Journal of Information Security and Applications 89 (2025) 103999

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Privacy-preserving word vectors learning using partially homomorphic encryption

Shang Ci [a], Sen Hu [a], Donghai Guan [a], Çetin Kaya Koç [a,b,c,*]

[a] Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China
[b] Iğdır University, Iğdır, 76000, Turkiye
[c] University of California Santa Barbara, Santa Barbara, 93106, United States

ARTICLE INFO

Keywords:
Privacy-preserving
Homomorphic encryption
Word vectors learning
Cloud computing

ABSTRACT

This paper introduces a privacy-preserving scheme for learning **GloVe** word vectors on encrypted data. Users first encrypt their private data using a partially homomorphic encryption algorithm and then send the ciphertext to a cloud server to execute the proposed scheme. The cloud server generates high-quality word vectors for subsequent machine learning tasks by filtering out disturbances. We conduct a theoretical analysis of the security and efficiency of the proposed approach. Experimental results on real-world datasets demonstrate that our scheme effectively trains word vectors without compromising user privacy or the integrity of the word vector model, while keeping the user-side implementation lightweight and offline.

1. Introduction

Machine learning (ML) can be trained on large-scale datasets to obtain a high-precision classification, recognition and prediction models, which have applications in various fields. Certain classes of classical artificial intelligence tasks such as text recognition [1], image classification [2] and machine translation [3] have seen significant progress with the help of ML algorithms. However, ML requires a large amount of computing and storage overhead, and ordinary users cannot perform ML training independently locally due to resource constraints. Therefore, users have a strong desire to outsource local data to a cloud server, and use the powerful computing and storage capacity of cloud servers to efficiently complete ML tasks, thus reducing the huge costs.

Cloud computing provides distributed computing and storage resources for users' various personalized services. In order to lower the operation costs, more and more users now outsource their data and computing needs to cloud service providers. However, due to the different levels of trust between the data owner and the cloud service providers, there are data security issues [4]. To deal with these security issues, private data are encrypted before being outsourced, and the cloud server receives encrypted data. However, in ciphertext domain, the cloud server cannot effectively provide various services such as query and computation. The training process of machine learning becomes more difficult to execute compared with the plaintext domain.

Secure computing ML technology based on ciphertext has been the main research direction, such as secure multi-party computation and homomorphic encryption. The secure multi-party computation usually uses secret sharing of data from multiple participants to ensure privacy and security. Its computing efficiency is close to plaintext computing, such as SecureML [5] and Quotient [6]. However, this technology requires participants to participate in computations online, which may reduce the user's computing experience. Homomorphic encryption technology is usually used for privacy-preserving ML in the cloud environment. The cloud side uses homomorphic operations to securely implement ML algorithms through the collected encrypted data. In this process, users only need to encrypt the data, and expensive computing operations are outsourced to cloud servers. It makes the user side lightweight. Moreover, users can offline, do not need to participate in privacy computations and more interactive operations. Therefore, this paper uses homomorphic encryption technology to complete the privacy-preserving ML task.

Natural Language Processing (NLP) is an important research field in ML, and it is a technology of interactive communication between human and machine. In a variety of NLP tasks, the **word** is always the smallest unit of processing. Expressing words in a vector space reveals the laws of language and its syntax and semantics. By training the corpus to generate word vectors, these trained word vectors can achieve better performance in subsequent machine learning tasks. **GloVe** [7] is a statistic-based model, which uses global statistical features and local context features to predict the frequency of each word in each context.

To solve these problems, *a privacy-preserving **GloVe** word vectors learning scheme on encrypted data is proposed in this paper*. To this

* Corresponding author at: University of California Santa Barbara, Santa Barbara, 93106, United States.
E-mail addresses: cishang@nuaa.edu.cn (S. Ci), hu_sen@nuaa.edu.cn (S. Hu), dhguan@nuaa.edu.cn (D. Guan), cetinkoc@ucsb.edu (Ç.K. Koç).

https://doi.org/10.1016/j.jisa.2025.103999

end, our scheme solves three major problems arising from training on large-scale ciphertext data. **First**, to address user side overhead, we use a dual-cloud model to participate in the training, which has been widely studied and applied in other projects [8–10]. The dual-cloud model can take advantage of the cloud server by taking on more computing and communication tasks to reduce the cost of the user side and provide more security. **Second**, to protect data privacy, the Paillier public-key cryptosystem is used to encrypt the plaintext space, and the real numbers involved in the encryption computation are transformed into integers by fixed-point representation. For logarithm encryption in **GloVe** word vectors learning, using Taylor series to perform polynomial approximation loses only a small amount of precision. In the computation of ciphertext multiplication and power operation, the secure multiplication protocol is used for effective computation. **Third**, in order to ensure the safe operation of **GloVe** word vectors learning method, security weight calculation and security comparison algorithms are developed, which implement secure computing between two cloud servers based on the Paillier homomorphic encryption. The main contributions of this paper are summarized as follows:

- We designed and implemented secure weight function and secure comparison algorithms to enable the cloud server to perform some basic operations safely.
- We used the designed data encryption scheme to implement secure word vectors learning on large-scale ciphertext data. It not only ensures the privacy of user data and the security of the result of word vectors, but also protects the security of the model of word vectors and resists collusion attack. In addition, the low computing complexity of the user side is ensured, and the user need not be online all the time.
- We evaluated the proposed scheme on representative real-world datasets, and the experimental results show that our scheme has high practicability and effectiveness, and compared with the results obtained in the plaintext domain. Theoretical analysis shows the security of our scheme.

The remainder of this paper is summarized as follows. Section 2 reviews the work on privacy-preserving machine learning. Section 3 introduces some basic knowledge. The proposed system model is described in Section 4. In Section 5, we construct our scheme in detail. Theoretical analysis and experimental evaluation are carried out in Sections 6 and 7, respectively. Finally, Section 8 summarizes our paper.

2. Related works

In the field of NLP, there are three major privacy issues: (1) when text data is made directly public, the content of the text is subject to privacy threats [11]; (2) information about the model (such as the number of neural network layers) may be leaked to an attacker during the execution of the training [12]; (3) privacy attacks against ML models for NLP [13].

From the perspective of data, the most common privacy issues of text data are related to its content, including some personal information. Martinelli et al. [14] trained artificial intelligence models in general knowledge domains to recognize and label private information in corpora using hybrid NLP techniques. When publishing documents such as medical records or government reports, the author's information is usually deleted to protect privacy. However, the malicious model may re-identify the private information. Therefore, document re-identification must provide formal and effective protection, such as the differential privacy model [15]. Qu et al. [16] proposed an adaptive language model **BERT** pre-training scheme based on differential privacy, which can improve the effectiveness of the model while retaining a high level of privacy.

From the perspective of NLP model, the relevant information of the model (such as bias and medical data) can be used by adversaries to carry out attacks, thereby revealing private data for training. In emotional analysis, Sweeney et al. [17] used antagonistic learning to remove the association between word vectors and emotions in demographic models, thus eliminating the affective bias in word embedding. It can protect emotional analysis from the social dimension. In the embedding layer of the neural network, NLP models for healthcare data are often threatened by shared vocabulary dictionaries, including patient identity information, disease type, etc. Alawad et al. [12] proposed a privacy-preserving transmission learning method based on deep learning NLP without sharing private data. Wang et al. [18] studied the use of neural network learning algorithms to train word vectors in secure collaboration on large-scale encrypted data, which can protect a large amount of potential private data. However, this scheme requires the server to distribute the private key to the users. In the process, if the server and some users try to collusion attack, they will combine their own information to infer the private data of other users, there may be a risk of privacy disclosure.

From the perspective of privacy attack, external attacks may expose personal privacy information, mainly including adversarial attacks, member inference attacks, and eavesdropping attacks. Malicious modification of text data does not affect readability, but enables the machine learning model to output erroneous tags, which constitutes an adversarial attack [19]. In the member inference attack, the word vectors of the general language model capture lots of the privacy information from the original input data. If the privacy information accessed by an adversary, these word vectors will be reverse engineered, the privacy of the user will be compromised [20]. The eavesdropping attacks occur in multiple scenarios with computing devices. The attacker can eavesdrop on the hidden representation of a text classifier in the neural network and attempt to recover the privacy information about input text [21].

Recently, other privacy preserving NLP models also need attention. Zhang et al. [22] used functional encryption to protect the privacy data of participants and provided an efficient scheme for training secure word vectors with functional encryption with inner product predicates. However, this scheme only considers the situation where there is no collusion between the crypto server and the remote server, and does not consider the collusion between the users and the remote server. Hua et al. [23] used a lightweight encryption technique to train word vectors, which utilizes two cloud computing servers to participate in the computation process to protect users privacy data. It greatly improves computational efficiency. However, communication costs between users and two servers, as well as between the two servers, may also need to be considered. Kim et al. [24] proposed an efficient look-up table evaluation algorithm and encryption indicator function to protect privacy in machine learning. CKKS fully homomorphic encryption was used to ensure computational accuracy, and the high efficiency and memory optimization advantages of this scheme were verified in some word embedding models. Moghaddam et al. [25] proposed a privacy-preserving sentiment analysis method using CKKS fully homomorphic encryption on pre-trained deep learning models. This scheme encrypts critical data from the client to protect privacy and reduce computational costs on the server. However, this scheme did not consider the computational burden on the client side.

Summary: Currently, privacy protection measures are mainly used to protect the security of data and models, including differential privacy and homomorphic encryption. Differential privacy is a cryptographic technique that protects privacy by adding noise [15]. Adding a small amount of noise can achieve effective privacy protection, making differential privacy technology easier to deploy and apply in practical scenarios [16]. In machine learning, it is generally used to protect the privacy of training datasets and model parameters. However, training more complex NLP models with differential privacy may impact the usability of the model.

Homomorphic encryption allows computations to be performed directly on ciphertext, and the decrypted result after the operation is the same as the result of the operation on the plaintext.

Table 1
Notation descriptions.

Notation	Description
$[\![\cdot]\!]$	Encrypted value
CS_1	The public cloud
CS_2	The private cloud
f_i	ith user data file
$n(f_i)$	Total number of words
$(w_1, \dots, w_{n(f_i)})$	Sequence of words
V	Size of vocabulary
d	The dimension of word vectors
X	Co-occurrence matrix
X_{ij}	Number of times word j occurs in the context of word i

Homomorphic encryption is commonly used in outsourced computing and storage to protect privacy. In the context of privacy preserving NLP model training, fully homomorphic encryption (FHE) and partially homomorphic encryption (PHE) are mainly used [23–25]. FHE can compute any circuit of infinite depth, while PHE supports evaluating circuits containing only one type of gate (such as addition or multiplication). FHE supports calculations on integers and real numbers. While FHE theoretically enables arbitrary calculations, it requires a fixed circuit depth, which restricts the ability to perform infinite addition and multiplication operations. Although there are currently some real-valued operations and optimized FHE schemes, the efficiency of FHE remains a challenge. This is due to the significant expansion of data scale, increasing computational load, and fitting calculation errors of nonlinear activation functions.

Our scheme uses PHE to protect the privacy of NLP models and data. Compared to FHE, PHE offers lower computational overhead. While it is less efficient than differential privacy in terms of computational cost, PHE provides high availability, low communication overhead, and is suitable for widely applicable NLP scenarios.

3. Preliminaries

This section introduces algorithmic preliminaries and the notation, as summarized in Table 1.

3.1. Word vectors representation

Word vectors representation provides a method for converting words into continuous numeric vectors, which can be used as input to the machine learning models. Word vectors have been widely used in NLP applications in recent years. Early word representation mainly focused on the study of word distinction and frequency, such as one-hot encode, bag of words model and the N-gram method. However, these methods either have large word vectors dimensions and sparse data, or they do not consider semantic information between word order and context. With the rapid development of neural networks, the machine builds a language model by training the context in the text, thus indirectly obtaining the word vectors representation. As a classical neural network language model, **word2vec** can not only learn the expression of words, but also pay more attention to the representation of words in the context [26]. In this method, the word vectors are obtained by constructing a Hoffman tree and calculating the weighted path of the word in the tree. **Transformer** model based on adaptive mechanisms performs well in extracting text semantic features [27].

Arora et al. [28] investigated the performance of three word vectors from two aspects: data size and language characteristics. The three word vectors include **BERT** (mainly structured as **Transformer** model), **GloVe**, and **Random** word vectors. Language characteristics include complexity of sentence structure, ambiguity of words and unlisted words. The results show that: (1). The larger the data size, the closer the three word vectors performed, and **BERT** performs better on small datasets; (2). **BERT** word vectors performs better in sentences with high complexity and more ambiguous words, while **GloVe** performs better for sentences in more unlisted words; (3). **Random** word vectors and **GloVe** have some advantages when training time and computational resources are taken into account. Feusner et al. [29] used **GloVe** model to transform the obtained user text into a linear space of the word vectors, generate a co-occurrence matrix, and complete word clustering through principal component analysis. Rustam et al. [30] first analyzed various widely used machine learning algorithms such as random forests, and then proposed a machine learning model for predicting employee job satisfaction. Finally, they evaluated various text feature extraction methods such as **GloVe** model. Therefore, the **GloVe** model has some advantages in some application scenarios.

In this paper, we focus on the word vectors representation global vector model **GloVe** that proposed by Pennington et al. [7]. **GloVe** model integrates the advantages of previous methods, not only paying attention to the local context information between words, but also considering the global features. It is mainly reflected in the calculation of co-occurrence matrix. A co-occurrence matrix calculates the probability of each word appearing in the context of a word within a given window size. The probability in the co-occurrence matrix can indicate the correlation between words. Theoretically, the higher the probability of co-occurrence of similar words, the higher the correlation between words. Interested readers can refer to the original paper for more details [7].

3.2. Paillier cryptosystem

Homomorphic encryption allows certain operations to be performed on the encrypted ciphertext, and the decryption of the generating encryption results matches the results of the same operations performed on the plaintext. We are using the Paillier cryptosystem in our scheme, which satisfies homomorphism for individual addition operations and has semantic security [31]. The Paillier cryptosystem is briefly described as follows:

(1) *Key Generation*: Choose distinct large primes p and q, such that $\gcd(p, q-1) = 1$ and $\gcd(p-1, q) = 1$, and compute $n = p \times q$ and $\lambda(n) = \mathrm{lcm}(p-1, q-1)$. Choose a semi-random base $g \in \mathbb{Z}^*_{n^2}$, such that n divides $\mathrm{ord}(g)$, where $\mathrm{ord}(g)$ denotes the order of g in the cyclic group $\mathbb{Z}^*_{n^2}$. The public key pk and the private key sk are (n, g) and $\lambda(n)$, respectively.

(2) *Encryption*: Let $m \in \mathbb{Z}_n$ be a plaintext, choose a random $u \in \mathbb{Z}^*_{n^2}$. Then the ciphertext c is computed as follow:

$$c \equiv g^m \times u^n \pmod{n^2}. \tag{1}$$

For each plaintext to be encrypted, select a different random number u to ensure security.

(3) *Decryption*: For the ciphertext c, the original plaintext m can be recovered using private key $\lambda(n)$ as follow:

$$m \equiv \frac{L(c^{\lambda(n)} \pmod{n^2})}{L(g^{\lambda(n)} \pmod{n^2})} \pmod{n}, \tag{2}$$

where function $L(z) = (z-1)/n$.

(4) *Homomorphic Properties*: Given $m_1, m_2 \in \mathbb{Z}_n$, let k be an integer. The Paillier cryptosystem satisfies the following properties:

- Additively Homomorphic: $[\![m_1 + m_2]\!] = [\![m_1]\!] \times [\![m_2]\!]$
- Multiplicative Homomorphic by a Constant: $[\![k \times m]\!] = [\![m]\!]^k$
- Additively Homomorphic by a Constant: $[\![k + m]\!] = [\![m]\!] \times g^k$

3.3. Data representation

In the training process, the word vectors are represented by real number vectors. There are two methods to represent real numbers: floating-point and fixed-point. The floating-point representation offers the advantage of large range. However, it is difficult to perform arithmetic operations on floating-point numbers in a data-agnostic manner. More importantly, the message space of the Paillier cryptosystem is positive integers, i.e., $\mathbb{Z}_n$. Thus, using fixed-point representation is more suitable [32].

Given a real number, its fixed-point representation is given by the following equation:

$$\widetilde{a} = \lfloor a \cdot 2^p \rfloor ,$$

where the exponent p is fixed, and $\lfloor \cdot \rfloor$ is a round down function.

The fraction length of bits p can be selected as the system parameter, or p can be chosen according to the accuracy required by the algorithm. In addition, this fixed-point data type will produce rounding errors that are inversely proportional to p. It means that choose a larger p can obtain higher system accuracy. In order to represent negative numbers, we will use the 2's complement representation. If $\widetilde{a}$ is a negative number, and α is a fixed-point data type, then the corresponding complement can be represented $\widetilde{a} + 2^{\alpha}$.

Using the above methods, we can simplify all arithmetic operations on real numbers to arithmetic on integers, in particular:

- Addition/Subtraction: $\widetilde{a \pm b} = \widetilde{a} \pm \widetilde{b}$
- Multiplication: $\widetilde{a \cdot b} = \left\lfloor \widetilde{a} \cdot \widetilde{b} / 2^p \right\rfloor$
- Division: $\widetilde{a/b} = \left\lfloor \widetilde{a} \cdot 2^p / \widetilde{b} \right\rfloor$

For example, to perform division between fixed points $\widetilde{a}$ and $\widetilde{b}$, we just shift p bits to the left (equivalent to multiplying by 2^p) and perform integer division with $\widetilde{b}$, and thus obtain the fixed-point representation of a/b.

3.4. Secure multiplication protocol

In order to achieve the multiplication of two plaintexts on ciphertext, we require two servers to jointly run the secure multiplication protocol [33]. This protocol is based on the following property:

$$a \cdot b = (a + r_1) \cdot (b + r_2) - a \cdot r_2 - b \cdot r_1 - r_1 \cdot r_2. \tag{3}$$

In our scheme, the public cloud CS_1 has two ciphertext $[\![\widetilde{a}]\!]$ and $[\![\widetilde{b}]\!]$, and the private cloud CS_2 has the private key sk generated by the Paillier cryptosystem. Without losing generality, set the fixed factor of fixed-point data type to 2^p. The detailed steps of the secure multiplication protocol are as follows:

(1) CS_1 chooses two random positive integers r_1 and r_2, and compute $[\![\widetilde{a} + r_1]\!] = [\![\widetilde{a}]\!] \cdot [\![r_1]\!]$, $[\![\widetilde{b} + r_2]\!] = [\![\widetilde{b}]\!] \cdot [\![r_2]\!]$, then sends $[\![\widetilde{a} + r_1]\!]$ and $[\![\widetilde{b} + r_2]\!]$ to CS_2;
(2) CS_2 decrypts the received two ciphertext using the private key sk and computes intermediate result $z = (\widetilde{a} + r_1) \cdot (\widetilde{b} + r_2)$, then re-encrypts the result to get $[\![z]\!]$ and returns to CS_1;
(3) CS_1 computes $[\![\widetilde{a} \cdot \widetilde{b}]\!] = [\![z]\!] \cdot [\![\widetilde{a}]\!]^{-r_2} \cdot [\![\widetilde{b}]\!]^{-r_1} \cdot [\![r_1 \cdot r_2]\!]^{-1}$ according to homomorphic properties,

where $[\![x]\!]^{-1}$ represents the modular inverse of $[\![x]\!]$. According to the homomorphic property of the Paillier cryptosystem, we have $[\![\widetilde{a - b}]\!] = [\![\widetilde{a}]\!] \cdot [\![\widetilde{b}]\!]^{-1}$.

According to the above scheme, to compute the inner product of two encrypted vectors containing only addition and multiplication, we can easily make use of the secure multiplication protocol and the homomorphic property of the Paillier cryptosystem. For the convenience of representation, we use $\otimes$ to represent the multiplication of two ciphertexts or the inner product of two encrypted vectors, namely $[\![\widetilde{a \cdot b}]\!] = [\![\widetilde{a}]\!] \otimes [\![\widetilde{b}]\!]$.

3.5. Logarithm function

In the cost function of **GloVe** model, the logarithm function $f(x) = \ln(x)$ needs to be computed. But the Paillier cryptosystem does not directly support the homomorphic logarithm function. However, we can compute the logarithm function approximately. The theoretical analysis and experimental verification show that the approximate effect can be almost ignored when compared with the results in the plaintext.

In our scheme, we use Taylor series for logarithmic function. Specifically, considering the convenience of $\ln(x+1)$ does not affect the result, so we expand $\ln(x+1)$ using the following equation:

$$\begin{aligned}\ln(x+1) &= \int \frac{1}{1+x}\, dx \\ &= \sum_{n=0}^{\infty} -\frac{1}{n+1}(-x)^{n+1} \\ &= \sum_{n=1}^{\infty} -\frac{1}{n}(-x)^{n} \\ &= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots\end{aligned}$$

Therefore, we can implement secure computation of logarithm function by implementing the homomorphic properties and secure multiplication protocol on ciphertext. In addition, the number of terms in the expansion can be flexibly selected according to the accuracy requirements and efficiency.

4. Security statement

In this section, the system model and threat model are described in detail, and the design goals for this paper are determined.

4.1. System model

This paper studies the representation of training word vectors on ciphertext in cloud computing environment. As shown in Fig. 1, our scheme consists of three roles, including multiple participating users, the public cloud CS_1 and the private cloud CS_2.

There are n users, and u_i represents ith user for $i = 1, \ldots, n$. Each user has a private file f_i and hopes to learn the word vectors with all other users without privacy disclosure. Users do not need the results of word vectors learned in the future. Intuitively, the private cloud generates public and private keys and publishes the public key to each user and the public cloud. Users encrypt their private data into ciphertext using the public key, and then outsources it to the public cloud. After receiving the ciphertext data, the public cloud trains all the data to generate high-quality word vectors and models, which can be used in various NLP scenarios.

In addition, public and private clouds together make up the hybrid cloud, which has both the computing power of the public cloud and the security of the private cloud [34]. In practice, the public cloud is usually a large enterprise like Google, while the private cloud is generally a government agency under supervision. Considering the reputation impact, it is impossible for them to collude with each other. In our scheme, the private cloud is responsible for initializing the system. The main work is to initialize parameters, provide encrypted keys for all participants, and jointly complete secure interactive computing with the public cloud. The public cloud constructs a word vectors model from the ciphertext data collected by users for training, and collaboratively runs the training protocol with the private cloud. This is similar to executing a learning algorithm on plaintext data. Finally, high-quality word vectors are obtained. Considering the potential commercial value of word vectors for subsequent NLP tasks, the public cloud is unwilling to disclose the model information to others.

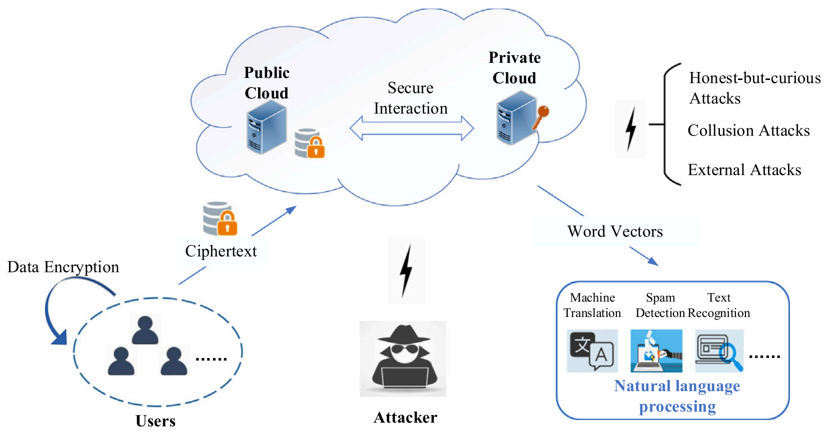

Fig. 1. Secure word vectors learning model in cloud computing.

4.2. Threat model

Based on some privacy threats in real-world, in this paper, we consider the following three types of threat attacks:

(1) Honest-but-curious: All participants honestly perform each step of the algorithms. However all participants are curious and untrustworthy, interested in privacy data and computation results, and try to understand and infer more useful information during the word vectors learning process [35].
(2) Collusion: The public cloud colludes with some users to obtain the privacy data of other users, but there is no collusion between public and private clouds, which is reasonable in reality.
(3) External: In addition to the participants, we suppose there exists an external adversary who can eavesdrop on the communication channel among the user and the two cloud servers to obtain the transmission data and attempt to obtain the user's private data, model information and word vectors result.

There are two main potential adversaries in our scheme: the public cloud and an external adversary. The public cloud can provide the high quality service for the users, thus attracts more users to use. However, cloud service providers are profitable organization that provides services in an unsupervised and opaque, and there may be some dishonest behavior. Because the user needs to upload private data to the public cloud, the public cloud will be honest with the pre-designed protocol for computing, but it may be for other purposes to steal the user's private data and computing results. Therefore, our scheme can resist honest-but-curious attacks from the public cloud. Please note that there is no collusion between public and private clouds, and that the private key can only be kept by the private cloud, while only the public key is involved in computing. In addition, there is a malicious external adversary whose behavior is uncontrollable. Specifically, an external adversary would attack the communication channel between the user and the server to obtain the homomorphic encryption ciphertext in transmission, and then decrypt ciphertext to obtain the private data and the results of the computation. Due to our scheme is based on the semantic security of the Paillier homomorphic encryption algorithm, namely indistinguishability under chosen-plaintext attack (i.e. IND-CPA secure). This indicates that our scheme can resist chosen-plaintext attack [36]. Without a private key, it is difficult for an external adversary to obtain any information from the original plaintext by deciphering the ciphertext.

4.3. Research objectives

Based on the above system model and threat model, our research objectives mainly include the following three parts:

(1) Privacy: Considering that all participants are semi-honest in the scheme, the security of all parties should be protected, including users' private data, the model information and word vectors results of the public cloud's computation. Specifically, users' private data should not be disclosed to the hybrid cloud and external adversaries, and word vectors model and results should not be disclosed to another cloud server (the private cloud), users and external adversaries.
(2) Efficiency: Due to the limited computation and communication capabilities, the user side should be lightweight and undertake less computation and communication overhead during the whole word vectors training process.
(3) User offline: In order to optimize the user's experience, users can go offline after sending encrypted data to the cloud server. Obviously, supporting users to be offline has advantages in improving the scalability of the solution.

5. Privacy-preserving word vectors learning scheme

In this section, we describe a privacy-preserving word vectors learning scheme in detail, which adopts the **GloVe** word vectors representation method. The core of our scheme consists of the following four steps.

1. System initialization: CS_2 generates a public key and a private key based on the Paillier cryptosystem, and distributes the public key to other participants, including the user and CS_1.
2. Data outsourcing: Users encrypt the original text data to generate the ciphertext using the received public key and send them to CS_1.
3. **GloVe** word vectors learning: CS_1 runs the word vectors representation scheme based on **GloVe** and users input to generate encrypted word vectors result.
4. Returning the results: (a) CS_1 perturbs the encrypted word vectors and sends them to CS_2, (b) CS_2 decrypts them and returns them to CS_1, (c) CS_1 gets the word vectors after eliminating the disturbance.

5.1. System initialization

CS_2 is responsible for setting the initial environment. Given a security parameter λ, CS_2 randomly generates a public key and private key pair (pk_{CS_2}, sk_{CS_2}) based on the Paillier cryptosystem, and keeps the private key sk_{CS_2} strictly secret. The public key pk_{CS_2} is made available to the other participants. Each user and CS_1 can obtain the public key pk_{CS_2} generated by CS_2. Table 2 summarizes the key distribution of users, CS_1 and CS_2.

Table 2
Key distribution.

Users	CS_1	CS_2
pk_{CS_2}	pk_{CS_2}	(pk_{CS_2}, sk_{CS_2})

5.2. Data outsourcing

Before outsourcing data to the cloud server, data processing and encryption are required to protect users' privacy. In our system model, each user has own private data file, and hopes to collaborate with and have access to all other users' private data files for learning. Intuitively, users can protect their privacy by encrypting sensitive data and then outsourcing the ciphertext to the cloud server. The data file f_i of the user u_i contains a collection of sentences; the words in all sentences are expressed as $(w_1, \ldots, w_{n(f_i)})$. In order to obtain the number of word vectors, the user u_i counts the vocabulary size V (i.e. the number of non-repeating words) for the corpus in data file f_i. In addition, the co-occurrence matrix needs to be calculated for the corpus of all users. Each item represents the number of occurrences of word j in the context of word i, i.e., the co-occurrence value X_{ij}. For each X_{ij} we need to convert the original real value to an integer representing $\widetilde{X_{ij}}$, and then use the Paillier cryptosystem to encrypt it to obtain $[\![\widetilde{X_{ij}}]\!]$. Moreover, because there are some zero elements in the co-occurrence matrix, in order to save storage and computing resources, only non-zero elements need to be encrypted. Finally, each user sends the vocabulary size V_{u_i} and the encrypted co-occurrence matrix $[\![\widetilde{X}]\!]_{u_i}$ to the cloud server CS_1.

5.3. GloVe word vectors learning

GloVe word vectors learning is based on the co-occurrence matrix, where each row is the main word and each column is the context word. The dimension of the word vectors of the context word is reduced to learn the word vectors representation of the main word. In the process of **GloVe** word vectors learning, CS_1 receives the data sent by the user, and then collaborates with CS_2 to generate the new word vectors. The main operations of these two cloud servers can be divided into the following aspects: (1) secure computation of gradients and updating word vectors; (2) secure computation of weight function; and (3) secure comparison.

5.3.1. Secure computation of gradients and updating word vectors

The cost function of the **GloVe** model is given in Eq. (4).

$$J(i,j) = \sum_{i,j=1}^{V} f(X_{ij})(\overrightarrow{w_i}^T \overrightarrow{w_j} + b_i + b_j - \log(X_{ij}))^2. \tag{4}$$

The training goal is to maximize this function, where $\overrightarrow{w_i} \in \mathbb{R}^d$ is the word vector of the main word, $\overrightarrow{w_j} \in \mathbb{R}^d$ is the word vector of the context word, d is the dimension of the vector, b_i and b_j are the bias items of $\overrightarrow{w_i}$ and $\overrightarrow{w_j}$, respectively. Here $f(X_{ij})$ is the weight function [7] and we can take it as

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}.$$

In the co-occurrence matrix X of a training sample, there are $V \times V$ items. The co-occurrence matrix is a symmetric sparse matrix. Since **GloVe** does not compute words with zero co-occurrence, many zero elements do not participate in the training of word vectors. The elements of the upper and lower half angles of the symmetric matrix are V terms, so there are $2V$ co-occurrence values in the input layer, each of these items can be represented by a tuple (main word index, context word index, co-occurrence value). In addition, the number of initialization word vectors is $2V$, and the dimension is d.

Our training goal is to minimize the cost function $J(i,j)$, which is implemented using a stochastic gradient descent algorithm to obtain the trained word vectors at the output level. From the original cost function, we can get the gradient $\overrightarrow{w_i}$, $\overrightarrow{w_j}$, b_i and b_j, and notice that the weight function $f(X_{ij})$ does not depend on any of the parameters.

$$\nabla_{\overrightarrow{w_i}} J(i,j) = \sum_{j=1}^{V} f(X_{ij})(\overrightarrow{w_i}^T \overrightarrow{w_j} + b_i + b_j - \log(X_{ij})) \odot \overrightarrow{w_j}, \tag{5}$$

$$\nabla_{\overrightarrow{w_j}} J(i,j) = \sum_{i=1}^{V} f(X_{ij})(\overrightarrow{w_i}^T \overrightarrow{w_j} + b_i + b_j - \log(X_{ij})) \odot \overrightarrow{w_i}, \tag{6}$$

$$\frac{\partial J(i,j)}{\partial b_i} = \sum_{j=1}^{V} f(X_{ij})(\overrightarrow{w_i}^T \overrightarrow{w_j} + b_i + b_j - \log(X_{ij})), \tag{7}$$

$$\frac{\partial J(i,j)}{\partial b_j} = \sum_{i=1}^{V} f(X_{ij})(\overrightarrow{w_i}^T \overrightarrow{w_j} + b_i + b_j - \log(X_{ij})), \tag{8}$$

where $\nabla_{\overrightarrow{w_i}} J(i,j)$ is the gradient of the cost function of the main word vector $\overrightarrow{w_i}$, $\nabla_{\overrightarrow{w_j}} J(i,j)$ is the gradient of the cost function of the context word vector $\overrightarrow{w_j}$, $\frac{\partial J(i,j)}{\partial b_i}$ is the gradient of the bias term b_i, $\frac{\partial J(i,j)}{\partial b_j}$ is the gradient of the bias term b_j. The operator $\odot$ represents the product of element and vector.

The updated results of the word vector $\overrightarrow{w_i}, i \in 2V$ of the main word and the word vector $\overrightarrow{w_j}, j \in 2V$ of the context word are shown in Eqs. (9) and (10), respectively.

$$\overrightarrow{w_i} = \overrightarrow{w_i} - \eta \nabla_{\overrightarrow{w_i}} J(i,j), \tag{9}$$

$$\overrightarrow{w_j} = \overrightarrow{w_j} - \eta \nabla_{\overrightarrow{w_j}} J(i,j), \tag{10}$$

where η is learning rate. The updated results of the bias items b_i and b_j are shown in Eqs. (11) and (12), respectively.

$$b_i = b_i - \eta \frac{\partial J(i,j)}{\partial b_i}, \tag{11}$$

$$b_j = b_j - \eta \frac{\partial J(i,j)}{\partial b_j}. \tag{12}$$

Algorithm 1 Secure **GloVe** Word Vectors Learning

```
Input: An encrypted training matrix of word–word co-occurrence $[\![\widetilde{X}]\!]$,
    which contains $2V$ encrypted entries $[\![\widetilde{X_{ij}}]\!]$
Output: Updated word vectors $[\![\overrightarrow{w_i}]\!]$, context vectors $[\![\overrightarrow{w_j}]\!]$, bias $[\![\widetilde{b_i}]\!]$
    and $[\![\widetilde{b_j}]\!]$
    $CS_1$:
 1: Initialize word vectors $\overrightarrow{w_i}$ and $\overrightarrow{w_j}$, bias terms $b_i$ and $b_i$
 2: Compute weight function $[\![\widetilde{f(X_{ij})}]\!] = \mathrm{SCWF}([\![\widetilde{X}]\!])$
 3: Compute logarithmic function $[\![\widetilde{l(X_{ij})}]\!] = \ln([\![\widetilde{X_{ij}}]\!])$
 4: Compute $[\![\widetilde{M(i,j)}]\!] = [\![\overrightarrow{w_i}]\!] \otimes [\![\overrightarrow{w_j}]\!] \times [\![b_i]\!] \times [\![b_j]\!] \times [\![\widetilde{l(X_{ij})}]\!]^{-1}$
 5: Compute $[\![\widetilde{J(i,j)}]\!] = [\![\widetilde{f(X_{ij})}]\!] \otimes [\![\widetilde{M(i,j)}]\!] \otimes [\![\widetilde{M(i,j)}]\!]$
 6: for $j = 1, \ldots, V$ do
 7:     Compute $[\![\widetilde{g_{\overrightarrow{w_i}}}]\!] = [\![\widetilde{f(X_{ij})}]\!] \otimes [\![\widetilde{M(i,j)}]\!] \otimes [\![\overrightarrow{w_j}]\!]$
 8:     Compute $[\![\widetilde{g_{b_i}}]\!] = [\![\widetilde{f(X_{ij})}]\!] \otimes [\![\widetilde{M(i,j)}]\!]$
 9:     Update $[\![\overrightarrow{w_i}]\!] = [\![\overrightarrow{w_i}]\!] \times [\![\widetilde{g_{\overrightarrow{w_i}}}]\!]^{-\eta}$
10:     Update $[\![\widetilde{b_i}]\!] = [\![\widetilde{b_i}]\!] \times [\![\widetilde{g_{b_i}}]\!]^{-\eta}$
11: end for
12: for $i = 1, \ldots, V$ do
13:     Compute $[\![\widetilde{g_{\overrightarrow{w_j}}}]\!] = [\![\widetilde{f(X_{ij})}]\!] \otimes [\![\widetilde{M(i,j)}]\!] \otimes [\![\overrightarrow{w_i}]\!]$
14:     Compute $[\![\widetilde{g_{b_j}}]\!] = [\![\widetilde{f(X_{ij})}]\!] \otimes [\![\widetilde{M(i,j)}]\!]$
15:     Update $[\![\overrightarrow{w_j}]\!] = [\![\overrightarrow{w_j}]\!] \times [\![\widetilde{g_{\overrightarrow{w_j}}}]\!]^{-\eta}$
16:     Update $[\![\widetilde{b_j}]\!] = [\![\widetilde{b_j}]\!] \times [\![\widetilde{g_{b_j}}]\!]^{-\eta}$
17: end for
```

Algorithm 1 summarizes the cloud server CS_1 implements the **GloVe** word vectors learning scheme securely. Given an encrypted word vectors co-occurrence matrix $[\![\widetilde{X}]\!]$ for training, CS_1 first computes the weight function values securely using Algorithm 2. For logarithmic functions $\ln([\![\widetilde{X_{ij}}]\!])$ in the ciphertext domain, we use the approximate method designed in Section 3 for securely computation. For the computation of cost function, multiplication operation needs to be carried out

Table 3
Comparison result.

t_1	t_2	C	$t_1 \oplus t_2$
0	0	$a > b$	0
0	1	$a < b$	1
1	0	$a < b$	1
1	1	$a > b$	0

among various items, thus secure multiplication protocol can be used. In line 5, CS_1 computes the cost function $[\![\widetilde{J(i,j)}]\!]$ according to Eqs. (4) and using the secure multiplication protocol. According to Eqs. (5)–(8), the gradients of each parameter are computed securely. Finally, according to Eqs. (9)–(12), the word vector $[\![\vec{w_i}]\!]$ of the main word and the word vector $[\![\vec{w_j}]\!]$ of the context word are updated securely, and the corresponding bias terms $[\![\widetilde{b_i}]\!]$ and $[\![\widetilde{b_j}]\!]$ are also updated securely.

5.3.2. Secure computation of weight function

In Algorithm 2, CS_1 securely computes the weight function. For all co-occurrence values $[\![\widetilde{X_{ij}}]\!]$ in the co-occurrence matrix $[\![\widetilde{X}]\!]$, the security comparison scheme of Algorithm 3 is used to compare the encrypted co-occurrence value $[\![\widetilde{X_{ij}}]\!]$ and the size of the given encryption parameter $[\![x_{\max}]\!]$. If $[\![\widetilde{X_{ij}}]\!] > [\![x_{\max}]\!]$, then it computes the weight function $[\![\widetilde{f(X_{ij})}]\!]$ as in line 6. Since it is necessary to calculate the exponential power of α, we can use the secure multiplication protocol to multiply α for $[\![\widetilde{f(X_{ij})}]\!]$ consecutive times; otherwise, $[\![\widetilde{f(X_{ij})}]\!]$ is computed according to line 8.

Algorithm 2 Securely Compute Weight Function (SCWF)

```
Input: An encrypted training matrix of word–word co-occurrence $[\![\widetilde{X}]\!]$,
    where contains $2V$ encrypted entries $[\![\widetilde{X_{ij}}]\!]$
Output: Encrypted weight function $[\![\widetilde{f(X_{ij})}]\!]$
    $\mathrm{CS}_2$:
 1: Choose $x_{\max}$ and $\alpha$ according to weighting function properties and
    send to $\mathrm{CS}_1$
    $\mathrm{CS}_1$:
 2: for all $[\![\widetilde{X_{ij}}]\!]$ in $[\![\widetilde{X}]\!]$ do
 3:   Compare result $C = \mathrm{SC}([\![\widetilde{X_{ij}}]\!], [\![x_{\max}]\!])$
 4:   if $C \leftarrow 1$ then
 5:     Compute $[\![\widetilde{f(X_{ij})}]\!] = [\![\widetilde{X_{ij}}]\!] \otimes [\![x_{\max}]\!]^{-1}$
 6:     Compute $[\![\widetilde{f(X_{ij})}]\!] = [\![\widetilde{f(X_{ij})}]\!] \otimes \cdots \otimes [\![\widetilde{f(X_{ij})}]\!]$ $\alpha$ times
 7:   else
 8:     Compute $[\![\widetilde{f(X_{ij})}]\!] = [\![1]\!]$
 9:   end if
10: end for
```

5.3.3. Secure comparison

Algorithm 3 is used to compare the size of two integers in the ciphertext domain. Assume $a, b \in (-\theta_1, \theta_1)$, CS_1 randomly selects integers $r_1, r_2, r_3 \in U(0, \theta_2)$, and the interval $U(0, \theta_2)$ is uniformly distributed, $r_3 < r_1, r_2$, and $2\theta_1\theta_2 < n/2$, where n is part of the public key. From lines 3 and 5, if $t_1 = 0$, then $s = r_1(a-b)+r_3$; otherwise $s = r_2(b-a)+r_3$. If $s < n/2$, the computation result is positive, and CS_2 sets $t_2 = 1$; otherwise sets $t_2 = 0$. Then CS_2 sends t_2 to CS_1. CS_1 will judge after receiving t_2. When $t_1 = t_2$, then $a > b$; otherwise, $a < b$. In this process, CS_2 does not know the result of the secure comparison. According to Table 3, the final comparison result is exclusive XOR operation for t_1 and t_2.

5.4. Returning the results

CS_1 generates the encrypted word vectors result according to Algorithm 1. In order to get the word vectors under the plaintext domain, the ciphertext needs to be decrypted. Only CS_2 has the private key sk_{CS_2}, thus the ciphertext of the word vectors needs to be sent to CS_2 for decryption. In order to protect the privacy of the word vectors, CS_1 needs to disturb the word vectors ciphertext $[\![\vec{w_i}]\!]$ and $[\![\vec{w_j}]\!]$. CS_1 generates random numbers r_x and r_y, computes $[\![\vec{w_i}]\!] \times [\![\vec{r_x}]\!] = [\![\vec{w_i} + \vec{r_x}]\!]$ and $[\![\vec{w_j}]\!] \times [\![\vec{r_y}]\!] = [\![\vec{w_j} + \vec{r_y}]\!]$, and sends the two ciphertext to CS_2. CS_2 decrypts the two ciphertexts using the private key to get $\vec{w_i} + \vec{r_x}$ and $\vec{w_j} + \vec{r_y}$, and sends the decrypted results to CS_1. CS_1 then subtracts the random numbers r_x and r_y to eliminate the disturbance, and finally obtains the word vectors $\vec{w_i}$ and $\vec{w_j}$.

Algorithm 3 Secure Comparison(SC)

```
Input: Two encrypted data $[\![a]\!]$ and $[\![b]\!]$, $a, b \in (-\theta_1, \theta_1)$
Output: Comparison result $C$
    $\mathrm{CS}_1$:
 1: Randomly choose integers $r_1, r_2, r_3 \in U(0, \theta_2)$ and $r_3 < r_1, r_2$,
    randomly choose $t_1$ from $\{0, 1\}$
 2: if $t_1 \leftarrow 0$ then
 3:   Compute $[\![s]\!] = [\![r_3]\!] \times ([\![a]\!] \times [\![b]\!]^{-1})^{r_1}$
 4: else
 5:   Compute $[\![s]\!] = [\![r_3]\!] \times ([\![a]\!]^{-1} \times [\![b]\!])^{r_2}$
 6: end if
 7: $CS_1 \rightarrow CS_2 : [\![s]\!]$
    $\mathrm{CS}_2$:
 8: Decrypt $[\![s]\!]$ to obtain $s$
 9: if $s < n/2$ then
10:   $t_2 \leftarrow 1$
11: else
12:   $t_2 \leftarrow 0$
13: end if
14: $CS_2 \rightarrow CS_1 : t_2$
    $\mathrm{CS}_1$:
15: if $t_1 = t_2$ then
16:   $C : a > b$
17: else
18:   $C : a < b$
19: end if
```

6. Theoretical analysis

In this section, we present a theoretical analysis on our scheme from the perspectives of security, complexity and performance.

6.1. Security analysis

As mentioned before, the goal of our scheme is to ensure that the public cloud and the private cloud cannot obtain the content of the data provided by users, including the original word and its related word vectors, the model information of the public cloud computing and the results of the generated word vectors are not leaked. We analyze security from two aspects: the users and the hybrid cloud (i.e., public and private clouds).

6.1.1. Against external attacks

The external attack mainly occurs in communication, mainly during the process that the client sends the ciphertext of the private data to the public cloud. The public cloud obtains the encrypted data in the data outsourcing phase, including the vocabulary size of the corpus and the encrypted co-occurrence matrix. Vocabulary is a statistic of the number of non-repeated words in the corpus, in order to determine the number of word vectors, it will not reveal the privacy of the words in the original corpus. Furthermore, the co-occurrence matrix can reflect the privacy of the original corpus words, thus the Paillier cryptosystem is used to encrypt them. Due to the model information and the word vector results are also encrypted, it is difficult for an external adversary to obtain the real information. The Paillier cryptosystem is based on the problem of compound residuals, which is semantically secure for chosen-plaintext attack [31]. Therefore, even if an external adversary obtains the ciphertext, it cannot have access to the original data.

6.1.2. Against the hybrid cloud attacks

In the process of privacy-preserving word vectors learning through the public cloud, Algorithms 2 and 3 exchange information and are designed based on Algorithm 1. Thus we first analyze how the basic Algorithms 3 and 2 protect privacy.

- Security of Algorithm 3: CS_1 randomly generates three integers r_1, r_2, r_3, and then uses the homomorphic property to compute $[\![s]\!]$ according to the randomly selected t_1, and sends it to CS_2. After receiving $[\![s]\!]$, CS_2 decrypts it to obtain s, and assigns 0 or 1 to t_2 according to the size values of s and n. In this process, the encrypted a and b are input, thus CS_1 does not know the true values of these two data. Since r_1, r_2, r_3 is randomly generated, the true values of a and b cannot be obtained after decryption. Moreover, the comparison result is equivalent to $t_1 \oplus t_2$. CS_1 and CS_2 cannot obtain any information about a and b. Because CS_2 does not know the value of t_1, CS_2 cannot obtain the comparison result. Only CS_1 knows the comparison result. Notably, even if CS_1 knows the results of the comparison, it would be difficult to obtain the privacy information of the original data or the model. In Algorithm 2, the comparison is between $[\![x_{\max}]\!]$ and $[\![\widetilde{X_{ij}}]\!]$, because the two ciphertext are encrypted by using Paillier algorithm, it is difficult to get the original information of the data, only know the information after comparing the two data. Therefore, Algorithm 3 can protect private data from the cloud server attacks and protect the intermediate results of CS_1 computations.
- Security of Algorithm 2: CS_2 first selects the weight function parameters $[\![x_{\max}]\!]$ and α, and sends to CS_1, CS_1 runs the security comparison algorithm to get the comparison results. According to the comparison results, CS_1 computes the weight function $[\![\widetilde{f(X_{ij})}]\!]$. If the comparison result is 0, the weight function value can be obtained as 1. CS_1 only needs to encrypt 1, the ciphertext involved in the subsequent and other ciphertext computation. Since these are computed intermediate values, the impact on privacy in the entire algorithm can be ignored. In this step, all data except for the parameters that have been exposed is calculated with encryption. As a result, Algorithm 2 is protected against attacks from the cloud server.
- Security of Algorithm 1: CS_1 first uses Algorithm 2 to compute the weight function, because Algorithm 2 is secure and contains Algorithm 3. CS_1 only knows the encrypted inputs, parameters, and some intermediate results, and cannot learn anything useful information from Algorithms 2 and 3. In addition, Algorithm 1 uses the properties of homomorphic encryption and secure multiplication protocol to eventually generate word vectors result. In the process, no privacy information will be disclosed.

In the result return phase, CS_1 obtains the ciphertext of the word vectors. Without the private key, it is impossible to decrypt the ciphertext. CS_1 sends the ciphertext to CS_2 and generates a random number plus disturbance. CS_2 decrypts the ciphertext to obtain the disturbed plaintext, thus CS_2 does not know the original plaintext information. Then the disturbed plaintext is sent to CS_1, and CS_1 gets the word vectors plaintext by eliminating the disturbance. In this process, CS_2 cannot get the original information of the word vectors, which protects the privacy of the word vectors.

6.1.3. Against collusion attacks

According to Algorithm 1, the honest-and-curious public cloud can infer the user's private data using ciphertext input and the results of a trained word vectors. However, since there is no collusion between the public cloud and the private cloud, CS_1 or CS_2 cannot know the intermediate results such as $[\![\widetilde{f(X_{ij})}]\!]$ at the same time, which also guarantees the security of Algorithm 1. Furthermore, even if CS_1 or CS_2 colludes with some users to obtain the word vectors result, useful information cannot be known from Algorithm 1. Because CS_2 cannot access Algorithm 1, CS_1 and CS_2 cannot learn any private input information by combining Algorithms 2 and 3. From the above analysis, we can conclude that even if the cloud server obtain the help of users, it cannot obtain the original privacy data. Therefore, our scheme can resist collusion attacks.

6.1.4. Data privacy

The objectives of this paper is to protect the privacy of the original word data, model information and the final word vector results. According to statement in Section 4.2, the risk of privacy data leakage comes from the hybrid cloud and an external adversary. With the support of Theorem 1, our scheme can achieve data privacy. First, we introduce two definitions of semantic security [36].

Definition 1 (*Semantic Security, IND-CPA*). An encryption scheme $\varepsilon = (KeyGen, Enc, Dec)$ is indistinguishable under IND-CPA secure. If there is a polynomial time adversary $\boldsymbol{A}$, then within the polynomial time n, there is a negligible function $neg()$ that satisfies

$$\Pr[\boldsymbol{A} \text{ is correct in IND-CPA experiment}] \le \frac{1}{2} + neg(n),$$

where the IND-CPA experiment included an adversary $\boldsymbol{A}$ and a challenger $\boldsymbol{C}$. The process of experiment is as follows: $\boldsymbol{A}$ sends a pair of equal-length messages m_0 and m_1 to $\boldsymbol{C}$. $\boldsymbol{C}$ randomly selects a number b from $\{0,1\}$, generates the key pair $(sk, pk) \leftarrow KeyGen(\lambda)$ using the security parameter λ, and encrypts message m_b with pk to obtain ciphertext $Enc(m_b) \overset{pk}{\longleftarrow} m_b$. $\boldsymbol{A}$ can access the ciphertext and guess the output b', if $b' = b$, then $\boldsymbol{A}$ guesses correctly. The probability of this guess being correct is no more than $\frac{1}{2}$.

Definition 2 (*Data Privacy*). In a scheme that achieves data privacy through outsourcing policies, if there are all adversaries who can access the polynomial time of the ciphertext data and the corresponding public key, they cannot obtain the privacy information of the original plaintext data.

Theorem 1. *According to Definitions 1 and 2, our scheme can guarantee the privacy security of inputs and outputs data.*

Proof. Before outsourcing phase, the user encrypts the co-occurrence matrix using the Paillier cryptosystem. Due to Paillier cryptosystem is a semantically secure algorithm, even if CS_1 can access the ciphertext matrix $[\![\widetilde{X}]\!]$, it cannot obtain the information of X without the private key sk_{CS_2}. In addition, CS_2 with the private key sk_{CS_2} at this stage, has no access to the ciphertext data.

Throughout the entire outsourcing computation phase, neither CS_1 nor CS_2 can access the plaintext information of X_{ij}. According to Algorithm 2, we need to compare the sizes of $[\![x_{\max}]\!]$ and $[\![\widetilde{X_{ij}}]\!]$, which can be compared using the designed Algorithm 3 designed without revealing privacy information to CS_1 and CS_2. For CS_1, it can obtain results of the comparison, but does not know the details of the original data privacy information. Although CS_2 has private key sk_{CS_2}, it does not access the original ciphertext data in algorithms and can only obtain the intermediate results computed by CS_1. Furthermore, in Algorithm 1, it is almost impossible to obtain information about the model without any additional information. Therefore, without knowledge of X_{ij}, anyone cannot learn useful information about word vectors $\overrightarrow{w}_i$ and $\overrightarrow{w}_j$ and bias terms b_i and b_j from computing the cost function $[\![\widetilde{J(i,j)}]\!]$. In the result return stage, CS_2 obtains disturbed ciphertext, which is computationally indistinguishable from ordinary ciphertext. Even if CS_2 has private key sk_{CS_2} that can decrypt the ciphertext, it still obtains meaningless disturbed data.

According to the above analysis, our scheme can guarantee the privacy of co-occurrence matrix X, word vectors and model information. □

6.2. Complexity analysis

Our complexity analysis mainly focuses on the following three phases: (1) Outsourcing the data; (2) Learning the **GloVe** word vectors; and (3) Returning the results.

In the phase of data outsourcing, using co-occurrence matrix as training samples, the number of word vectors to be updated is $2V$, i.e., the number of co-occurrence values. When encrypting the co-occurrence value in the co-occurrence matrix each time, the number of times you need to use the Paillier cryptosystem depends on the dimension d of the word vectors. Therefore, the computational complexity of generating the ciphertext from each user's training sample is proportional to $2V \cdot d$, the communication complexity is $2V \cdot d \cdot \mathrm{len}(l)$ bits. Here, $\mathrm{len}(l)$ is length of the ciphertext encrypted using the Paillier cryptosystem.

In the phase of **GloVe** word vectors learning, in order to facilitate understanding, we first analyze the complexity of the sub-algorithm, and then analyze the complexity of the main algorithm. For Algorithm 3, $2V \cdot d$ times homomorphic operations are required to transmit $2V \cdot d \cdot \mathrm{len}(l)$ bits. In Algorithm 2, we need to run Algorithm 3, and perform $2V \cdot d \cdot \alpha$ times homomorphic operations without communication overhead. Therefore, the computational cost of Algorithm 2 is $2V \cdot d \cdot (\alpha+1)$ and the communication cost is $2V \cdot d \cdot \mathrm{len}(l)$ bits. In Algorithm 1, Algorithm 2 needs to be used to compute the weight function, and $2V \cdot d$ times homomorphic operations and $2V \cdot d$ times approximate logarithmic function need to be performed, without communication. Therefore, the computation cost of Algorithm 1 is $2V \cdot d \cdot (\alpha+3)$, and the communication cost is $2V \cdot d \cdot \mathrm{len}(l)$ bits.

In the phase of result return, the public cloud CS_1 needs to disturb the word vectors ciphertext and perform $2V \cdot d$ times homomorphic computations. Then the disturbed ciphertext is sent to CS_2, which decrypts the ciphertext and returns the result to CS_1. The communication cost is $4V \cdot d \cdot \mathrm{len}(l)$ bits. The user does not need to participate in this process, thus the user has no overhead.

6.3. Performance analysis

This section discusses some of the accuracy losses in our scheme. Since there are real numbers in the data, it needs to be scaled to an integer to participate in the computation. For addition or subtraction, we scale the real number to 2^l. The output for the multiplication operation is 2^{2l}, we also scale it to 2^l. In this process, the absolute error of $\omega \cdot 2^l$ is introduced, where $\omega \in [0,1]$. When calculating the inner product of two d dimension vectors, the error is $d \cdot 2^l$.

Next, we discuss the error range of the nth-order Taylor expansion of the logarithmic function $\ln(1+x)$. Since the error in the expansion is in the remainder, we only consider the Lagrange remainder. We take $f(x) = \ln(1+x)$, convergence domain is $(-1,1]$, let $x = \frac{1}{5}$. The remainder of Taylor's equation is

$$r_n = \frac{f^{(n)}(\xi)}{n!} x^n (0 < |\xi| < |x|),$$

where the n-order derivative of the logarithmic function is

$$f^{(n)}(x) = \frac{(-1)^{n+1}(n-1)!}{(1+x)^n},$$

Table 4
The basic characteristics of datasets.

Dataset	Storage costs (MB)	Training words	Vocabulary size
D1	7.98	413,906	9,238
D2	15.9	7,487,726	31,167
D3	20.8	15,603,147	75,208
D4	80.1	50,041,812	88,337

we can obtain the result of $|r_n|$ and scale it. Thus we can know

$$|r_n| = \frac{x^n}{n(1+\xi)^n} < \frac{x^n}{n} = \frac{1}{5^n n} < 10^{-3}.$$

At this point, the error margin has reached the level of 10^{-3}. By using the above equation, for example, we need to compute the error boundary of the third-order Taylor expansion of the logarithmic function in the convergence domain of $(-1,1]$. When $n = 3$, it is not difficult to find that the possible maximum error is $\frac{x^n}{n} = \frac{1}{5^n n} \approx 0.00267$.

7. Experimental evaluation

In this section, we conducted experiments on real-world datasets of different scale to evaluate the performance of our scheme.

7.1. Setup

The experiment was implemented on a machine with an Intel Core i5-12400F 2.50GHZ CPU and 16 GB of RAM, and installed with Windows 11 64-bit version. The public cloud and the private cloud were equipped with on two Intel Xeon machines E5-2660 V3 2.60 GHz CPUs and 16 GB of RAM, respectively, and installed with Ubuntu server 18.04 64-bit version.

We use *scikit-learn* machine learning library to import processed data, and use high-precision arithmetic operation library *gmpy2* to implement the Paillier cryptosystem for encryption. Considering the security and efficiency, we set the key size of the Paillier cryptosystem to 1024 bits. In addition, in the training phase, the default number of iterations is used, and the initialization word vectors range is a uniform random value in the range $[-0.5, 0.5]$.

7.2. Datasets and comparison algorithm

We chose a real-world language model benchmark that contains about a billion words of pre-processed text, available from the Google Code Archive.[1] We divided the dataset into four parts: 1/4, 2/4, 3/4 of the documents in the dataset and all the documents, namely, D1, D2, D3 and D4, and use these documents as a training corpus. The number of words trained in each corpus varies from hundreds of thousands to tens of millions. Table 4 summarizes some of the basic characteristics of the dataset.

We discussed some existing privacy-preserving NLP methods in Section 2. As we explained, Wang et al. [18] constructed a privacy-preserving **word2vec** word vectors learning method that uses homomorphic encryption to protect the user's privacy data. Specifically, we compare the four schemes proposed by Wang et al. namely CBOW_HS, CBOW_NEG, Skip-gram_HS, Skip-gram_NEG. Qu et al. [16] utilized **BERT** model to train word vectors and used local differential privacy (LDP) to protect single-token embedding. They achieved differential privacy by adding an n-dimensional random noise density distribution $p(N) \propto \exp(-\eta \|N\|)$ to select Euclidean distances. The privacy mechanism can be expressed as $P(x) = x + N$, where x represents the embedding vector. In our implementation, we use the same sentence embedding mechanism, noise $N \in \mathbb{R}^n$ is sampled from (r, p), where r

[1] http://code.google.com/p/word2vec/.

Table 5
The default parameters of our scheme.

Windows size	Word vector dimensions	Learning rate
10	100	0.05
Fraction length	Number of terms	
24	3	

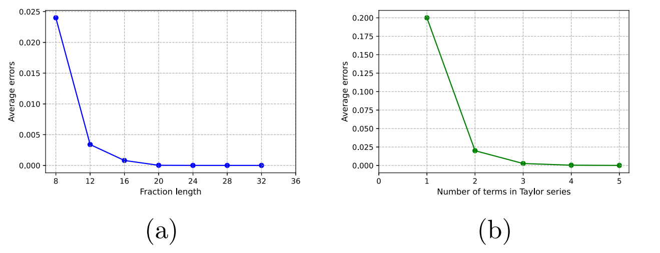

Fig. 2. The average error of the parameters. (a) The average error of fraction length. (b) The average error of the number of terms in the Taylor series.

is the distance to the origin and p is a point in $\mathbb{B}^n$ (the unit hypersphere in $\mathbb{R}^n$). r is sampled from Gamma distribution $\Gamma(n, \frac{1}{\eta})$, and p is uniformly sampled in $\mathbb{B}^n$. η is a privacy parameter. We adopt the BERT pre-training model "bert-large-uncased" to train and set different privacy parameters η to observe the experimental results. We compare our privacy-preserving word vectors learning scheme against collusion attack with their scheme.

7.3. Accuracy

In our scheme, there are two main factors that affect the accuracy of the structure. The use of fixed-point data type to represent real numbers introduces rounding errors and the use of Taylor series for logarithmic function introduces approximation errors.

For these two cases, we designed two experiments to test the error rate, namely, the fraction length and the number of Taylor expansion terms. The parameters we use are as follows: Windows size: 10; Word vector dimensions: 100; Learning rate: 0.05. Through a large number of experiments on different data, we get the experimental results in Fig. 2. Fig. 2(a) shows the relationship between the length of the fraction part of the fixed-point data type and the average error generated by the training result, i.e., the error generated decreases gradually with the increase of the length of the fraction part. Similarly, Fig. 2(b) shows the relationship between the number of terms in the Taylor expansion and the average error. Therefore, these two experimental result figures can provide guidance for parameter configuration of the scheme and try to achieve more ideal results.

7.4. Efficiency

According to the above accuracy analysis, by balancing with the average error rate, the default parameters we used in the experiment are shown in Table 5. The overall efficiency of the experiment is analyzed from the perspective of time complexity and communication cost by using the above four datasets of different scales.

Fig. 3 shows the time to run the original **GloVe** algorithm without privacy protection, including computing the co-occurrence matrix time and computing the word vectors time. Since all operations are performed in the plaintext, there is no cost of communication between the user and the cloud server. In our privacy-preserving scheme, it can be divided into the efficiency of the user side and the efficiency of the cloud server side.

First, we test the computing and communication overhead of the user side. We observe the results of the experiment in terms of the changes in the dataset. As shown in Fig. 4, the time spent on the user side encryption co-occurrence matrix and the communication overhead with the cloud server increases as the number of training words increases. For the co-occurrence matrix of dataset D4, it takes about 10 h to complete the encryption operation and about 2.43 GB to transmit the ciphertext data, both of which are actually acceptable to the user. In addition, in our secure word vector learning process, the user uploads the ciphertext to the cloud server, and does not need to participate in subsequent computing operations, so the user does not incur communication costs in the training phase.

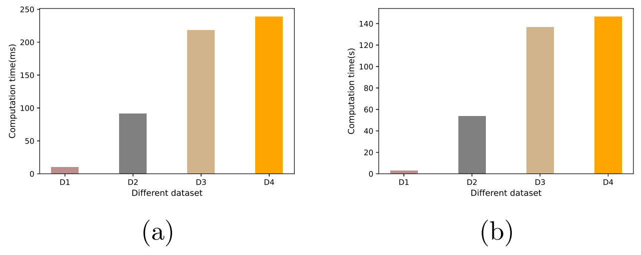

Fig. 3. Efficiency evaluation of the original **GloVe**. (a) The computation time of the co-occurrence matrix. (b) The computation time of the word vectors.

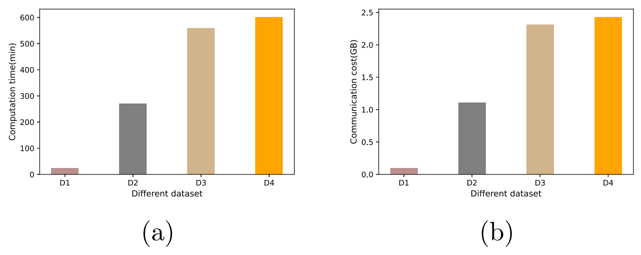

Fig. 4. Efficiency evaluation of user side. (a) The encryption time of the co-occurrence matrix. (b) The communication cost of ciphertext.

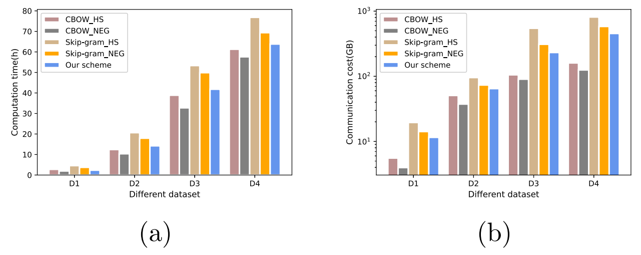

Fig. 5. Efficiency evaluation of the cloud server side. (a) The computation time of the word vectors. (b) The communication cost of ciphertext.

Second, we evaluate the computation and communication costs on the cloud server side (i.e., public and private clouds). As shown in Fig. 5, for each dataset, we compare our scheme with the comparison scheme [18]. As the size of dataset increases, the computational and communication overhead of our scheme and the comparison scheme increases linearly. In our scheme, it has some advantages over Skip-gram_HS and Skip-gram_NEG methods. Therefore, the experimental results show that the proposed privacy word vectors training scheme is efficient and lightweight for users, because a large amount of computation and communication costs are transferred to the cloud server.

Third, for Qu et al.'s noise-based LDP-BERT scheme, we test the computation cost at different levels of noise parameters η (smaller represents larger noise). As shown in Fig. 6, with the size of the dataset increases, the time of adding noise and word vectors computation also increases. With the increases of η, the added noise also decreases, and the computation time decreases to a certain extent. However, the corresponding level of privacy-preserving will also decrease. Therefore,

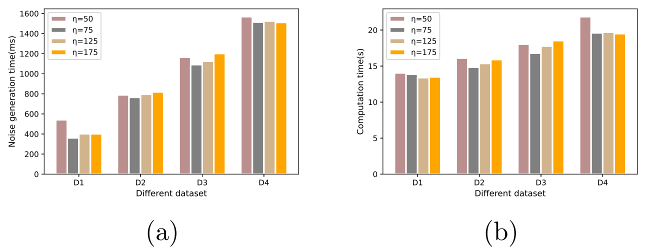

Fig. 6. Efficiency evaluation of LDP-BERT word vectors scheme. (a) The generation time of noise. (b) The computation time of the word vectors.

Table 6
Comparison of word vectors accuracy.

	Dataset	Semantic (%)	Syntactic (%)	Total (%)
GloVe	D1	46.63	39.17	43.62
	D2	51.47	41.84	47.53
	D3	58.94	42.31	50.04
	D4	62.18	48.65	54.75
Our scheme	D1	46.02	38.42	42.93
	D2	51.14	40.75	46.97
	D3	58.52	41.84	49.83
	D4	61.87	48.36	54.37
Relative changing percentage	D1	−0.61	−0.75	−0.69
	D2	−0.33	−1.09	−0.56
	D3	−0.42	−0.47	−0.21
	D4	−0.31	−0.29	−0.38

compared to our scheme, Qu et al.'s scheme has certain advantages in local computing overhead. However, our solution is outsourced to cloud servers for computing. Except for users who need to undertake the cost of encryption operations, all other computing costs are undertook by cloud servers, which is also lightweight for users.

7.5. Effectiveness

To verify the effectiveness of the word vectors, we run our privacy-preserving training scheme and the original **GloVe** training scheme proposed by Pennington et al. [7] on four datasets according to the default parameter settings given in Table 5, and compare the validity of the word vectors. To this end, we use word analogy to measure the quality of word vectors. Word analogy can be explained as a kind of problem, i.e., 'a to b, c to_?'. The dataset we used contains 19,544 such problems and is divided into two major categories, namely semantic problems and syntactic problems. Semantic problems are often analogies of people and places, such as 'Fingers to the palm as toes to_?'. Syntactic problems are usually analogies of verb tenses or adjective forms, such as 'Sing to singing as swim to_?'. Therefore, the trained word vectors should be able to correctly answer these questions and uniquely identify the missing items in the questions. Only an accurate answer can be considered as a correct match. We use cosine similarity to find the word closest to the problem.

Table 6 summarizes the comparison of the accuracy rate of all questions answered by the word vectors obtained from the original **GloVe** and our scheme. When the nearest word vector found by cosine similarity is exactly the same as the correct word in the question, the question is answered correctly. In general, the higher the quality of the word vectors obtained through training, the higher the accuracy of answering questions. The experimental results show that the accuracy of the word vectors learned by our privacy-preserving scheme is very close to the original **GloVe** scheme.

We report the effectiveness of Qu et al.'s LDP-BERT word vectors learning scheme with different privacy parameters η. As shown in Table 7, with the increases of η, it means that the less differential privacy noise is added to the original data, the higher the accuracy

Table 7
Accuracy of LDP-BERT word vectors scheme with different privacy parameters.

	Dataset	Semantic (%)	Syntactic (%)	Total (%)
$\eta = 50$	D1	25.39	22.37	23.48
	D2	28.52	23.56	27.13
	D3	29.28	26.73	26.94
	D4	32.71	27.35	29.51
$\eta = 75$	D1	27.84	26.09	27.19
	D2	30.93	27.73	30.12
	D3	33.11	29.29	32.87
	D4	34.23	32.84	33.35
$\eta = 125$	D1	35.24	32.47	33.57
	D2	37.65	36.90	37.53
	D3	40.82	38.27	39.04
	D4	43.59	41.14	42.88
$\eta = 175$	D1	42.35	37.31	40.56
	D2	46.58	44.79	45.12
	D3	51.07	47.63	50.28
	D4	55.32	51.98	53.35

of the learned word vectors, and the corresponding level of privacy protection is also decrease. When $\eta = 175$, the noise level reaches the minimum, and on some datasets, the syntactic and total accuracy of the LDP-BERT scheme are higher than our scheme. In general, the accuracy of our privacy-preserving scheme is better than the scheme adding differential privacy noise, and has a higher level of privacy-preserving.

7.6. Discussion

Our scheme has two performance challenges on large-scale datasets and high-dimensional word vectors: (1) computational complexity: significantly increasing the computational cost of homomorphic encryption. NLP tasks involve matrix and vector operations, which have lower computational efficiency on encrypted data; (2) memory overhead: encrypting data typically requires larger storage space, and as the size and dimensions of the dataset increase, memory overhead will significantly increase.

To address the above challenges, we propose several potential optimization methods for scalability: (1) using model design optimization, such as reducing the computational complexity of privacy protection algorithms through hierarchical training, distributed computing, or model compression techniques; (2) combining multiple privacy protection technologies, such as using differential privacy and homomorphic encryption, can leverage their respective advantages in certain scenarios and reduce performance bottlenecks caused by a single method; (3) at the hardware level, using accelerators such as GPU and TPU to accelerate the computation of privacy protection solutions, especially when dealing with high-dimensional data and large-scale datasets, can significantly improve performance.

8. Conclusion

To ensure the secure and effective generation of high-quality word vectors for NLP tasks, this paper proposes a privacy-preserving word vectors training model. In this model, all computational tasks are executed on a cloud server, ensuring that no privacy-sensitive information can be disclosed or inferred. The collaboration between public and private clouds optimizes the capabilities of the cloud server. Our scheme not only enhances efficiency but also safeguards both user privacy and the confidentiality of the word vectors model.

Building on a robust security framework, we introduce a privacy-preserving scheme based on **GloVe**. This approach employs a hybrid cloud model, the Paillier cryptosystem, and various arithmetic primitives, including data representation, secure multiplication protocols, and logarithmic function computations. We evaluate the performance of our proposed scheme through designed experiments, focusing on the practicality and effectiveness of the generated word vectors. The experimental results indicate that the accuracy of the word vectors produced by our privacy-preserving scheme closely resembles that of the original **GloVe** method.

However, homomorphic encryption can impose significant computational overhead, particularly when applied to large-scale datasets and complex NLP tasks such as **BERT** and **Transformer** architectures. In such cases, encryption operations may become inefficient and challenging to implement on conventional hardware. Additionally, for tasks like machine translation and dialogue generation, homomorphic encryption might hinder the model's ability to accurately capture contextual information, potentially degrading the quality of the generated outputs.

Looking ahead, we aim to investigate the integration of various privacy protection methods for NLP. Specifically, we plan to combine the strengths of differential privacy and homomorphic encryption to enhance data privacy, improve computational efficiency, and elevate model performance.

CRediT authorship contribution statement

Shang Ci: Software, Methodology, Formal analysis, Conceptualization. **Sen Hu:** Validation, Software. **Donghai Guan:** Validation, Supervision, Software. **Çetin Kaya Koç:** Writing – review & editing, Writing – original draft, Project administration, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Cetin Kaya Koc reports financial support was provided by Nanjing University of Aeronautics and Astronautics. Shang Ci reports a relationship with Nanjing University of Aeronautics and Astronautics that includes: employment. Sen Hu reports a relationship with Nanjing University of Aeronautics and Astronautics that includes: employment. Donghai Guan reports a relationship with Nanjing University of Aeronautics and Astronautics that includes: employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported in part by the Jiangsu Province 100 Foreign Experts Introduction Plan BX2022012. This research is also supported in part by TÜBİTAK Projects 2232-118C332 and 1001-121F348.

Data availability

Data will be made available on request.

References

[1] Wu L, Xu Y, Hou J, Chen CLP, Liu C-L. A two-level rectification attention network for scene text recognition. IEEE Trans Multimed 2023;25:2404–14. http://dx.doi.org/10.1109/TMM.2022.3146779.

[2] Ju L, Wang X, Wang L, Mahapatra D, Zhao X, Zhou Q, Liu T, Ge Z. Improving medical images classification with label noise using dual-uncertainty estimation. IEEE Trans Med Imaging 2022;41(6):1533–46. http://dx.doi.org/10.1109/TMI.2022.3141425.

[3] Kumar A, Pratap A, Singh AK. Generative adversarial neural machine translation for phonetic languages via reinforcement learning. IEEE Trans Emerg Top Comput Intell 2023;7(1):190–9. http://dx.doi.org/10.1109/TETCI.2022.3209394.

[4] Wang S, Ge C, Zhou L, Wang H, Liu Z, Wang J. Privacy-preserving classification in multiple clouds ehealthcare. IEEE Trans Serv Comput 2022;16(1):493–503. http://dx.doi.org/10.1109/TSC.2022.3144265.

[5] Mohassel P, Zhang Y. SecureML: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy. SP, San Jose, CA, USA: IEEE; 2017, p. 19–38. http://dx.doi.org/10.1109/SP.2017.12.

[6] Agrawal N, Shahin Shamsabadi A, Kusner MJ, Gascón A. QUOTIENT: Two-party secure neural network training and predictio. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. London United Kingdom: ACM; 2019, p. 1231–47. http://dx.doi.org/10.1145/3319535.3339819.

[7] Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing. EMNLP, Doha, Qatar: Association for Computational Linguistics; 2014, p. 1532–43. http://dx.doi.org/10.3115/v1/D14-1162.

[8] Fu A, Chen Z, Mu Y, Susilo W, Sun Y, Wu J. Cloud-based outsourcing for enabling privacy-preserving large-scale non-negative matrix factorization. IEEE Trans Serv Comput 2022;15(1):266–78. http://dx.doi.org/10.1109/TSC.2019.2937484.

[9] Liu L, Chen R, Liu X, Su J, Qiao L. Towards practical privacy-preserving decision tree training and evaluation in the cloud. IEEE Trans Inf Forensics Secur 2020;15:2914–29. http://dx.doi.org/10.1109/TIFS.2020.2980192.

[10] Zhao Y, Yang LT, Sun J. A secure high-order CFS algorithm on clouds for industrial internet of things. IEEE Trans Ind Inform 2018;14(8):3766–74. http://dx.doi.org/10.1109/TII.2018.2816343.

[11] Clinchant S, Chidlovskii B, Csurka G. Transductive adaptation of black box predictions. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: short papers). Berlin, Germany: Association for Computational Linguistics; 2016, p. 326–31. http://dx.doi.org/10.18653/v1/P16-2053.

[12] Alawad M, Yoon H-J, Gao S, Mumphrey B, Wu X-C, Durbin EB, Jeong JC, Hands I, Rust D, Coyle L, Penberthy L, Tourassi G. Privacy-preserving deep learning NLP models for cancer registries. IEEE Trans Emerg Top Comput 2021;9(3):1219–30. http://dx.doi.org/10.1109/TETC.2020.2983404.

[13] Qiu S, Liu Q, Zhou S, Huang W. Adversarial attack and defense technologies in natural language processing: A survey. Neurocomputing 2022;492:278–307. http://dx.doi.org/10.1016/j.neucom.2022.04.020.

[14] Martinelli F, Marulli F, Mercaldo F, Marrone S, Santone A. Enhanced privacy and data protection using natural language processing and artificial intelligence. In: 2020 international joint conference on neural networks. IJCNN, Glasgow, United Kingdom: IEEE; 2020, p. 1–8. http://dx.doi.org/10.1109/IJCNN48605.2020.9206801.

[15] Fernandes N, Dras M, McIver A. Generalised differential privacy for text document processing. In: International conference on principles of security and trust. Cham: Springer International Publishing; 2019, p. 123–48. http://dx.doi.org/10.1007/978-3-030-17138-4.

[16] Qu C, Kong W, Yang L, Zhang M, Bendersky M, Najork M. Natural language understanding with privacy-preserving bert. In: Proceedings of the 30th ACM international conference on information & knowledge management. 2021, p. 1488–97. http://dx.doi.org/10.1145/3459637.3482281.

[17] Sweeney C, Najafian M. Reducing sentiment polarity for demographic attributes in word embeddings using adversarial learning. In: Proceedings of the 2020 conference on fairness, accountability, and transparency. Barcelona Spain: ACM; 2020, p. 359–68. http://dx.doi.org/10.1145/3351095.3372837.

[18] Wang Q, Du M, Chen X, Chen Y, Zhou P, Chen X, Huang X. Privacy-preserving collaborative model learning: The case of word vector training. IEEE Trans Knowl Data Eng 2018;30(12):2381–93. http://dx.doi.org/10.1109/TKDE.2018.2819673.

[19] Zhang WE, Sheng QZ, Alhazmi A, Li C. Adversarial attacks on deep-learning models in natural language processing: A survey. ACM Trans Intell Syst Technol 2020;11(3):1–41. http://dx.doi.org/10.1145/3374217.

[20] Pan X, Zhang M, Ji S, Yang M. Privacy risks of general-purpose language models. In: 2020 IEEE symposium on security and privacy. SP, San Francisco, CA, USA: IEEE; 2020, p. 1314–31. http://dx.doi.org/10.1109/SP40000.2020.00095.

[21] Coavoux M, Narayan S, Cohen SB. Privacy-preserving neural representations of text. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Brussels, Belgium: Association for Computational Linguistics; 2018, p. 1–10. http://dx.doi.org/10.18653/v1/D18-1001.

[22] Zhang M, Li Z-A, Zhang P. A secure and privacy-preserving word vector training scheme based on functional encryption with inner-product predicates. Comput Stand Interfaces 2023;86:103734. http://dx.doi.org/10.1016/j.csi.2023.103734.

[23] Hua Z, Tong Y, Zheng Y, Li Y, Zhang Y. PPGloVe: Privacy-preserving glove for training word vectors in the dark. IEEE Trans Inf Forensics Secur 2024;19:3644–58. http://dx.doi.org/10.1109/TIFS.2024.3364080.

[24] Jae-yun Kim JL, Cheon JH. Privacy-preserving embedding via look-up table evaluation with fully homomorphic encryption. In: Proceedings of the 41st international conference on machine learning. 2024, doi:https://dl.acm.org/doi/10.5555/3692070.3693050.

[25] Moghaddam AE, Ganesh B, Palmieri P. Privacy-preserving sentiment analysis using homomorphic encryption and attention mechanisms. In: Andreoni M, editor. Applied cryptography and network security workshops. Cham: Springer Nature Switzerland; 2024, p. 84–100. http://dx.doi.org/10.1007/978-3-031-61489-7_6.

[26] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Proceedings of the 27th international conference on neural information processing systems - volume 2. Red Hook, NY, USA; 2013, p. 3111–9, doi:https://dl.acm.org/doi/10.5555/2999792.2999959.

[27] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proceedings of the 31st international conference on neural information processing systems. Vol. 30, Red Hook, NY, USA; 2017, p. 6000–10. http://dx.doi.org/10.48550/arXiv.1706.03762.

[28] Arora S, May A, Zhang J, Ré C. Contextual embeddings: When are they worth it? In: Proceedings of the 58th annual meeting of the association for computational linguistics. Online; 2020, p. 2650–63. http://dx.doi.org/10.18653/v1/2020.acl-main.236.

[29] Feusner JD, Mohideen R, Smith S, Patanam I, Vaitla A, Lam C, Massi M, Leow A. Semantic linkages of obsessions from an international obsessive-compulsive disorder mobile app data set: big data analytics study. J Med Internet Res 2021;23(6):e25482. http://dx.doi.org/10.2196/25482.

[30] Rustam F, Ashraf I, Shafique R, Mehmood A, Ullah S, Sang Choi G. Review prognosis system to predict employees job satisfaction using deep neural network. Comput Intell 2021;37(2):924–50. http://dx.doi.org/10.1111/coin.12440.

[31] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographicTechniques. Springer; 1999, p. 223–38. http://dx.doi.org/10.1007/3-540-48910-X_16.

[32] Nikolaenko V, Weinsberg U, Ioannidis S, Joye M, Boneh D, Taft N. Privacy-preserving ridge regression on hundreds of millions of records. In: 2013 IEEE symposium on security and privacy. Berkeley, CA: IEEE; 2013, p. 334–48. http://dx.doi.org/10.1109/SP.2013.30.

[33] Elmehdwi Y, Samanthula BK, Jiang W. Secure K-nearest neighbor query over encrypted data in outsourced environments. In: 2014 IEEE 30th international conference on data engineering. Chicago, IL, USA: IEEE; 2014, p. 664–75. http://dx.doi.org/10.1109/ICDE.2014.6816690.

[34] Huang X, Du X. Achieving big data privacy via hybrid cloud. In: 2014 IEEE conference on computer communications workshops. INFOCOM WKSHPS, Toronto, ON, Canada: IEEE; 2014, p. 512–7. http://dx.doi.org/10.1109/INFCOMW.2014.6849284.

[35] Bost R, Popa RA, Tu S, Goldwasser S. Machine learning classification over encrypted data. In: Proceedings 2015 network and distributed system security symposium. San Diego, CA: Internet Society; 2015, p. 1–34. http://dx.doi.org/10.14722/ndss.2015.23241.

[36] Shafi G, Micali S. Probabilistic encryption. J Comput System Sci 1984;28(2):270–99. http://dx.doi.org/10.1016/0022-0000(84)90070-9.