PrivItem2Vec: A privacy-preserving algorithm for top-N recommendation

Zhengqiang Ge¹, Xinyu Liu¹, Qiang Li¹,‡, Yu Li¹ and Dong Guo²,‡

Abstract
To significantly protect the user’s privacy and prevent the user’s preference disclosure from leading to malicious entrapment, we present a combination of the recommendation algorithm and the privacy protection mechanism. In this article, we present a privacy recommendation algorithm, PrivItem2Vec, and the concept of the recommended-internet of things, which is a privacy recommendation algorithm, consisting of user’s information, devices, and items. Recommended-internet of things uses bidirectional long short-term memory, based on item2vec, which improves algorithm time series and the recommended accuracy. In addition, we reconstructed the data set in conjunction with the Paillier algorithm. The data on the server are encrypted and embedded, which reduces the readability of the data and ensures the data’s security to a certain extent. Experiments show that our algorithm is superior to other works in terms of recommended accuracy and efficiency.

Keywords
Recommended-internet of things, privacy-preserving algorithm, item2vec, bidirectional long short-term memory, random mapping sets

Date received: 8 January 2021; accepted: 23 October 2021

Handling Editor: Peio Lopez Iturri

Introduction
The internet of things (IoT) is an extended network based on the Internet that enables the interconnection of people, devices, and things at anytime and anywhere.¹ It also means Internet of everything. In the new network, people can get information about various changes, ratings, numbers, clicks, purchases, and other relevant attributes of items through devices. It is important to note that devices are not just phones. Devices also include third-party computing platforms, servers, social networks between people, and more, which constitute the uncertainty and variability of the IoT. However, devices in the IoT generate tons of information in various formats, and the problem of information overload makes it difficult for users to quickly obtain effective information. Therefore, recommendation algorithm is proposed to solve the problem of information filtering.²

To meet user’s personalized needs, recommendation algorithm analyzes user’s personal preferences, according to its behavior and attributes (e.g. browsing, purchasing, and other behavior information or the user’s gender, age, and place of residence). The traditional recommendation algorithm, collaborative filtering,³–⁵ collects the user’s preferences and recommends based on the user’s historical behaviors. However, data mining is highly vulnerable to the user’s privacy, so the privacy

¹College of Computer Science and Technology, Jilin University, Changchun, China
²Key Laboratory of Symbol Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, China

Corresponding author:
Dong Guo, Key Laboratory of Symbol Computation and Knowledge Engineering of Ministry of Education, Jilin University, No. 2699 Qianjin Street, Changchun 130012, China.
Email: guodong@jlu.edu.cn

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
protection of the recommended system has gradually become an important issue.6

Currently, common privacy recommendations can be summarized as the following two algorithms: random disturbance and homomorphic encryption.7 Both algorithms have advantages and disadvantages; random disturbance ensures data privacy by adding data noise, but sacrificing accuracy, while the homomorphic encryption encrypts all data at the expense of algorithm efficiency. Therefore, the implementation of the privacy recommendation algorithm must balance the effect of recommendation with degree of user confidentiality.

Zhan et al.8 have proposed a more effective privacy protection collaboration recommendation system based on the standard product protocol compared to mainstream cryptology. Casino et al.9 proposed a protocol based on the data perturbation while ensuring user’s K-anonymity. Recommended algorithms that use K-anonymous methods for encryption still must consider solving only specific attacks. The frameworks mentioned in literature are to separate the user side from the server side, making it impossible for the server side to obtain the user’s data but to transfer only relevant parameters. However, the user’s privacy remains a threat. For example, when combined with a reconstruction attack, attackers can still gain access to the user’s private information.10–12 Even if noise is added to the user-item matrix, it is highly likely that the server will rebuild the user’s true rating. Therefore, determining preferred method to better protect user’s privacy from malicious attacks remains a huge challenge. In addition to privacy issues, there are two issues that deserve attention:13,14 (1) the accuracy of recommended results and (2) data sparseness: the user-item matrix is often a huge sparse matrix, making it difficult to analyze and calculate. Therefore, we would like to find a privacy recommendation algorithm to deliver the best possible assurance of accuracy and efficiency.

In recent years, combination of deep learning and recommended algorithms has been greatly discussed.15 Barkan and Koenigstein16 combined Word2Vec with item-based collaborative filtering (CF) to propose the item2vec algorithm. This method is effective for the processing of sparse data and is superior to singular value decomposition algorithm in the classification accuracy of data sets. However, recommendation of the item2vec lacks the time effect; however, biLSTM can solve this problem. Therefore, the item2vec algorithm is chosen as the basic algorithm for data processing.

For the current work of privacy-preserving recommendation algorithm, there are several problems with the method based on random distribution. Although this method can effectively protect the user’s privacy, it causes a significant decline in the accuracy of recommendation and thereby loses the significance of practical use. In addition, the method based on the secure multiparty computation will not cause accuracy loss; however, it requires a lot of computational time, resulting in low efficiency due to the need to add relevant cryptographic operations to entire computational process.

Based on the above problems, we propose a privacy recommendation algorithm, PrivItem2Vec, and selected the item2vec and biLSTM models for data pre-processing to compress the data dimensions and improve the transmission efficiency, using the homomorphic encryption method, which only encrypts part of the data, such as the user ID, and sends it to the server. In the homomorphic encryption scenario, the data can still be computed without decryption, and the results will be returned to user for decryption. The server and intermediate participants will not get any relevant privacy data; while ensuring recommendation accuracy, it introduces less redundant computation. The main contributions of this article are as follows:

1. The new privacy recommendation algorithm PrivItem2Vec. We propose the concept of recommended-internet of things (R-IoT) and a privacy recommendation algorithm, PrivItem2Vec. Using the Paillier encryption algorithm to construct an encrypted index mapping set, the server can only obtain encrypted or low-readability information. In the process of data computation and information transmission of the recommended algorithm, the encrypted key belongs to the client, and the server cannot interpret any of the acquired information. In addition, we improve the recommendation efficiency by establishing an encrypted privacy mapping set to reduce the amount of computation.

2. Add biLSTM for improving efficiency. We integrate biLSTM into the recommendation algorithm based on item2vec and reduce the dimensionality of the data. At the same time, we make up for the shortcomings of the recommendation results that do not consider the timing of the data. The joint training of biLSTM and item2vec improves the final recommendation hit rate (HR) and recommendation accuracy rate of the recommendation algorithm. We chose cosine similarity when calculating the recommended sequence, and the Top-N recommended sequence is obtained, using heap sorting to reduce time complexity.

3. The better experiment results. We compare and analyze with the popular methods proposed in recent years by testing on the real data set. When comparing the time of algorithm before and after using Paillier encryption, we can prove that this method will not change the
recommendation results, and the recommendation effect is better than some algorithms that have practicability.

We discuss several related works of privacy recommendation algorithms based on random disturbance and secure multiparty computation (SMPC) in section “Related work.” In section “Theoretical basis,” we briefly introduce the theoretical basis of recommendation algorithms and homomorphic encryption. In section “PrivItem2Vec,” we propose the algorithm PrivItem2Vec and provide the description in detail, the parameter settings, and the security analysis. In section “Experimental results and analysis,” we introduce the experimental results and analysis and give the evaluation indicators, the experimental data set, and the comparison and analysis with the previous works. In the Conclusion section, we summarize our key points about our work and its impact, and as applicable, we may recommend new areas for future research; the Reference section will contain our cited references.

**Related work**

The GDPR Act highlights the issue of personal privacy, and the use of big data increases access and analysis to the user’s sensitive privacy data, endangering the privacy of users and violating data protection laws when handling sensitive information. Therefore, privacy recommendations have always been a concern for discussion. The privacy recommendation algorithms are divided into two main categories. The first category is the random disturbance method; before the user passes the information to the server, noise is added to the information to achieve the effect of privacy protection. Polat and Du proposed an item-based Top-N retractable privacy recommendation algorithm and combined the MinHash and Jaccard similarity calculation, so that user’s privacy in the item similarity calculation process is strictly protected. Users move in the anonymous random to add their own information, so that other users or the server cannot know whether a piece of data are from the target user; they achieved efficient calculation with low-complexity calculation. Guerraoui et al. proposed a distance-based differential privacy algorithm, D2P. D2P can be applied to solve common CF recommended systems’ privacy problems that are based on k-nearest neighbors (KNN) and achieved a good trade-off between privacy and overhead. To ensure the privacy of the recommendation system, Zhu and Sun proposed an item-based differential privacy recommendation algorithm and a user-based differential privacy recommendation algorithm. The item-based privacy recommendation method calculates relevant item list based on a rating matrix when calculating an item that a user might like and recommends it to the user, based on the history of purchase. The user’s privacy is guaranteed by combining the similarity of the index mechanism with the sensitivity when calculating the related items. Xiong et al. proposed a client/server framework to create a private recommendation system (PrivateRS). The system assumes that the server side is not trustworthy. On the client side, each user initially rates the item and then randomizes the rate using differential privacy mechanism.

The second category is SMPC based on homomorphic encryption algorithms. The main problem solved by SMPC is how parties without a trusted third party can jointly calculate the same problem without disclosing their own private data, which is also an important application of cryptography. Therefore, for semi-honest or malicious servers or user partners to achieve personalized recommendation without disclosing personal privacy information, SMPC is theoretically a good choice; however, recommendation algorithm is often faced with tons of levels or significant data checks; such data encryption is undoubtedly a challenge to the client’s device, and the user’s device performance requirements are very high. Therefore, in the process of using secure multiparty computing and related cryptography, we must consider how to improve the efficiency of computing and the saving of resources. Wang et al. proposed an encryption-friendly machine-learning method and proposed the protocol CrytoRec, a security two-party protocol that does not require the interaction. To resolve the privacy recommendation system, Wang et al. designed a privacy-protected multi-user word vector training joint model learning protocol in the scenario of a cloud server and password service provider (CSP) that is assumed to be non-complicit. We compare and summarize the categories and technical points of the above related work as shown in Table 1.

### Table 1. Comparison of categories and technical points of related work.

| Article | Category         | Technical point                        |
|---------|------------------|----------------------------------------|
| Polat and Du | Random disturbance | MinHash, Jaccard similarity          |
| Taziki et al. | Distance-based differential privacy | Index mechanism, sensitivity         |
| Zhu and Sun | Differential privacy | Homomorphic encryption               |
| Xiong et al. | SMPC              | Homomorphic encryption, word vector    |
| Wang et al. | SMPC              | Homomorphic encryption, biLSTM         |
| Our work  | SMPC              |                                        |

SMPC: secure multiparty computation.
Theoretical basis

Item2vec and biLSTM

Personalized recall algorithm. Recall in the recommended algorithm refers to the selection of a portion of items as a candidate set from the item set. The reason for selecting a portion of the data is that different users in actual recommendation system will like different types of items, followed by service performance considerations. For example, if you choose the full set as a candidate set, the item’s magnitude is large, and the subsequent sorting will consume a lot of time, causing a huge delay in the response time of recommended system back-end service. A personalized recall is a candidate set based on the characteristics of user’s information, such as user’s attributes, user’s behavior, user’s context, and other relevant information, to select items from the entire set of items that the user will be interested in. During the recommendation process, the recall determines the upper bound of the final recommendation. In addition, contents in the candidate set will eventually be presented to user’s item, so the recall algorithm needs to be carefully selected.

Item2Vec. In the existing recommendation algorithms, Item2Item recommendation’s effect is significant. In many scenarios, Item2Item is recommended as a better method than User2Item. Item2Item recommendation means that after obtaining the item similarity matrix, a similar item is found based on the item associated with the user’s recent behaviors and then completes the recommendation, such as the ItemCF method. User2Item refers to the user based on the basic attributes, historical behaviors, etc. and uses a certain model to calculate the user’s most-liked item list, when the user accesses the system; it is recommended to the system, such as UserCF, latent factor model, Person Rank, and other algorithms. The neural network has the characteristic abstract ability. The input and hidden layers are all connected in the neural network, and the abstract ability of the neural network is more powerful than the shallow model. The algorithm item2vec, based on neural network Item2Item, is produced in this context.

The physical meaning of item2vec. (1) Convert the sequence of behavior generated by the user into a sentence composed of the item. In a system, such as a common e-commerce system, or a movie, or book scoring system, the user performs browsing and clicks favorites, reviews, and other behaviors that can be extracted from the log of the user’s behavior sequence in a given day. This series of behaviors have corresponding items that can be abstracted with the item sequence related to the user’s behavior on a given day. The relationship between the item and the sequence of that item, which is composed of each user’s behavior, also forms a relationship similar to word and word in Word2Vec. (2) Item embedding is similar to word embedding. Word embedding must provide the language material into the Word2Vec training to get the word vector, and the vector can be characterized whether the word is far or near. Using the same method, the item sequence obtained in (1) as input into the Word2Vec can be used to get the result of item embedding, and the resulting vector can also be characterized between implicit semantics of the near range, that is, the similarity between the items.

BiLSTM and item2vec. LSTM is known as the long short-term memory, which is part of the recurrent neural network (Figure 1). LSTM is ideal for modeling the time series data, such as text data, because of its design characteristics. BiLSTM is an acronym for bidirectional long short-term memory and is a combination of

---

Figure 1. The hybrid model of biLSTM and item2vec for data processing. The final movie embedding results, which combine timing and users’ preference, are finally obtained.
forward LSTM and forward LSTM. Both are often used to model contextual information in the natural language processing tasks. When considering the lack of timing to improve the accuracy of the algorithm we propose in this article, aILSTM allows us to better capture two-way semantic dependencies in the model.

As shown in Figure 3, each user who played excessive movies is entered into the model through item2vec and biLSTM. The output sequence we obtain at this point will take into account the user’s rating preferences and the impact of rating time.

Paillier public key homomorphic encryption algorithm

Homomorphic encryption algorithm. The homomorphic encryption algorithm, which was originally proposed by Rivest et al., is a special encryption algorithm. The result of direct addition or multiplication of the redaction obtained by the encryption algorithm is the same as the result of the addition or multiplication operation in which the text is done in advance. Homomorphic encryption algorithm that satisfies the two equations is called addition and multiplication homomorphic encryption algorithm. If two equations are met at the same time, it is called full homogenous encryption. The most widely used homogenous encryption algorithms are RSA multiplication homomorphic encryption algorithm and Paillier plus homomorphic encryption algorithm. A little later, another algorithm was proposed, known as the Gentry full homogeneous encryption algorithm; however, the Gentry algorithm’s running process requires a lot of space to store public keys, greatly increasing the difficulty with the application’s practical operation.

Homomorphic encryption can be used in the construction of privacy-preserving machine-learning algorithms, as well as in federated learning. It plays an important role in federated learning when delivering ciphertext gradients to the central parameter server. It can also be used in large-scale genome association research.

Description of the Paillier encryption. Paillier encryption is based on the difficult problem of composite residual classes. The encryption process is divided into three steps: key generation, encryption, and decryption.

The basis for the security and stability of the Paillier encryption algorithm: The Paillier encryption mechanism is semantically secure and protects against ciphertext selection attacks, if and only if the decision compound residue hypothesis is satisfied.

The reasons for choosing Paillier are as follows: First, during the data transmission between the user side and the server side, it is very likely to be attacked and the data will be obtained by a third party. To avoid being deciphered by an attacker to obtain a direct guess of user’s preferences, the nature of Paillier homomorphic encryption is used in this article. We construct a pseudo item-id to protect user’s preference. Second, requirements and load of the homomorphic encryption algorithm for the device cannot be ignored, and the Paillier algorithm can not only be used for the public key addition homomorphism encryption but also can be better combined with a variety of cloud computing applications. Given the security of private information, data such as bank credit from a user or platform, can contain a large amount of private or corporate information, often without trust in the data processing by a separate third-party cloud platform. But if we combine this method with the homomorphic encryption technology, then users can use the system with confidence. Same-state encryption applied to cloud services can solve the cloud service data in the confidential storage and confidential computing problems. Considering the computing load on the user side, the next step in the algorithm’s improvement will be related to joining cloud computing, placing encrypted data on the cloud platform, and reducing the computational and storage pressure on the client. Third, relying only on the item recommendation form is too single, if we consider joining the social network in the future. Then for the social network semi-honest or malicious friends, also must be encrypted after the transmission of the data processing, at this time the Paillier algorithm homogeneity will be greatly used.

PrivItem2Vec

Algorithm description

Traditional models are typically server-centric, as Figure 2 shows. Considering the existence of semi-
honest or malicious servers, the server side cannot access the user’s privacy-related data, so the user side does not pass data directly to the server side.

We need to avoid server centralization. Hence, we construct a model that the users collaborate with the server. To protect against reconstruction attacks in the transmission process, we must combine the encryption algorithms. Figure 3 shows the architecture of the PrivItem2Vec, and the recommended algorithm based on the item2vec and biLSTM algorithm is selected to process the vector file. We use the Paillier encryption algorithm and vector files to build a map set of encrypted items, vectors, and pass them to the server side. We subtract random number from the item and encrypt it, using the Paillier algorithm, to form the encrypted pseudo item, which is passed to the server side at the same time as the encrypted random number. This action ensures that the data received on the user side is encrypted or low readable.

The use of the pseudo item on the server side is added to the number of random numbers after encryption, and the encrypted item is obtained. Next, we look up in the vector file and use cosine similarity to calculate the similarity between the vectors and sort the Top-N sequence using the large root heap. It is important to note that the Top-N sequence is still an encrypted sequence at this time. Then, we use the biLSTM model to normalize the timestamps for each movie, reorder the Top-N sequence, and eventually decrypt it on the user side to get a series that is recommended to the user.

Why not encrypt the vector data and choose to construct an encrypted map set? First, we combined a basic principle of hybrid encryption, “encrypting shorter symmetric keys with public key encryption algorithms, encrypting larger data with symmetric encryption algorithms.” The Paillier encryption algorithms are the public key homomorphic encryption algorithms, so the choice of a relatively brief item-id for encryption construction is also intended to reduce computational cost on the user side. If the vector is directly encrypted, the Paillier algorithm cannot meet the operational requirements in the subsequent similarity calculation process.

**Experiment implementation and parameter settings**

The experimental platform uses a virtual machine Ubuntu 16.04, 2 GB of memory and win 10 system, CPU 3.40 GHz, and a 64-bit operating system with 16 GB of memory. The experimental implementation language is JAVA and Python, and the connection bridge uses Jyype.

The Google version of the Word2Vec algorithm was used during the training. Word2Vec’s parameter settings:

- The vector dimension size of the embedding is 128;
- the context window length is the default value 5; the hs is 0 using negative sampling; the number of iterations in training is 50; the training mode uses skip-gram and thus cbow is equal to 0; the learning rate alpha is 0.025 (Table 2).

![Figure 3. The proposed recommended algorithm, PrivItem2Vec. The algorithm structure consists of the user side and the server side. The item of the user’s query is encrypted and passed to the server side. On the server side, the top recommendation list is calculated through the operation of a random mapping set and the homomorphic encryption algorithm, and then the encrypted list is decrypted on the user side.](image)

**Table 2. Parameter settings for Word2Vec.**

| Parameter | Setting |
|-----------|---------|
| size      | 128     |
| window    | 5       |
| hs        | 0       |
| iter      | 50      |
| cbow      | 0       |
| alpha     | 0.025   |
Algorithm 1. EncryptItem.

Input: itemvec
Output: key, itemvec_dic

1. k = encrypt.key() #Get the private key
2. for line in item_vec:
3. item_id = Encrypt.encryption( item_id, key) item_id
4. itemvec_dic[item_id] = for vec in item_vec[1:]
5. end for
return key, itemvec_dic

Algorithm 2. Random_Map.

Input: item_id, key
Output: pseudo_item_id, random_index, random

1. random = random.randint(1,10)
2. random_index = index[random]
3. item_id = item_id—random
4. pseudo_item_id = Encrypt.encryption(item_id, key)
return pseudo, random_index

Algorithm 3. ItemSimilarity.

Input: pseudo_item_id, random
Output: entopk_list

1. enRandom = enIndex[random]
2. enItem_id = homomorphic.add(pseudo_item_id + enRandom)
3. itemvec = itemvec_dic[enItem_id]
4. for tmp_item_id in itemvec_dic:
5. if tmp_item_id == enItem_id:
6. continue
7. tmp_itemvec = itemvec_dic[tmp_item_id]
8. denominator =|tmp_itemvec|*|itemvec|
9. if denominator == 0:
10. scores[tmp_item_id] = 0
11. else:
12. scores[tmp_item_id] = dot(itemvec, tmp_vec)/denominator
13. for score in scores:
14. entopk_list = sorted(score)[:topk]
return entopk_list

Algorithm 1

EncryptItem. The user side uses the itemvec file processed by Word2Vec, using the Paillier encryption algorithm to encrypt the item-id and construct the encrypted id-vector map set as itemvec_dic and passes it to the server.

In the Algorithm 1, what we need to do is to use the private key in the Paillier homomorphic encryption algorithm to encrypt the id of each item, which is analyzed in advance. After the ID is encrypted and sent to the server, since the server does not have a key for decryption, the server can only perform corresponding operations on the ciphertext and cannot obtain any information about the user, so the user’s data privacy can be guaranteed. In Step 1, we obtain the key required to encrypt the id through the Paillier algorithm. From Steps 2 to 5, we traverse each row of data in item_vec and use the key to encrypt the corresponding item_id. Assuming that the number of rows is n, the time required for traversal is O(n), and assuming that the security parameter of the Paillier algorithm is τ, the time complexity of the Algorithm 1 is O(τn).

Algorithm 2

Random_Map. The user side makes the pseudo item through the constructed random mapping set and sends it to the server side, so that even if the attacker obtains data during the passing process, the result is the pseudo item. The attacker cannot obtain the user’s preference information based on the pseudo item.

Algorithm 2 further improves the privacy of data. On the premise that it has been encrypted, the randomness is added, making it more difficult for the server to distinguish the order of user data, and it is unable to associate specific data with specified users, further ensuring privacy. Step 1 is to randomly generate random numbers. Step 2 is to rearrange the data position random index according to the random number obtained. The “item id-random” in Step 3 further randomizes item id. Step 4 obtains the pseudo item id after randomization and encryption. All other steps can be completed in O(1), and the time complexity required for Step 4 is still O(τ) for the Paillier algorithm.

Algorithm 3

ItemSimilarity. Server receives the pseudo item from the user and the random index of the random mapping set. Then, the server uses the random index to find the corresponding random number of encrypted numbers. We use the encryption algorithm to make the random and pseudo item add-up to get the real encrypted item, enItem_id. Then, according to the ciphertext vector mapping dictionary, itemvec_dic, for vector similarity calculation, we use the cosine similarity calculation method. Finally, we sort to take out the Top-N encryption recommendation list and send it back to the user side.

Algorithm 3 first uses homomorphic addition to obtain the encrypted user ID and then uses the corresponding algorithm to get the Top-N encryption recommendation list, which no longer involves the homomorphic operations, and it is no different from the algorithm performed under plaintext. The homomorphic addition is due to the randomness introduced by the previous algorithm. Steps 1–2 are to recover the original item_id before randomization. In Step 2, the additive homomorphism of the homomorphic encryption algorithm is used. The previously encrypted two
ideal interaction

Z

ties send their inputs to a trusted third party to

parties execute the protocol

briefly introduced as follows: in a real interaction, the

to prove the security of the protocol. This paradigm is

the real-world ideal world simulation paradigm method

about the data transmitted during the process. We use

according to the protocol correctly and are only curious

semi-honest security model; that is, they are executed

system, including the user and the server, both obey the

assume that all participants in this recommendation

We have added the security proof for our scheme. We

numbers pseudo_item_id and enRandom are added

under the ciphertext. It is the same as the result of adding

them in plaintext and then encrypting, so the confidentially of item_id is still maintained. Step 3 finds the
data vector, corresponding to the ciphertext enItem_id.

Steps 4–12 calculate the score, corresponding to each item. The time complexity required for traversal is

O(n). Steps 13–14 sort the final result to get the Top-N
list, using the heap sort, and the required time complexity is O(\log n).

Algorithm 4.

**DecryptItem.** The user side receives encrypted Top-N
sequences, the encrypted Top-N list that is transmitted
back by the server after calculating the similarity. The list contains recommended item list and the correspond-
ing similarity value for each item. Then, we use the Paillier encryption algorithm to decrypt the
encrypted item-id in the list and output the decrypted Top-N recommended item and its similarity.

Algorithm 4 is to recover the plaintext data from the results calculated by the server and obtain each user ID
and its corresponding recommendation data. Steps 1–4 traverse the topK list, using the decryption method of
the Paillier algorithm to decrypt corresponding ID and get the similarity. Assuming that the length of the list is
k and the security parameter is \( \tau \), the total time complexity is \( O(\tau k) \).

Security analysis

We have added the security proof for our scheme. We assume that all participants in this recommendation
system, including the user and the server, both obey the semi-honest security model; that is, they are executed
according to the protocol correctly and are only curious about the data transmitted during the process. We use
the real-world ideal world simulation paradigm method to prove the security of the protocol. This paradigm is
briefly introduced as follows: in a real interaction, the parties execute the protocol \( \pi \) in a certain environment
Z, where an adversary \( A \) exists, and there is another ideal interaction \( I \). In the ideal environment \( I \), all par-
ties send their inputs to a trusted third party to implement the protocol \( F \) completely. Finally, to prove
the security of the protocol, for each adversary \( A \) that exists in the real interaction \( R \), there is a simulator \( S \) in
the ideal interaction, and if the environment cannot distinguish between the two interactions, this protocol is
secure. Thus, the information obtained by the adversary in the real interaction and the information obtained by
the simulator in ideal interaction are the same in the category of informatics and are indistinguishable. As a
proof, it is only necessary to check whether the designed simulator has the ability to generate messages that are
indistinguishable from real-world interaction messages.

In our protocol, sending data only occurs after the user performs the homomorphic encryption and ran-
don substitution and after the server calculates the result. Since the other steps do not involve data trans-
mission, the calculation is only performed locally, and simulation is not required. The data sent by the user is
calculated by homomorphic encryption and random substitution. The security of our algorithm is based on
the security of the Paillier algorithm, which is used for homomorphic confidentiality to ensure that the
encrypted data are indistinguishable from the plaintext data in the category of informatics, and the plaintext
cannot be decrypted without the private key. Under the ideal model, the simulator \( S \) can also send data with the
same distribution of the model to the ideal function, and it cannot be distinguished. Therefore, the Paillier
algorithm can ensure that the server cannot obtain any information about the data sent by the client to the ser-
ver, such as the user ID. Furthermore, we use random \( S \) substitution to further destroy the association
between the data and user’s sensitive information; it is more difficult for the server to steal private data. When
sending data from the server back to the client, since the private key to decrypt the data is only known to the
user, the user can obtain the plaintext by decrypting the data locally after receiving the data, so other partici-
pants will not obtain the data. Therefore, our protocol can securely implement the expected privacy recom-
mandation model.

Experimental results and analysis

Evaluation indicators

The evaluation of the Top-N recommendation algo-

rithm is commonly used by the HR. The definition of

HR is as follows

\[
HR = \frac{\#hits}{\#users}
\]  

(1)

\#users represents the number of users in the test set, and \#hits represents the number of users successfully
recalled by the model and recommended. The HR can
measure the performance of the model directly on the data that the user has given feedback.

Discounted cumulative gain (DCG): taking into account the sort order factors, the top-ranked item gain is higher, and the bottom-ranked item is discounted.

\[
DCG@N = \sum_{i=1}^{N} \frac{reli}{\log_2(i+1)}
\]  

Normalized DCG: an ideal DCG score is calculated for each user, represented by IDCG, and then the ratio of each user’s DCG to IDCG is used as the normalized score for each user. Finally, the final score, the NDCG, is averaged for each user

\[
nDCG@N = \frac{DCG@N}{IDCG@N}
\]

Precision represents the prediction results, and its meaning is the probability that the actual positive sample is in all the samples predicted to be positive, and the expression is as follows

\[
\text{precision} = \frac{TP}{TP + FP}
\]

**Experimental data set**

The algorithm is evaluated in ml-latest-small, a sub-data set in the Data Set of GroupLens Research provided by GroupLens Research. The data are described in Table 3, which contains 100,386 ratings for 9472 movies by 610 users, with each user rating at least 20 movies, representing how much users like the movie. The data format is userId, movieId, rating, and timestamp. Timestamp records the time users submitted reviews since 29 March 1996.

**Methods for comparison**

The main application scenario of the PrivItem2Vec method proposed in this article is the Top-N method in the recommendation system, so the common recommended algorithms of Top-N is selected.

BPR: Bayesian personalized ranking, which does not optimize the user’s rating of items but only by scoring to optimize the user’s sorting of items; it is a real ranking algorithm.

NCF: The neural collaborative filtration method proposed by He et al.\(^ {31} \) improved the recommended algorithm using the multi-layer sensor fusion generalized matrix decomposition.

Item2vec: The basic algorithm demonstrated in this article.

| Type | Detail |
|------|--------|
| Name | ml-latest-small |
| Total number | 100836 |
| Users | 610 |
| Items | 9742 |
| Rating scale | 1–5 |

**Table 4. List of recommendations for PrivItem2Vec and item2vec.**

| Item-id | Similarity | Item-id | Similarity |
|---------|------------|---------|------------|
| 87      | 0.841      | 38      | 0.766      |
| 56      | 0.797      | 73      | 0.741      |
| 38      | 0.782      | 71      | 0.727      |
| 93      | 0.735      | 48      | 0.686      |

**Experiment comparison and analysis**

**Experiment 1.** The data tested using the PrivItem2Vec algorithm were directly compared with the item2vec algorithm, and the following table shows a set of test samples and results. When the test data item-id is 54, the resulting Top-5 recommended items, and its corresponding similarity calculation results are as follows in Table 4. PrivItem2Vec is on the left and item2vec is on the right, both of which have repeated items, but it is clear that the similarity on the left side can be observed more precisely.

**Experiment 2.** The BPR, NCF, item2vec, and PrivItem2Vec (item-lstm) algorithm run on the data set, and the results of the four methods in the Top-N recommendation are calculated and compared. From Figure 4, it can be seen that the recommended hr, nDCG, precision of BPR, NCF, and item2vec are lower than the PrivItem2Vec (item-lstm) algorithm in the process of changing of N; our algorithm does not lose the accuracy and can still be recommended efficiently. Here, we need to emphasize that all the results are the mean values obtained through 10 runs, so we also give the corresponding standard deviation bars in the figure below.

**Experiment 3.** We simulate an attack on the data transfer process, using Wireshark to grab and parse the passing data. The data in the packet are encrypted data or a mapping file consisting of encrypted data and vector sets, at which point even if the server is maliciously complicit with the attacker, the encrypted data cannot
be parsed because only the user side has the key. Even if the key is obtained, the passed item data are calculated pseudo-data with a random number, which does not indicate the user’s true query goals and preferences. It is proved that the PrivItem2Vec method can guarantee the user’s privacy and security.

**Experiment 4.** The time consumption of PrivItem2Vec method and item2vec-based recommendation algorithm in constructing the crypto-map sets on the user side varies with the N shown in Figure 5, and the similarity calculation time consumption of the two methods is shown in Figure 6. Combined with the experimental results, the average running time difference between the two methods on the server side was 0.148 s, while the time of constructing the user-side’s privacy map set was close to 16 s, which was 25.76 times that of non-encrypted. Although the user side needs time to generate the mapping set, only one
calculation is required during one update. The subsequent queries’ operation does not need to repeatedly submit the mapping set to the server. To use the time consumption of the first query with passing the map set, the system only needs to regularly update the data set when the user is not in use and pass it to the server, so the time consumption of the user side constructing the mapping set does not affect users more when they use it.

**Experiment 5.** The decryption time of Top-N encryption recommendation list received by the user, which is back from the server changes with the N value, as shown in Figure 7. We can see that the consumption time changes nearly linearly with the change of the N value, and the time system decrypts for each encrypted item-id is almost identical. The N value in Top-N sets by the daily query is usually within 100 and is intended to recommend the relevant items that most likely are to be liked by the users. When the N setting is too large, the subsequent recommendation is almost meaningless. The user’s purpose is getting relevant recommendations quickly, without having to spend time browsing. When the number of recommended results exceeds a certain limit, we call it invalid recommendations, and users rarely browse through the recommended items in the invalid recommendation. When N is 100, the time to decrypt is 0.270305 s, at which point the additional consumption time is negligible when compared to the total run time.

**Conclusion**

In this article, the Paillier encryption algorithm is applied to the recommended algorithm based on the item2vec and biLSTM implementation. We used the biLSTM algorithm to compensate for the lack of timing in item2vec and to improve the accuracy of recommendations. By building a random set of crypto-maps and combining vectors, we can protect against malicious attacks during the transmission of the user-server. We use heap sorting to improve recommendation efficiency when recommending. Compared with the existing typical recommendation algorithm, the recommendation effect of the PrivItem2Vec algorithm is proved to be superior to other algorithms discussed in this article. In addition, when we compared the operation time with the unencrypted algorithm, the server side using the
encryption algorithm’s time consumption is 1.63 times than before encryption; the average decryption time of the user encrypting an item is 0.0028 s. Regardless of the time-consuming data preparation, the single recommendation time is almost unaffected. In the future work, we will consider extending the algorithm to support more kinds of recommendation algorithms, and how to continue to ensure data privacy and recommended efficiency when social networks are added to the algorithm.

Data availability
The MovieLens data supporting this study are from previously reported studies and data sets, which have been cited. The processed data are available at www.grouplens.org.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the National Natural Science Foundation of China (grant nos. 62072208 and 61772229) and the International Science and Technology Cooperation Projects of Jilin Province (grant no. 20210402082GH).

ORCID iDs
Qiang Li https://orcid.org/0000-0001-7510-4718
Dong Guo https://orcid.org/0000-0003-1341-6390

References
1. Madakam S, Ramaswamy R and Tripathi S. Internet of Things (IoT): a literature review. J Comput Commun 2015; 3(5): 164–173.
2. Resnick P and Varian HR. Recommender systems. Commun ACM 1997; 40(3): 56–58.
3. Goldberg D, Nichols D, Oki BM, et al. Using collaborative filtering to weave an information tapestry. Commun ACM 1992; 35(12): 61–70.
4. Herlocker JL, Konstan JA, Terveen LG, et al. Evaluating collaborative filtering recommender systems. ACM Trans Inform Syst 2004; 22(1): 5–53.
5. Linden G, Smith B and York J. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Intern Comput 2003; 7(1): 76–80.
6. Ramakrishnan N, Keller BJ, Mirza BJ, et al. When being weak is brave: privacy in recommender systems, 2001, https://arxiv.org/abs/cs/0105028
7. Jun Z, Xiaolei D and Zhenfu C. Research advances on privacy preserving in recommender systems. J Comput Res Develop 2019; 56(10): 2033.
8. Zhan J, Hsieh CL, Wang IC, et al. Privacy-preserving collaborative recommender systems. IEEE Trans Syst Man Cyber Part C Appl Rev 2010; 40(4): 472–476.
9. Casino F, Domingo-Ferrer J, Patsakis C, et al. A k-anonymous approach to privacy preserving collaborative filtering. J Comput Syst Sci 2015; 81(6): 1000–1011.
10. Aggarwal CC. On randomization, public information and the curse of dimensionality. In: Proceedings of the 2007 IEEE 23rd international conference on data engineering, Istanbul, 15–20 April 2007, pp.136–145. New York: IEEE.
11. Li D, Lv Q, Shang L, et al. Efficient privacy-preserving content recommendation for online social communities. Neurocomputing 2017; 219: 440–454.
12. Shobana S and Nagajothi P. Deriving private information from randomized dataset using data reorganization techniques. Data Min Knowl Eng 2012; 4(4): 191–194.
13. Bobadilla J, Ortega F, Hernando A, et al. Recommender systems survey. Knowl Based Syst 2013; 46: 109–132.
14. Gunawardana A and Shani G. A survey of accuracy evaluation metrics of recommendation tasks. J Mach Learn Res 2009; 10(100): 2935–2962.
15. Batmaz Z, Yurekli A, Bilge A, et al. A review on deep learning for recommender systems: challenges and remedies. Artif Intell Rev 2019; 52(1): 1–37.
16. Barkan O and Koenigstein N. ITEM2VEC: neural item embedding for collaborative filtering. In: Proceedings of the 2016 IEEE 26th international workshop on machine learning for signal processing (MLSP), Vietri sul Mare, 13–16 September 2016, pp.1–6. New York: IEEE.
17. Gruschka N, Mavroeidis V, Vishi K, et al. Privacy issues and data protection in big data: a case study analysis under GDPR. In: Proceedings of the 2018 IEEE international conference on big data, Seattle, WA, 10–13 December 2018, pp.5027–5033. New York: IEEE.
18. Polat H and Du W. Privacy-preserving top-N recommendation on horizontally partitioned data. In: Proceedings of the 2005 IEEE/WIC/ACM international conference on web intelligence (WI’05), Compiegne, 19–22 September 2005, pp.725–731. New York: IEEE.
19. Guerraoui R, Kermarrec AM, Patra R, et al. D2P: distance-based differential privacy in recommenders. Proc VLDB Endow 2015; 8(8): 862–873.
20. Zhu X and Sun Y. Differential privacy for collaborative filtering recommender algorithm. In: Proceedings of the 2016 ACM international workshop on security and privacy analytics, 2016, pp.9–16, https://www.researchgate.net/publication/311464434_Differential_Privacy_for_Collaborative_Filtering_Recommender_Algorithm
21. Xiong P, Zhang L, Zhu T, et al. Private collaborative filtering under untrusted recommender server. Fut Gener Comput Syst 2020; 109: 511–520.
22. Wang J, Arriaga A, Tang Q, et al. CryptoRec: privacy-preserving recommendation as a service, 2018, https://arxiv.org/abs/1802.02432v4
23. Wang Q, Du M, Chen X, et al. Privacy-preserving collaborative model learning: the case of word vector training. IEEE Trans Knowl Data Eng 2018; 30(12): 2381–2393.
24. Rivest RL, Adleman L and Dertouzos ML. On data banks and privacy homomorphisms. Found Secur Comput 1978; 4(11): 169–180.
25. Shand M and Vuillemin J. Fast implementations of RSA cryptography. In: Proceedings of the IEEE 11th symposium on computer arithmetic, Windsor, ON, Canada, 29 June–2 July 1993, pp.252–259. New York: IEEE.

26. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of the international conference on the theory and applications of cryptographic techniques, Prague, 2–6 May 1999, pp.223–238. Berlin: Springer.

27. Zhang C, Li S, Xia J, et al. BatchCrypt: efficient homomorphic encryption for cross-silo federated learning. In: Proceedings of the 2020 USENIX annual technical conference, 2020, pp.493–506, https://cse.hkust.edu.hk/~weiwa/papers/batchcrypt-atc20.pdf

28. Li J, Kuang X, Lin S, et al. Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. Inform Sci 2020; 526: 166–179.

29. Blatt M, Gusev A, Polyakov Y, et al. Secure large-scale genome-wide association studies using homomorphic encryption. Proc Natl Acad Sci U S A 2020; 117(21): 11608–11613.

30. Nguyen L, Safavi-Naini R and Kurosawa K. Verifiable shuffles: a formal model and a paillier-based efficient construction with provable security. In: Proceedings of the international conference on applied cryptography and network security, Yellow Mountains, China, 8–11 June 2004, pp.61–75. Berlin: Springer.

31. He X, Liao L, Zhang H, et al. Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, Perth, WA, Australia, 3–7 April 2017, pp.173–182. New York: ACM.