vectors from the server cluster; each worker computer trains the word vectors by using the read corpuses; and the server cluster performs model averaging according to training results of the worker computers.

FIG. 4 shows a more detailed procedure, which mainly includes the following steps:

S402: Each worker computer reads in a distributed manner a corpus or corpuses, creates a word pair $\{w, c\}$, and extracts wid list and cid list from the word pair, as shown by the worker computer 0 in FIG. 3.

S404: The worker computer obtains corresponding word vectors from the server cluster according to wid list and cid list.

S406: The worker computer calculates a gradient according to the word pair, then iteratively updates the word vectors, and specifically performs calculation by using the foregoing formula 1 and formula 2.

S408: After each worker computer performs iterative updating, the server cluster performs model averaging, to integrate iterative update results, and specifically performs calculation by using the foregoing formula 3 and formula 4.

Based on a same idea, an embodiment of the present application further provides an

apparatus corresponding to the foregoing method, as shown in FIG. 5.

FIG. 5 is a schematic structural diagram of a cluster-based word vector processing apparatus corresponding to FIG. 2, according to an embodiment of the present application. The cluster includes a plurality of worker computers, and the apparatus is located in the cluster, and includes an integration and update module 501 and a training module 502 that is located in the worker computer.

The training module 502 of each of the worker computers separately performs the following steps:

- obtaining a word and context words of the word that are extracted from a corpus or corpuses;

- obtaining word vectors for the word and the context words of the word; and

- training the corresponding word vectors according to the word and the context words of the word; and

- the integration and update module 501 updates word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.

Optionally, the worker computer further includes a reading module 503, and before the training module 502 obtains the word and the context words of the word that are extracted from a corpus or corpuses, the reading module 503 of each of the worker computers obtains a corpus or corpuses through distributed reading; and

- that the training module 502 obtains the word and the context words of the word that are extracted from a corpus or corpuses specifically includes:

- the training module 502 creates a corresponding word pair according to the corpus obtained by the reading module 503 of the worker computer in which the training module 502 is located, where the word pair includes a current word and a context word of the current word.

Optionally, the cluster further includes a plurality of servers, and that the training module 502 obtains the word vectors for the word and the context words of the word specifically includes:

- the training module 502 obtains a current word set and a context word set through extraction according to each word pair created by the training module 502; and

- obtains word vectors of words included in the current word set and the context word set from the server.

- Optionally, that the training module 502 trains the corresponding word vectors according

to the word and the context words of the word specifically includes:

the training module 502 trains the corresponding word vectors according to a designated loss function, a designated negative sample word, and each word pair created by the training module 502.

Optionally, that the training module 502 trains the corresponding word vectors according to the designated loss function, the designated negative sample word, and each word pair created by the training module 502 specifically includes:

the training module 502 traverses the corpus obtained through reading; and

calculates a gradient according to the designated loss function, the designated negative sample word, and each word pair created by the training module 502, and iteratively updates word vectors of traversed current words and context words of the current words according to the gradient.

Optionally, that the training module 502 calculates the gradient according to the designated loss function, the designated negative sample word, and each word pair created by the training module 502, and iteratively updates the word vectors of the traversed current words and context words of the current words according to the gradient specifically includes:

the training module 502 iteratively updates the word vectors of the traversed current words and context words of the current words according to the following formulas:

$$\vec{w}_{i,t+1} = \vec{w}_{i,t} + g \vec{c}_{i,t}, w \in B_{i,k} \text{ and}$$

$$\vec{c}_{i,t+1} = \vec{c}_{i,t} + g \vec{w}_{i,t}, c \in \Gamma(w),$$

$$\text{where } g = \alpha \left(y - \sigma(\vec{w} \cdot \vec{c}) \right), \quad y = \begin{cases} 1, \{w, c\} \\ 0, \{w, c'\} \end{cases}, \quad w \text{ represents a current word, } c$$

represents a context word of w , c' represents a negative sample word, $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,t}$ and $\vec{c}_{i,t}$ represent a t^{th} update on an i^{th} worker computer, $B_{i,k}$ represents a k^{th} corpus on the i^{th} worker computer, $\Gamma(w)$ represents a context word set of w , α represents a learning rate, and σ is a sigmoid function.

Optionally, that the training module 502 iteratively updates the word vectors of the traversed current words and context words of the current words specifically includes:

the training module 502 iteratively updates, by using one or more threads on the worker computer, the word vectors of the traversed current words and context words of the current

words in a manner of asynchronous calculation without locking.

Optionally, the integration and update module 501 is located in the server, and that the integration and update module 501 updates the word vectors of the same words according to the training results of one or more of the worker computers with respect to the word vectors of the same words specifically includes:

the integration and update module 501 obtains the training results of one or more of the worker computers with respect to the word vectors of the same words; and

performs average calculation according to the training results and the word vectors of the same words that are stored before training, to obtain vector increment values, and updates the word vectors of the same words according to the vector increment values.

Optionally, that the integration and update module 501 performs average calculation according to the training results and the word vectors of the same words that are stored before training, to obtain the vector increment values specifically includes:

the integration and update module 501 obtains the vector increment values through calculation according to the following formulas:

$$\Delta(\vec{w}) = \frac{\sum_{i=0}^I \lambda_i(w) (\vec{w}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(w)} \text{ and}$$

$$\Delta(\vec{c}) = \frac{\sum_{i=0}^I \lambda_i(c) (\vec{c}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(c)},$$

where w represents a current word, c represents a context word of w , $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,T}$ and $\vec{c}_{i,T}$ represent an iterative update result on an i^{th} worker computer, $\lambda_i(w)$ represents a quantity of occurrences of w on the i^{th} worker computer, and $\vec{w}_{srv}$ represents $\vec{w}$ stored by a respective server before the training.

Based on a same idea, an embodiment of the present application further provides a cluster-based word vector processing device corresponding to FIG. 2, and the device belongs to the cluster and includes:

at least one processor; and

a memory in communication connection with the at least one processor, where

the memory stores an instruction executable by the at least one processor to cause the at least one processor to:

obtain a word and context words of the word that are extracted from a corpus or corpuses;

obtain word vectors for the word and the context words of the word;

train the corresponding word vectors according to the word and the context words of the word; and

update word vectors of same words according to one or more respective training results of the word vectors of the same words. Based on a same idea, an embodiment of the present application further provides a non-volatile computer storage medium corresponding to FIG. 2. The non-volatile computer storage medium stores a computer executable instruction, and the computer executable instruction is set to be:

obtaining a word and context words of the word that are extracted from a corpus or corpuses;

obtaining word vectors for the word and the context words of the word; and

training the corresponding word vectors according to the word and the context words of the word; and

updating word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.

Embodiments of this specification are described above. Other embodiments fall within the scope of the appended claims. In some embodiments, the actions or steps recorded in the claims may be performed in sequences different from those in the embodiments and an expected result may still be achieved. In addition, the processes depicted in the accompanying drawings do not necessarily require specific sequences or consecutive sequences to achieve an expected result. In some implementations, multitasking and parallel processing may be feasible or beneficial.

The embodiments of this specification are all described in a progressive manner, for same or similar parts in the embodiments, refer to these embodiments, and descriptions of each embodiment focus on a difference from other embodiments. Especially, an apparatus embodiment, a device embodiment, or a non-volatile computer storage medium embodiment is basically similar to a method embodiment, and therefore is described briefly; for related parts, refer to partial descriptions in the method embodiment.

The apparatus, the device, and the non-volatile computer storage medium provided in the embodiments of this specification correspond to the method, so that the apparatus, the device, and the non-volatile computer storage medium also have beneficial technical effects similar to those of the corresponding method. Since the beneficial technical effects of the method

have been described in detail above, the beneficial technical effects of the corresponding apparatus, device, and non-volatile computer storage medium are not described herein again.

In the 1990s, improvements of a technology can be clearly distinguished between hardware improvements (for example, improvements to a circuit structure such as a diode, a transistor, a switch, etc.) and software improvements (improvements to a method procedure). However, with the development of technology, improvements of many method procedures can be considered as direct improvements of hardware circuit structures. Designers almost all program an improved method procedure to a hardware circuit, to obtain a corresponding hardware circuit structure. Therefore, it does not mean that the improvement of a method procedure cannot be implemented by using a hardware entity module. For example, a programmable logic device (PLD) such as a field programmable gate array (FPGA) is a type of integrated circuit whose logic function is determined by a user by programming the device. The designers perform voluntary programming to "integrate" a digital system into a single PLD without requiring a chip manufacturer to design and prepare a dedicated integrated circuit chip. Moreover, nowadays, instead of manually making integrated circuit chips, this programming is mostly implemented by using "logic compiler" software, which is similar to the software compiler used in program development and writing. The original code is written in a specific programming language before compiling, and this language is referred to as a hardware description language (HDL). There are various kinds of HDLs, for example, advanced Boolean expression language (ABEL), altera hardware description language (AHDL), Confluence, Cornell university programming language (CUPL), HDCal, Java hardware description language (JHDL), Lava, Lola, MyHDL, PALASM, Ruby hardware description language (RHDL), and the like. Currently, the most commonly used HDLs are very-high-speed integrated circuit hardware description language (VHDL) and Verilog. A person skilled in the art should also understand that as long as a method procedure is logically programmed and then programmed to an integrated circuit by using the foregoing hardware description languages, a hardware circuit that implements the logical method procedure can be easily obtained.

The controller can be implemented in any suitable manner, for example, the controller can take the form of, for example, a microprocessor or processor and a computer-readable medium storing computer-readable program code (for example, software or firmware) executable by the processor, a logic gate, a switch, an application-specific integrated circuit (ASIC), a programmable logic controller and an embedded microcontroller. Examples of the controller include, but are not limited to, the following microcontrollers: ARC 625D, Atmel

AT91SAM, Microchip PIC18F26K20 and Silicone Labs C8051F320. The memory controller can also be implemented as part of the memory control logic. A person skilled in the art will also appreciate that, in addition to implementing the controller in the form of pure computer-readable program code, it is also possible to implement the controller in the form of a logic gate, switch, application-specific integrated circuit, programmable logic controller, and embedded microcontroller and other forms to achieve the same function. Such a controller can thus be considered as a hardware component and apparatuses included therein for implementing various functions can also be considered as structures inside the hardware component. Alternatively, apparatuses configured to implement various functions can be considered as both software modules implementing the method and structures inside the hardware component.

The system, the apparatus, the module or the unit described in the foregoing embodiments can be specifically implemented by a computer chip or an entity or implemented by a product having a certain function. A typical implementation device is a computer. Specifically, the computer can be, for example, a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email device, a game console, a tablet computer, a wearable device, or a combination of any of these devices.

For ease of description, when the apparatus is described, the apparatus is divided into units according to functions, which are separately described. Certainly, during implementation of this specification, the function of the units may be implemented in a same piece of or multiple pieces of software and/or hardware.

A person skilled in the art should understand that the embodiments of this specification may be provided as a method, a system, or a computer program product. Therefore, the embodiments of this specification may use a form of hardware only embodiments, software only embodiments, or embodiments with a combination of software and hardware. Moreover, the embodiments of this specification may use a form of a computer program product that is implemented on one or more computer-usable storage media (including but not limited to a disk memory, a CD-ROM, an optical memory, and the like) that include computer-usable program code.

This specification is described with reference to the flowchart and/or block diagram of the method, device (system), and computer program product of the embodiments of this specification. It should be understood that computer program instructions can implement each procedure and/or block in the flowcharts and/or block diagrams and a combination of

procedures and/or blocks in the flowcharts and/or block diagrams. These computer program instructions may be provided to a general-purpose computer, a special-purpose computer, an embedded processor, or a processor of another programmable data processing device to generate a machine, so that an apparatus configured to implement functions specified in one or more procedures in the flowcharts and/or one or more blocks in the block diagrams is generated by using instructions executed by the general-purpose computer or the processor of another programmable data processing device.

These computer program instructions may also be stored in a computer readable memory that can guide a computer or another programmable data processing device to work in a specific manner, so that the instructions stored in the computer readable memory generate a product including an instruction apparatus, where the instruction apparatus implements functions specified in one or more procedures in the flowcharts and/or one or more blocks in the block diagrams.

These computer program instructions may also be loaded into a computer or another programmable data processing device, so that a series of operation steps are performed on the computer or another programmable data processing device to generate processing implemented by a computer, and instructions executed on the computer or another programmable data processing device provide steps for implementing functions specified in one or more procedures in the flowcharts and/or one or more blocks in the block diagrams.

In a typical configuration, the computer device includes one or more processors (CPUs), an input/output interface, a network interface, and a memory.

The memory may include a form such as a persistent memory, a random-access memory (RAM) and/or a non-volatile memory of computer readable media, for example, a read-only memory (ROM) or a flash memory (RAM). The memory is an example of the computer readable medium.

The computer readable medium includes a persistent medium and a non-persistent medium, a removable medium and a non-removable medium, which may implement storage of information by using any method or technology. The information may be a computer readable instruction, a data structure, a module of a program or other data. Examples of computer storage media include but are not limited to a phase change memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), other type of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or other memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or other optical

storage, a cassette magnetic tape, tape and disk storage or other magnetic storage device or any other non-transmission media that may be configured to store information that a computing device can access. Based on the definition in the present disclosure, the computer-readable medium does not include transitory computer readable media (transitory media), such as a modulated data signal and a carrier.

It should also be noted that the terms "include", "comprise" and any other variants mean to cover the non-exclusive inclusion. Thereby, the process, method, article, or device which include a series of elements not only include those elements, but also include other elements which are not clearly listed, or include the inherent elements of the process, method, article and device. Without further limitation, the element defined by a phrase "include one" does not exclude other same elements in the process, method, article or device which include the element.

This specification can be described in the general context of computer-executable instructions executed by a computer, for example, a program module. Generally, the program module includes a routine, a program, an object, a component, a data structure, and the like for executing a particular task or implementing a particular abstract data type. This specification may also be practiced in distributed computing environments in which tasks are performed by remote processing devices that are connected by using a communications network. In a distributed computing environment, the program module may be located in both local and remote computer storage media including storage devices.

The embodiments of this specification are all described in a progressive manner, for same or similar parts in the embodiments, refer to these embodiments, and descriptions of each embodiment focus on a difference from other embodiments. Especially, a system embodiment is basically similar to a method embodiment, and therefore is described briefly; for related parts, reference may be made to partial descriptions in the method embodiment.

The foregoing descriptions are merely embodiments of this specification and are not intended to limit this application. For a person skilled in the art, various modifications and variations can be made to this application. Any modification, equivalent replacement, or improvement made without departing from the spirit and principle of this application shall fall within the scope of the claims of this application.

CLAIMS

What is claimed is:

1. A cluster-based word vector processing method performed at a cluster comprising a plurality of worker computers,
the method comprising operations performed by each of the worker computers:
 - obtaining a word and context words of the word that are extracted from a corpus;
 - obtaining word vectors for the word and the context words of the word; and
 - training the word vectors according to the word and the context words of the word; andthe method comprising updating word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.
2. The method according to claim 1, wherein before the obtaining a word and context words of the word that are extracted from a corpus, the method further comprises:
 - obtaining, by each of the worker computers, a respective portion of the corpus through distributed reading; and
 - the obtaining a word and context words of the word that are extracted from a corpus comprises:
 - creating a corresponding word pair according to the respective portion of the corpus obtained through the distributed reading, wherein the word pair comprises a current word and a context word of the current word.
3. The method according to claim 2, wherein the cluster further comprises a plurality of servers, and the obtaining word vectors for the word and the context words of the word comprises:
 - obtaining a current word set and a context word set through extraction according to the created word pair; and
 - obtaining word vectors for words included in the current word set and the context word set from a respective server.
4. The method according to claim 2, wherein the training the word vectors according to the word and the context words of the word comprises:

training the word vectors according to a designated loss function, a designated negative sample word, and the created word pair.

5. The method according to claim 4, wherein the training the word vectors according to a designated loss function, a designated negative sample word, and the created word pair comprises:

traversing the corpus obtained through reading; and

calculating a gradient according to the designated loss function, the designated negative sample word, and the created word pair, and iteratively updating word vectors of traversed current words and context words of the current words according to the gradient.

6. The method according to claim 5, wherein the calculating a gradient according to the designated loss function, the designated negative sample word, and the created word pair, and iteratively updating word vectors of traversed current words and context words of the current words according to the gradient comprises:

iteratively updating the word vectors of the traversed current words and context words of the current words according to the following formulas:

$$\vec{w}_{i,t+1} = \vec{w}_{i,t} + g\vec{c}_{i,t}, w \in B_{i,k} \text{ and}$$

$$\vec{c}_{i,t+1} = \vec{c}_{i,t} + g\vec{w}_{i,t}, c \in \Gamma(w),$$

$$\text{wherein } g = \alpha(y - \sigma(\vec{w} \cdot \vec{c})), \quad y = \begin{cases} 1, \{w, c\} \\ 0, \{w, c'\} \end{cases}, \quad w \text{ represents a current word, } c$$

represents a context word of w , c' represents a negative sample word, $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,t}$ and $\vec{c}_{i,t}$ represent a t^{th} update on an i^{th} worker computer, $B_{i,k}$ represents a k^{th} corpus on the i^{th} worker computer, $\Gamma(w)$ represents a context word set of w , α represents a learning rate, and σ is a sigmoid function.

7. The method according to claim 6, wherein the iteratively updating the word vectors of the traversed current words and context words of the current words comprises:

iteratively updating, by using one or more threads on a respective worker computer, the word vectors of the traversed current words and context words of the current words in a

manner of asynchronous calculation without locking.

8. The method according to claim 3, wherein the updating word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words comprises:

obtaining, by the respective server, the training results of the one or more of the worker computers with respect to the word vectors of the same words; and

performing average calculation according to the training results and the word vectors of the same words that are stored before the training, to obtain vector increment values, and updating the word vectors of the same words according to the vector increment values.

9. The method according to claim 8, wherein the performing average calculation according to the training results and the word vectors of the same words that are stored before the training, to obtain vector increment values comprises:

obtaining the vector increment values through calculation according to the following formulas:

$$\Delta(\vec{w}) = \frac{\sum_{i=0}^I \lambda_i(w) (\vec{w}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(w)} \text{ and}$$

$$\Delta(\vec{c}) = \frac{\sum_{i=0}^I \lambda_i(c) (\vec{c}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(c)},$$

wherein w represents a current word, c represents a context word of w , $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,T}$ and $\vec{c}_{i,T}$ represent an iterative update result on an i^{th} worker computer, $\lambda_i(w)$ represents a quantity of occurrences of w on the i^{th} worker computer, and $\vec{w}_{srv}$ represents $\vec{w}$ stored by a respective server before the training.

10. A cluster-based word vector processing apparatus at a cluster, wherein the cluster comprises a plurality of worker computers, and the apparatus comprises an integration and update module, and a training module that is located in each of the worker computers;

wherein the training module of each of the worker computers separately performs operations including:

obtaining a word and context words of the word that are extracted from a corpus;
obtaining word vectors for the word and the context words of the word; and
training the word vectors according to the word and the context words of the word; and
wherein the integration and update module updates word vectors of same words according to training results of one or more of the worker computers with respect to the word vectors of the same words.

11. The apparatus according to claim 10, wherein each of the worker computers further comprises a reading module, and before the training module obtains the word and the context words of the word that are extracted from the portion of the corpus, the reading module of each of the worker computers obtains a respective portion of the corpus through distributed reading; and

the training module obtains the word and the context words of the word that are extracted from a corpus by:

creating a corresponding word pair according to the respective portion of the corpus obtained by the reading module of the respective worker computer in which the training module is located, wherein the word pair comprises a current word and a context word of the current word.

12. The apparatus according to claim 11, wherein the cluster further comprises a plurality of servers, and the training module obtains the word vectors for the word and the context words of the word by:

obtaining a current word set and a context word set through extraction according to each word pair created by the training module; and

obtaining word vectors for words included in the current word set and the context word set from a respective server.

13. The apparatus according to claim 11, wherein the training module trains the word vectors according to the word and the context words of the word by:

training the word vectors according to a designated loss function, a designated negative sample word, and the word pair created by the training module.

14. The apparatus according to claim 13, wherein the training module trains the word vectors according to the designated loss function, the designated negative sample word, and

the word pair created by the training module by:

traversing the corpus obtained through reading; and

calculating a gradient according to the designated loss function, the designated negative sample word, and the word pair created by the training module, and iteratively updating word vectors of traversed current words and context words of the current words according to the gradient.

15. The apparatus according to claim 14, wherein the training module calculates the gradient according to the designated loss function, the designated negative sample word, and the word pair created by the training module, and iteratively updates the word vectors of the traversed current words and context words of the current words according to the gradient by:

iteratively updating the word vectors of the traversed current words and context words of the current words according to the following formulas:

$$\vec{w}_{i,t+1} = \vec{w}_{i,t} + g \vec{c}_{i,t}, w \in B_{i,k} \text{ and}$$

$$\vec{c}_{i,t+1} = \vec{c}_{i,t} + g \vec{w}_{i,t}, c \in \Gamma(w),$$

$$\text{wherein } g = \alpha(y - \sigma(\vec{w} \cdot \vec{c})), \quad y = \begin{cases} 1, \{w, c\} \\ 0, \{w, c'\} \end{cases}, \quad w \text{ represents a current word, } c$$

represents a context word of w , c' represents a negative sample word, $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,t}$ and $\vec{c}_{i,t}$ represent a t^{th} update on an i^{th} worker computer, $B_{i,k}$ represents a k^{th} corpus on the i^{th} worker computer, $\Gamma(w)$ represents a context word set of w , α represents a learning rate, and σ is a sigmoid function.

16. The apparatus according to claim 15, wherein the training module iteratively updates the word vectors of the traversed current words and context words of the current words by:

iteratively updating, by using one or more threads on a respective worker computer, the word vectors of the traversed current words and context words of the current words in a manner of asynchronous calculation without locking.

17. The apparatus according to claim 12, wherein the integration and update module is located in a respective server, and the integration and update module updates the word

vectors of the same words according to training results of one or more of worker computers with respect to the word vectors of the same words by:

obtaining the training results of the one or more of the worker computers with respect to of the word vectors of the same words; and

performing average calculation according to the training results and the word vectors of the same words that are stored before the training, to obtain vector increment values, and updating the word vectors of the same words according to the vector increment values.

18. The apparatus according to claim 17, wherein the integration and update module performs average calculation according to the training results and the word vectors of the same words that are stored before the training, to obtain the vector increment values by:

obtaining the vector increment values through calculation according to the following formulas:

$$\Delta(\vec{w}) = \frac{\sum_{i=0}^I \lambda_i(w) (\vec{w}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(w)} \text{ and}$$

$$\Delta(\vec{c}) = \frac{\sum_{i=0}^I \lambda_i(c) (\vec{c}_{i,T} - \vec{w}_{srv})}{\sum_{i=0}^I \lambda_i(c)},$$

wherein w represents a current word, c represents a context word of w , $\vec{w}$ represents a word vector of w , $\vec{c}$ represents a word vector of c , $\vec{w}_{i,T}$ and $\vec{c}_{i,T}$ represent an iterative update result on an i^{th} worker computer, $\lambda_i(w)$ represents a quantity of occurrences of w on the i^{th} worker computer, and $\vec{w}_{srv}$ represents $\vec{w}$ stored by a respective server before the training.

19. A cluster-based word vector processing device, wherein the device belongs to a cluster and comprises:

at least one processor; and

a memory in communication connection with the at least one processor, wherein

the memory stores an instruction executable by the at least one processor to cause the at least one processor to:

obtain a word and context words of the word that are extracted from a corpus;

obtain word vectors for the word and the context words of the word;

train the word vectors according to the word and the context words of the word; and
update word vectors of same words according to one or more respective training results
of the word vectors of the same words.