Federated Latent Dirichlet Allocation for User Preference Mining

Xing Wu[1], Yushun Fan[1,*], Jia Zhang[2] and Zhenfeng Gao[3]

[1]*Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing, China*
[2]*Department of Computer Science, Southern Methodist University, Dallas, TX, USA*
[3]*Sangfor Technologies Inc., Shenzhen, China*
E-mail: gzf@sangfor.com.cn
[*]*Corresponding Author*

Received 07 May 2022; Accepted 19 June 2023;
Publication 24 October 2023

Abstract

In the field of Web services computing, a recent demand trend is to mine user preferences based on user requirements when creating Web service compositions, in order to meet comprehensive and ever evolving user needs. Machine learning methods such as the latent Dirichlet allocation (LDA) have been applied for user preference mining. However, training a high-quality LDA model typically requires large amounts of data. With the prevalence of government regulations and laws and the enhancement of people's awareness of privacy protection, the traditional way of collecting user data on a central server is no longer applicable. Therefore, it is necessary to design a privacy-preserving method to train an LDA model without massive collecting or leaking data. In this paper, we present novel federated LDA techniques to learn user preferences in the Web service ecosystem. On the basis of a user-level distributed LDA algorithm, we establish two federated LDA models in charge of two-layer training scenarios: a centralized synchronous federated LDA (CSFed-LDA) for synchronous scenarios and a decentralized

Journal of Web Engineering, Vol. 22_4, 639–678.
doi: 10.13052/jwe1540-9589.2244

asynchronous federated LDA (DAFed-LDA) for asynchronous ones. In the former CSFed-LDA model, an importance-based partially homomorphic encryption (IPHE) technique is developed to protect privacy in an efficient manner. In the latter DAFed-LDA model, blockchain technology is incorporated and a multi-channel-based authority control scheme (MCACS) is designed to enhance data security. Extensive experiments over a real-world dataset ProgrammableWeb.com have demonstrated the model performance, security assurance and training speed of our approach.

Keywords: Web service composition, user preference mining, federated learning, LDA, homomorphic encryption, blockchain.

1 Introduction

With the wide adoption of service-oriented architecture (SOA) and cloud computing [1], the number of Web services published on the Internet has exhibited a rapid growth in recent years [3, 27, 46]. However, individual Web services may not be able to fully meet users' complex functional needs [8]. Therefore, Web service composition (i.e., mashup or workflow), especially customized service composition according to users' specific requirements, plays a key role in the field of Web services computing [9, 17, 31, 36]. In order to make effective and efficient service composition for users, the first and paramount prerequisite is to mine user preferences.

ProgrammableWeb.com[1] represents by far the largest online repository of Web services, consisting of more than 24,400 services as of May 2022. Each service carries metadata such as name, tags and description that elaborates its functionality and features. These Web services can be exploited by third-party service composition developers to create mashups. Let us consider a user mashup request as, "*Search where the local exhibition is.*" None of existing mashups can meet the compound demand. If mining historical requirements can extract the user's preferences of arts and mapping, then we can pick up an art service "*ArtFacts*" and a mapping service "*Google Maps*" from ProgrammableWeb.com and compose them into a new mashup, e.g., named "*ArtNearBy.*" In the field of Web services computing, the latent Dirichlet allocation (LDA) [10] probability model has become a popular topic model for user preference mining in recent years [23,47]. In a typical scenario, LDA can

[1] https://www.programmableweb.com

be used to extract the potential preferences from user requirements, based on which service match-making can be performed between users and services, and the resulted services become candidates for service composition [40]. In order for LDA to accurately mine user preferences, though, large quantities of data (e.g., prior user requirements) have to be collected to train a model. As a result, users' raw requirements and latent preferences bear the risk of being leaked during the process.

With the prevalence of government regulations and laws on privacy protection such as GDPR,[2] protecting privacy has become a basic criterion nowadays in machine learning and Web services computing. As a result, it is becoming increasingly more difficult to accurately mine user preferences in the traditional manner. On the one hand, the traditional way of collecting data and feeding data to train models at a central server violates the current regulations and laws. On the other hand, developers cannot attend more optimized models as they are restricted to exchange users' data with other developers. Such "*isolated data islands*" formed due to the strict regulations hold back further development of Web service composition. Recently, federated learning (FL) [43] appears as a possible solution for alleviating these problems, and thus receives high attention.

Since the emergence of the original Google's paper [21], a number of researchers have been trying to break through the "*isolated data islands*." In particular, a collection of machine learning methods have investigated the implementation of federated learning [11, 33]. The basic idea of FL is to keep users' raw data at their local sides, while only transmitting the privacy-preserving intermediate data to a central server for global aggregation and training an optimized global model. Since LDA plays a key role in user preference mining and is considered as one fundamental technique in many Web services computing works [6, 16], we believe studying LDA under FL will be valid and will be a groundbreaking endeavor.

Our intended federated LDA optimization significantly differs from the distributed LDA training by Newman et al. [25] in three features. First, the basic principle in FL is to enhance privacy protection. Both users' privacy (users' raw requirements and latent preferences) and model's privacy (the trained model parameters) must be well protected. Second, communication among multiple developers has to be well planned, because they may reside on different networks with different bandwidths. Third, the partition of the data among different developers is usually heterogeneous. On the one

[2] https://eur-lex.europa.eu/legal-content/EN/TXT

hand, the distribution of dataset sizes may be unbalanced. For example, some developers may receive more user requirements than others. On the other hand, the data types may be different among developers. For example, some developers may only receive user requirements under specific domain. Therefore, it is necessary to discuss the effect of data heterogeneity on FL.

As to privacy protection, differential privacy (DP) [15] and homomorphic encryption (HE) [4, 28] are the two most commonly used types of techniques in FL. The former DP technique perturbs the original data by delicately injecting noise, so as to ensure a certain level of privacy protection. However, this method usually has to sacrifice model performance for privacy protection. The latter HE technique allows users' private data to be encrypted, so that they become invisible to the central server during the process of data transmission and aggregation. Afterwards, the aggregated data can be decrypted at developers' sites with private keys. Although HE could ensure decent model performance, the encrypting and decrypting processes are based on large integer calculation and thus can easily result in a computational bottleneck.

With the rise of Bitcoin [24], in recent years the blockchain technique [35] has become another choice in FL to replace the central server as the medium of intermediate data [5, 48]. Instead of incorporating an untrustworthy data collector, blockchain is applied as the data medium. Developers participating in FL can take the role of miner (in public chains such as Ethereum[3]), or peer (in permissioned chains such as Hyperledger Fabric[4]). In this way, the authority control mechanism of blockchain can be utilized to enhance data security.

In this research, we have explored LDA training conducted under a federated learning setting, to mine user preferences in a Web service ecosystem in favor of privacy and security. On the basis of a user-level distributed LDA model, two implementation prototypes are constructed to cover different training scenarios. At synchronous scenarios, we propose a centralized synchronous federated latent Dirichlet allocation (CSFed-LDA) model. A centralized computing server takes the role of a synchronous clock signal controller and an intermediate data aggregator. Each developer only has to send encrypted data to the central server. Considering the ubiquitous phenomenon where the computation time is often inconsistent among developers, we propose another decentralized asynchronous federated

[3] https://www.ethereum.org

[4] https://hyperledger.org/projects/fabric

latent Dirichlet allocation (DAFed-LDA). It is a decentralized federated learning prototype based on blockchain technology to mine user preferences in an asynchronous training manner. Since the network connections may be slow and expensive for some developers, which may result in higher communication cost [21], we have designed multi-step training strategies to reduce the communication cost.

In summary, the major contributions of this paper are three-fold.

(1) As one of the pioneering researches on federated LDA, we have investigated user preference mining in the field of Web services computing under federated learning setting. To the best of our knowledge, these are the first FL-based LDA implementation prototypes, i.e., CSFed-LDA and DAFed-LDA, towards different training scenarios.
(2) We have extended and optimized existing techniques and put forward CSFed-LDA and DAFed-LDA. In the former technique, we propose an innovative importance-based partially homomorphic encryption (IPHE) technique to solve the computational bottleneck problem existing in the original HE. In the latter technique, blockchain technology is incorporated to replace the role of central server in a traditional FL structure, and based on the basic framework of Hyperledger Fabric (HF), an innovative multi-channel access control scheme (MCACS) is developed to further enhance data security.
(3) Over the real-world dataset ProgrammableWeb.com, we conducted extensive experiments to evaluate the model performance, security, and training speed under various settings. The experimental results have proven the effectiveness of our techniques.

Although our research is conducted on the user preference mining area. Our LDA algorithms can also be easily implemented to other service computing areas like service representation and service recommendation. Our overall federated learning framework and the ideas behind it can also be applied to more machine learning algorithms.

The remainder of this paper is organized as follows. Section 2 briefly explains how to apply some basic techniques to build the foundation of our technique. Section 3 discusses the related work. Section 4 formulates our research problems. A user-level distributed LDA framework is introduced in Section 5. Our CSFed-LDA and DAFed-LDA models are described in Sections 6 and 7, respectively. Experimental results and analyses are presented in Section 8. Section 9 summarizes this paper and puts forward our future work.

2 Preliminaries

In this section, we explain how we apply some basic techniques to build the foundation for this research: LDA, federated learning, homomorphic encryption, and blockchain.

2.1 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is an implicit topic analysis model over collections of discrete text documents, based on machine learning and statistics [10]. Treating a collection of user requirements as text corpora, the main idea of applying LDA for user preference mining is that, the occurrence of a word in a specific user requirement is determined by the latent preference distribution of the user. Thus, a three-tier "user-preference-word" structure is formed. In order to extract the latent relationships using the LDA model, we have to infer the posterior distributions, including user-preference distribution and preference-word distribution.

2.2 Federated Learning

Federated learning (FL) is a distributed machine learning method where data owners collaboratively train a model, in which process users' raw data are locally retained and only the privacy-protected intermediate data are exchanged with each other. Taking into account the difference of data distribution in the feature and sample space, FL can be classified into three categories: horizontal FL, vertical FL, and federated transfer learning [43]. In the context of user preference mining, the same word appearing in different user requirements is always considered to reflect the same preference (e.g., the word "*exhibition*" is always related to preference "*art*"). This assumption means users' raw requirements distributed on each developer's site all share the same feature space. In this sense, our research to investigate federated LDA can be categorized as a horizontal FL research.

2.3 Homomorphic Encryption Application

Homomorphic encryption (HE) allows private data to be encrypted before it is released to the central server. Any third-party can aggregate the encrypted data without decrypting it in advance. For any two elements $m_1, m_2 \in R$ (R denotes the plaintext space as a ring), we can establish the following two

equations for fully HE [4]:

$$D(E(m_1) + E(m_2)) = m_1 + m_2 \tag{1}$$

$$D(E(m_1) \times E(m_2)) = m_1 \times m_2 \tag{2}$$

where E is the encryption algorithm, D is the decryption algorithm, $+$ and $\times$ denote the normal addition and multiplication, respectively.

Note that the ciphertext–ciphertext multiplication is not needed in this paper. Therefore, HE applied in our research is additively HE [28] and mainly consists of the following five functions:

- $KeyGen \rightarrow (pk, sk)$: Generate public key pk and secret key sk.
- $Enc(m, pk) \rightarrow c$: Encrypt plaintext m to ciphertext c with public key pk.
- $Dec(c, sk) \rightarrow m$: Decrypt ciphertext c to plaintext m with secret key sk.
- $Add(c_1, c_2, pk) \rightarrow c_a$: Add ciphertexts c_1 and c_2 with pk to calculate the ciphertext of plaintext addition c_a.
- $DecAdd(c_a, sk) \rightarrow m_a$: Decrypt c_a with sk to get the addition of plaintexts m_a.

2.4 Blockchain

A typical public blockchain system usually consists of multiple nodes that do not fully trust each other (e.g., Bitcoin and Ethereum) [14]. These nodes maintain a set of shared, global states and perform transactions to modify these states.

Permissioned blockchain (also called consortium blockchain) allows only authorized organizations to be involved in a blockchain system. That means the nodes can trust each other, so that resource-consuming consensus algorithms (e.g., proof-of-work [24]) are not needed. Therefore, compared with public blochchain, permissioned blockchain requires fewer resources and is able to reach smaller transaction latency and higher throughput. One of the typical permissioned blockchains is Hyperledger Fabric (HF), which supports fine-grained authority control.

Based on the typical assumption of honesty in horizontal FL, the developers participating in FL can form a consortium. Naturally, permissioned blockchain is more suitable in our research. Thus, HF is selected as the underlying blockchain structure of our DAFed-LDA model.

3 Related Work

Our work is related to two categorizes of literature: LDA in user preference mining and federated learning.

3.1 LDA in User Preference Mining

Latent Dirichlet allocation (LDA) [10] is one of the most widely adopted topic models in the area of user preference mining. [22] proposes a probabilistic service discovery approach, using the LDA probabilistic model to extract potential preferences between services and user requirements, and to perform service matching based on these potential preferences. As a kind of temporal LDA, DPDQ [47] is developed to extract user preferences and mine their changing patterns over time. [45] proposes SR-LDA, incorporating users' perceptions into service profiles to form comprehensive service representations. [25] proposes a distributed and parallel LDA training prototype. WrapLDA [12] and LightLDA [44] both optimize the distributed scheme and improve the performance in large-scale distributed applications.

Those works either collect data and train models in a central server, or implement distributed LDA in the absence of a privacy protection mechanism. Both ways are becoming inappropriate nowadays with the prevalence of government regulations and laws on privacy protection (e.g., GDPR). Different from them, we have investigated LDA training under a federated learning setting. In our models, both users' privacy and model's privacy are protected.

3.2 Federated Learning

From the inception of Google's paper [21], federated learning (FL), as a possible solution for training collaborative machine learning model while taking privacy protection into consideration, has attracted significant momentum in recent years. The goal is to design a secure and efficient privacy-preserving technique, while maintaining the accuracy of the algorithm.

Differential privacy (DP) [15] is a popular privacy-preserving technique, which perturbs the original data by delicately injecting noise. [42] proposes a secure multiparty computation protocol HybridAlpha, which combines DP with functional encryption. [37] designs a hybrid mechanism that works for both categorical and numerical data. [29] develops a private convolutional deep belief network (pCDBN) to perturb the energy-based objective functions of traditional CDBNs, rather than their results. However, DP-based methods lead to a decrease in model accuracy because of the noise injected.

Homomorphic encryption (HE) [4, 28] is another popular privacy-preserving technique, which encrypts original data before data is sent to the central server. FedMF [11] employs distributed machine learning and HE schemes to implement secure matrix factorization. [24] combines HE and Bayesian neural networks to protect both users' privacy and model's privacy. However, HE-based methods bring extra encryption computation cost and result in the decline of efficiency. In this paper, to reduce computation cost, we propose importance-based partial homomorphic encryption (IPHE) to identify and encrypt data which has the most critical impact on model performance.

Some research works also consider incorporating blockchain into FL. The key strategy is to use blockchain to replace the centralized aggregator. In blockFL [20], updated data will be aggregated into a block after a round of PoW consensus. [32] designs an FL aggregator BAFFLE and deploys it under the framework of Ethereum. In these works, high communication cost during the consensus process becomes the bottleneck hindering the development of FL. To the best of our knowledge, we are the first to incorporate permissioned blockchain into FL, and we propose a multi-channel-based authority control scheme (MCACS) to realize fine-grained management of data and enhance data security.

While various machine learning algorithms, such as neural networks [21] and matrix factorization [11], have been combined with FL, LDA is still under-investigated in FL. The most related work is LDP-FedLDA [39], which applies local differential privacy to protect privacy in the research of federated LDA. Compared with that work, the strength of our models is that model performance is not sacrificed.

4 Problem Formulation

In this section, we first define two important roles related to user preference mining, namely, users and developers. Afterwards, we formulate the problems of federated learning including federated modeling subjects, training scenarios and privacy protecting objectives.

Users: Users are the producers of data. In order to obtain customized Web service compositions to meet their business needs, users put forward requirement description documents to developers, which constitute the original corpus for LDA training. In this paper, we suppose that there are a total of M users, corresponding to M user requirements.

Developers: (Service composition) developers receive user requirements and mine user preferences based on these requirements. Different users may put forward requirements to different developers. Suppose that there are P developers, then all the users' data are partitioned into P parts $[M_1, M_2, \ldots, M_P]$, each part being kept by a developer independently.

Federated modeling subjects: Due to the restrictions of privacy protection laws and regulations, developers cannot directly exchange each other's original data. Therefore, they mutually hope to train a federated LDA model collaboratively. Unlike regular federated learning, such as Google's user research, which happens at a large number of end users' terminals [21], the subjects of federated modeling in our research are software/mashup developers collaborating in a software project. Thus, federated learning carried out in our context is similar to that among enterprises [13, 30].

Training scenarios: This paper considers two types of training scenario: synchronous and asynchronous training. Firstly, we propose CSFed-LDA aimed at the synchronous training scenario, which is typically considered in the classical federated learning and user preference mining area. One step further, we consider a ubiquitous phenomenon that the training speeds are inconsistent among developers, because of unbalanced dataset sizes, computation powers and so on. If a synchronous training strategy is adopted under this scenario, the developer with the slowest training speed will become the bottleneck due to the barrel effect. To address such an issue, we propose DAFed-LDA to perform asynchronous LDA training. Note that both two models are technical expansions of the user-level distributed LDA.

Privacy protecting objectives: Drawing on the typical assumption in horizontal federated learning research [11, 21], we make an assumption that both the server incorporated in CSFed-LDA and the developers are honest-but-curious. On the premise of this assumption, the attacker (could be the server or developer) may attack for users' privacy [30] and model's privacy [41] while honestly performing federated training operations. We thus make mathematical definitions to these two kinds of privacy and introduce corresponding threat models as follows.

Definition 1 (users' privacy): If the probability that the attacker reconstructs at least τ users' raw information satisfies the following formula, then we say the privacy protecting mechanism satisfies users' privacy (τ, ϵ)-UP:

$$Pr(r \geq \tau) \leq e^{-\epsilon} \tag{3}$$

where r is the proportion of correctly reconstruction information.

The threat model that leads to users' privacy disclosure is reconstruction attack. The smaller τ and the larger ϵ, the stronger the level of users' privacy protection represented by (τ, ϵ)-UP.

Definition 2 (model's privacy): If the model performance of the real model $Performance(M_r)$ and that of the attacker's inversed model $Performance(M_{in})$ satisfy the following formula, then we say the privacy protecting mechanism satisfies model's privacy (η)-MP:

$$Performance(M_r) \geq e^{\eta} Performance(M_{in}). \tag{4}$$

The threat model that leads to model privacy disclosure is model inversion attack. One consequence is that the attacker can inverse a similar model without contributing data. η represents the gap between the inversed model and the real model. A high η means the attacker could not inverse an effective model, so that model's privacy can be well protected. Note that since perplexity is used to evaluate model performance in this paper, which is a negative index, the formula (4) can be adjusted to formula (5).

$$Perplexity(M_{in}) \geq e^{\eta} Perplexity(M_r) \tag{5}$$

In CSFed-LDA, the server may launch both reconstruction attack and model inversion attack. In DAFed-LDA, we do not have to guard against attacks from the server, since blockchain is incorporated to replace the server as the data medium. However, some developers may still use the immediate data on blockchain to attack for users' privacy and model's privacy. In this paper, the technique to protect users' privacy is the user-level distributed LDA. As for model's privacy protection, CSFed-LDA uses IPHE, while DAFed-LDA relies on MCACS.

5 User-level Distributed LDA Framework

5.1 Overall Framework

As the foundation of federated LDA, we first construct a user-level distributed LDA framework. The notations related to LDA application are listed in Table 1.

Gibbs sampling (GS) [18] is used in this paper to train the user-level distributed LDA. For each preference k, developer p can calculate the local

Table 1 Notations and definitions

Identifier	Meaning
P	Number of developers
M	Number of user requirements
L	Number of words in each user requirement
K	Number of preferences
w	$M \times L$ user-word vector
V	Length of word vocabulary set
z	$M \times L$ word-preference assignment vector
N	$V \times K$ word-preference counting matrix
θ	User-preference probability distribution matrix
Φ	Preference-word probability distribution matrix
α	Hyper parameter $\theta_i \approx Dirichlet(\alpha)$
β	Hyper parameter $\Phi_k \approx Dirichlet(\beta)$

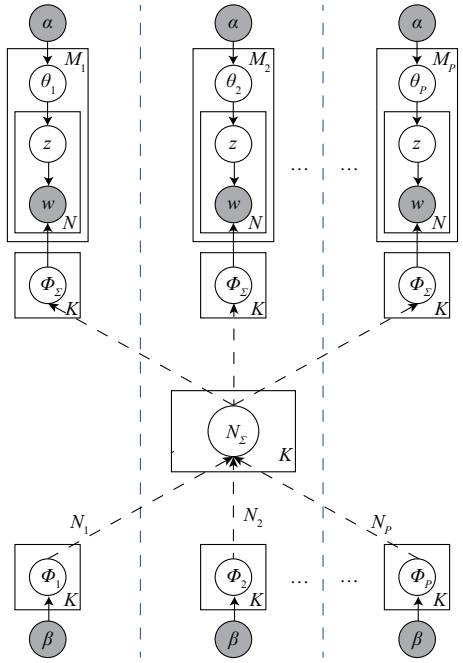


Figure 1 User-level distributed LDA. At every iteration, each developer renews local θ_p and Φ_p. Then preserve θ_p at local site, aggregate N_p to get $N_{\sum}$, and calculate $\Phi_{\sum}$ for the next iteration until model convergence.

user-preference distribution $\theta_{p,i}^k$ for the ith user:

$$\theta_{p,i}^k = \frac{N_{p,i}^k + \alpha}{\sum_{s=1}^{K}(N_{p,i}^s + \alpha)} \tag{6}$$

and preference-word distribution $\Phi_{p,w_{pij}}^k$ for the jth word in the ith user requirement:

$$\Phi_{p,w_{pij}}^k = \frac{N_{p,w_{pij}}^k + \beta}{\sum_{f=1}^{V}(N_{p,f}^k + \beta)} \tag{7}$$

where w_{pij} is the jth word in ith user requirement at pth developer, $N_{p,i}^k$ is the count of preference k in user requirement i at the pth developer, and $N_{p,f}^k$ is the count of the fth word in vocabulary assigned with preference k at the pth developer.

Note that θ_p represents users' latent preferences. It contains users' private information and should be preserved at local sites. In contrast, Φ represents the general latent features of words. It is the optimized objective of user-level distributed LDA and thus should be globally aggregated. Aggregate $N_{p,f}^k$ as Equation (8):

$$N_{\sum,f}^k = \sum_{p=1}^{P} N_{p,f}^k, f \in [1, V] \tag{8}$$

where $N_{\sum,f}^k$ is the total aggregation of the fth word in vocabulary assigned with preference k for all developers. The globally updated $\Phi_{\sum,w_{pij}}^k$ can be calculated at each developer's site as follows:

$$\Phi_{\sum,w_{pij}}^k = \frac{N_{\sum,w_{pij}}^k + \beta}{\sum_{f=1}^{V}(N_{\sum,f}^k + \beta)}. \tag{9}$$

Exclude z_{pij} and multiply Equation (6) by Equation (7), then the conditional distribution for GS becomes:

$$\begin{aligned} p(z_{pij} = k|\cdot) &= \theta_{p,i}^{k,\neg pij} \cdot \Phi_{\sum,w_{pij}}^{k,\neg pij} \\ &= \frac{N_{p,i}^{k,\neg pij} + \alpha}{\sum_{s=1}^{K}(N_{p,i}^{s,\neg pij} + \alpha)} \frac{N_{\sum,w_{pij}}^{k,\neg pij} + \beta}{\sum_{f=1}^{V}(N_{\sum,f}^{k,\neg pij} + \beta)} \end{aligned} \tag{10}$$

where z_{pij} is the preference assigned to w_{pij}; $\neg pij$ means the current z_{pij} is not taken into consideration when calculating θ, Φ and N.

After the GS reaches a convergence state, we can obtain the final updated *local* user-preference distribution matrix θ_p and *global* preference-word distribution matrix $\Phi_{\sum}$. θ_p is the extracted preferences of historical user requirements. Based on the trained $\Phi_{\sum}$, we can model a new user and extract new user's preferences.

Figure 1 depicts the calculation process of user-level distributed LDA. It is worth noting that the core idea of the user-level distributed LDA borrows from [25]. Our framework, however, differs from [25]. Firstly, the global aggregation in Equation (8) could be performed at either server site or developer site according to a centralized or decentralized setting. More importantly, our framework could protect users' privacy well against reconstruction attack which will be discussed below.

5.2 Analysis of Reconstruction Attack

In this section, we will prove that the user-level distributed LDA can achieve a high level of user privacy protection. Suppose that the attacker has received word-preference counting matrix N_p from a developer, which gathers a collection of words $[n_1 \times w_1, \ldots, n_v \times w_v]$ from m users, where the jth word w_j appears n_j times. The length of the ith user's requirement is l_i.

Under the proposed structure, n_j and w_j $j \in [1, v]$ are visible to the attacker, while $l_i, i \in [1, m]$ are invisible. Utilizing the visible information, the probability of fully reconstructing the raw requirements of all users for the attacker is:

$$p(r = 1) = \frac{1}{\prod_{j=1}^{v} C_{m+n_j-1}^{m-1}} \tag{11}$$

where C is for *combination* computation, r is the proportion of correctly reconstructed words. It is a quite small number actually. Assuming $v = 20$ and $n_i = 20, \forall i \in [1, v]$, even if $m = 2$, $p(r = 1)$ is still less than 10^{-9}, corresponding to (1, 20.72)-UP.

Here we discuss the probability of partially reconstructing the raw requirements one step further. To simplify the calculation, assume the average length of user's requirement is l, $n_j = 1, \forall j \in [1, v]$. The probability of correctly reconstructing at least τ words for all users is:

$$p(r \geq \tau) = \frac{\sum_{j=\tau ml}^{ml} C_{ml}^{j} (m-1)^{ml-j}}{m^{ml}}. \tag{12}$$

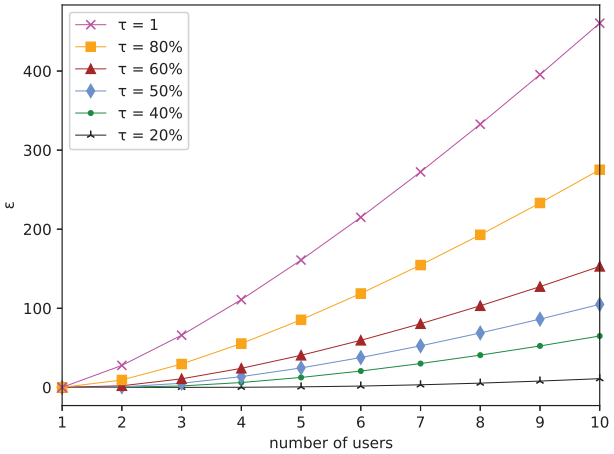


Figure 2 The changing trend of ϵ as the number of users increases with different τ. The larger the value of ϵ, the more difficult it is to reconstruct at least τ for all users.

Assume $l = 20, m = 10$, then $p(r \geq 20\%) = 0.0003, p(r \geq 50\%) = 5.9 \times 10^{-31}$, corresponding to (0.2, 8.11)-UP and (0.5, 71.38)-UP respectively. We can draw from Figure 2 that it is even more impossible to reconstruct raw requirements as the number of users increases. Since the collection of users is usually large in real scenarios, the real probability of reconstructing user's raw requirements is even far less than the above calculation results. Therefore, we believe users' privacy can be well protected through our user-level distributed LDA mechanism.

6 Centralized Synchronous Federated Latent Dirichlet Allocation

Based on the user-level distributed LDA, the key issue of federated LDA is to design a privacy-preserving and efficient transmission and aggregation mechanism for counting matrix N_p to against model inversion attack. Considering the typical LDA training at synchronous scenarios, We propose a centralized synchronous federated latent Dirichlet allocation (CSFed-LDA) model as one solution to tackle this key issue. To protect privacy, we incorporate homomorphic encryption, based on which we propose importance-based partially homomorphic encryption (IPHE) to further improve training efficiency. We also adopt a sparse representation of N_p and propose a multi-step training strategy to reduce the communication cost in our model.

6.1 Overall Structure and Workflow

The overall structure of our CSFed-LDA is illustrated in Figure 3. It carries a workflow comprising seven major steps as described below.

1. Key generation: The homomorphic encryption introduced in Section 2 is based on asymmetric encryption technique [34], where a pair of public keys *pk* and secret keys *sk* is required. Firstly, we generate *pk* and *sk*, which can be carried out by one of the developers (e.g., the developer with the largest dataset size or the developer who launches federated learning). *pk* can be shared among developers including the server, while *sk* must be protected against the server.
2. Vocabulary alignment: Different collections of user requirements may consist of different word vocabularies. Every local vocabulary is aligned with those of other developers to form a global vocabulary.
3. Parameter initialization: This step initializes the hyper parameters of LDA, α, β, K, maximum iteration steps and so on. Parameters are kept consistent between the server and developers.
4. Model initialization: This step assigns a random preference to each word at every developer's site as an initial state, and calculates the initial user-preference distribution θ_p and word-preference counting matrix N_p. N_p is encrypted and then sent to the server.

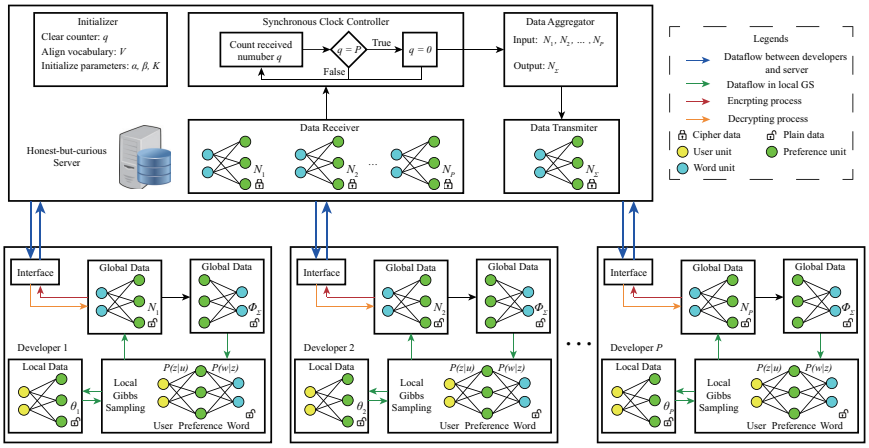


Figure 3 The structure of CSFed-LDA. There are five modules at the server site, *initializer* for preparation work, *data receiver*, *synchronous clock controller*, *data aggregator*, and *data transmitter* for the global aggregation process.

5. Global aggregation: The process of global data aggregation is performed at the central server. Four modules are involved within this process. *Data receiver* collects the encrypted N_p from the participating developers. *Synchronous clock controller* counts the total received N_p to check whether all developers have completed local GS and sent the encrypted data. After receiving all N_p, the *data aggregator* uses public key pk to calculate the aggregated $N_{\sum}$ without decrypting it. Finally, the *data transmitter* transmits $N_{\sum}$ to every developer.
6. Local GS: After receiving $N_{\sum}$, every developer uses sk to decrypt it, and calculates $\Phi_{\sum}$ as defined in Equation (9). θ_p and $\Phi_{\sum}$ are used to renew the conditional distribution as Equation (10) defines and samples all the preferences for every word. After completing all sampling, each developer calculates the local word-preference counting matrix N_p, encrypts it and transmits it to the server.
7. This step repeats step 5 and step 6 until the model converges. Each developer saves the final updated θ_p and $\Phi_{\sum}$.

Note that steps 1–3 are the preparation work for CSFed-LDA. The module of *initializer* in the central server can serve as the intermediary to help with the process of vocabulary alignment and parameter initialization. Steps 4–7 elaborate the training process, whose details are described in Algorithm 1.

Considering the effect of communication cost and training speed, we have designed single-step and multi-step training strategies. The single-step training completes one iteration of local GS at the developer's site before aggregating data, and the multi-step training completes multiple iterations of local GS at the developer's site before aggregating data. Compared with the single-step strategy, our multi-step training strategy can significantly reduce communication cost. The reason is that the immediate parameters need to be shared after every iteration in the single-step strategy, but in our multi-step strategy, immediate parameters are only shared once after multi-iterations. With the set maximum iterations, fewer communications are needed for the multi-step strategy. The multi-step training strategy can be applied to the occasions when communication cost is the major cost and faster training speed is necessary. In the later experiment section, we will further investigate the effect of the number of steps on model performance and training speed.

6.2 Importance-based Partially Homomorphic Encryption

Before global aggregation, a developer needs to encrypt local N_p and send it to the server. N_p is a counting matrix of size $V \times K$. In order to facilitate

Algorithm 1: Training process of CSFed-LDA

Input: LDA hyper parameters α β K , local user-word vector $\{w_p\}$, maximum iterative number $maxIter$;

Output: Global preference-word distribution $\Phi_{\sum}$, local user-preference distribution $\{\theta_p\}$;

initialization;
randomly initialize $\{z_p\}$, count the initial $\{N_p\}$, encrypt $\{N_p\}$ and send it to the server;
set counter $q = 0$;
for *iteration steps = 1,2, …,* $maxIter$ **do**
    global aggregation;
    while *counter* $q < P$ **do**
        if *receiving* N_p *from any developer* **then**
            $q \leftarrow q + 1$
        end
    end
    $N_{\sum} \leftarrow$ aggregate $N_1, N_2, \ldots, N_P$;
    transmit $N_{\sum}$ to every developer;
    $q = 0$;
    local sampling;
    for *developer-id p = 1,2, …, P* **do**
        decrypt $N_{\sum}$ and calculate $\Phi_{\sum}$;
        for *user-id i = 1,2, …,* M_p **do**
            for *word-id j = 1,2, …,* L_p^i **do**
                sample a new preference $z_{p,i}^j$ for word $w_{p,i}^j$;
            end
            update θ_p;
            calculate N_p;
        end
        encrypt N_p and send it to the server;
    end
end
return $\{\theta_p\}$, $\Phi_{\sum}$;

encryption and transmission, the original matrix is replaced by $V \times K$ triplets $T_p = \{(f, k, N_{p,f}^k)\}$. At the developer's site, $\{(f, k, N_{p,f}^k)\}$ will be encrypted and sent to the server. At the server's site, similar triplets $T_{\sum} = \{(f, k, N_{\sum,f}^k)\}$ are maintained, and the process of aggregating N_p is actually the summation of $N_{p,f}^k$ with the same f and k from different developers, where $N_{p,f}^k$ should be encrypted in advance. The homomorphic encryption introduced in Section 2 will help attain our objective. In our framework, we choose Paillier encryption [28] as the encryption and decryption technique.

Note that the Paillier encryption is a probabilistic encryption schema based on composite residuosity problem, where large integer calculation is necessary. As a result, homomorphic encryption can easily become the computational bottleneck during CSFed-LDA training. In this paper, we propose an innovative importance-based partially homomorphic encryption (IPHE) mechanism to accelerate the encryption and decryption process.

Definition 3. Let δ be a percentage; if we only do Paillier encryption to δ important data, then we say that δ-IPHE is applied to this FL training process.

We shall identify important data first. Recall the objective of this research is to mine users' preferences. Note that for a specific preference, some words bear high frequency. For example, for a specific preference *art*, related words like *art* and *exhibition* are considered high-frequency words. The ability to correctly identify the classification of high-frequency words plays an important role in mining users' preferences. Therefore, high-frequency words are important data in the scenario of federated LDA.

In our CSFed-LDA model with δ-IPHE, we sort the words by their frequency in descending order, and do homomorphic encryption to the top δ words. Since only partial data are encrypted, the communication process is significantly accelerated. In the later section, theoretical analysis combined with experimental results will prove that even though only partial parameters are encrypted, model privacy can still be effectively protected.

6.3 Sparse Representation

Note that N_p may be a sparse matrix due to two reasons: (i) the length of local word vocabulary $\leq V$, and (ii) some low-frequency words only appear few times. For example, if $K = 50$ and $word_f$ appears five times, then even the minimum sparsity of $word_f$ can be up to 90%.

To reduce communication time, we incorporate sparse representation as an extended optimization module of δ-IPHE. The most important δ words would be encrypted before transmitted to protect model privacy. As for the remaining $1 - \delta$ unencrypted words, we apply sparse representation and transmission. That means for the $1 - \delta$ unencrypted words, $\{(f, k, N_{p,f}^k)\}$ will be sent to the server only if $N_{p,f}^k > 0$ and only received $\{(f, k, N_{p,f}^k)\}$ will be aggregated, while other unreceived $\{(f, k, N_{p,f}^k)\}$ mean $N_{p,f}^k = 0$ and have no influence on aggregation result.

With δ-IPHE and sparse representation, the generative process of transmission data is summarized in Algorithm 2.

Algorithm 2: Transmission data generative process

Input: Encryption proportion δ, descending ordered vocabulary, vocabulary length V, local $\{N_p\}$;
Output: The collection of transmission data $\{U_p\}$;

```
for developer id = 1, 2, ..., P do
    for word-index f = 1,2, ..., V do
        for preference-id k = 1,2, ..., K do
            if f ≤ δV then
                encrypt N^k_{p,f}
                add (f, k, N^k_{p,f}) to U_p
            end
            else
                if N^k_{p,f} ≠ 0 then
                    add (f, k, N^k_{p,f}) to U_p
                end
            end
        end
    end
end
return {U_p}
```

6.4 Analysis of Model Inversion Attack

Suppose that a developer has obtained correct word-preference distribution φ. Then for a new user with requirement $[n_1 \times w_1, \ldots n_v \times w_v]$, the real user-preference distribution θ_r is as follows:

$$\theta_r = \frac{1}{L} \sum_{j=1}^{v} n_j \cdot \varphi_j \tag{13}$$

where L is the length of user requirement.

Assume the server has launched model inversion attack. The technique to protect model's privacy in our work is δ-IPHE. If we only do homomorphic encryption to word w_f, then the real word-preference distribution for w_f becomes invisible to the server. One strategy is to assign equal preference $\kappa = [1/k, \ldots, 1/k]$ to w_f. Then the inversed user-preference distribution is:

$$\theta_{in} = \frac{1}{L} \left(\sum_{j \neq f} n_j \varphi_j + n_f \kappa \right). \tag{14}$$

We use function $f(\theta_r, \theta_{in})$ to calculate the error ζ between the real distribution and inversed one. $f(\cdot)$ can be the Euclidean metric. Then,

$$\zeta = f(\theta_r, \theta_{in}) = |\theta_r - \theta_{in}|_2 = \frac{1}{L} n_f |\varphi_f - \kappa|_2 \tag{15}$$

where $|\cdot|_2$ is 2-norm calculation. Note that we have to determine the encryption data before training and do encryption before every communication. $|\varphi_f - \kappa|_2$ is unpredictable before training. However, we can reasonably assume that the frequency orders in the training dataset and testing dataset are similar, when both dataset sizes are large enough. Therefore, we can find the maximum word frequency in the collaborative training dataset without exchanging raw data using homomorphic encryption. Afterwards, we choose to encrypt the word with the maximum frequency so as to maximize the expectation of ζ.

When $\delta = 1$, we encrypt all data and the server is completely unable to inverse the model. When $\delta = 0$, the server can inverse the real model. We can adjust δ to make the error ζ big enough, so that the model's privacy can be efficiently protected. Since vanilla LDA is an unsupervised machine learning model, we cannot judge whether the correct word-preference distribution is obtained. Therefore, the major index to compare two models is perplexity [10] in the experiments, which will be discussed later. We will also calculate η defined in Definition 2 based on the real dataset.

7 Decentralized Asynchronous Federated Latent Dirichlet Allocation

In this section, we propose a decentralized asynchronous federated latent Dirichlet allocation (DAFed-LDA) model to perform asynchronous LDA training. Again we adopt the typical assumption of horizontal FL with *P* honest developers. No central server is needed, since we incorporate blockchain as the data carrier for transmitting word-preference counting matrix N_p, and Hyperledger Fabric (HF) is selected in our model.

7.1 Blockchain Perspective

From a blockchain perspective, the structure of DAFed-LDA is illustrated in Figure 4. There are three main types of nodes in HF: (1) the *certification authorization (CA) node* is responsible for issuing certificates to developers. Only trusted developers will be allowed to participate in FL. CA does not

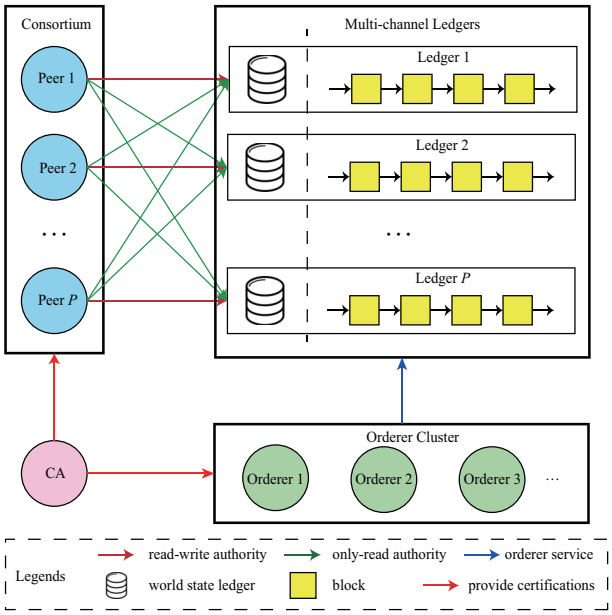


Figure 4 The blockchain perspective of DAFed-LDA.

interfere with the training process. (2) The *peer node* is the key element in our model. It functions as the proxy to connect developers and ledgers. All transmission data are stored at ledgers and each peer maintains a copy. Chaincode defining the basic logic of data transmission is deployed at peers, through which developers can submit the local updated data and download other developers' latest version of data. All participating developers must deploy at least one peer node so that they can maintain the ledgers and run chaincode to make read-write operations on ledgers. (3) The *orderer node* packages the submitted data into a block and transmits the block to the peer nodes. It participates in data transmission but has no authority to modify data. A consensus algorithm (e.g., Raft [26]) runs on the cluster of orderer nodes in order to ensure consistency of distributed ledgers. Orderer nodes can be deployed and run at partial or all developers' sites.

In addition, the channel is one of the core concepts of HF. HF allows multiple channels to be created. Each channel contains a ledger. In this work, based on the basic framework of HF, we develop an innovative multi-channel based authority control scheme (MCACS) to realize fine-grained management of data and further enhance data security. The basic idea of MCACS is

to create *P* channels (corresponding to *P* ledgers) among *P* developers. The ith channel stores the data from the ith developer.

We have developed two ways of implementing read and write authority control in MCACS.

(1) **Data reading authority control based on channel access.** In HF, different peers join a channel to maintain a distributed ledger. In other words, peers excluded by a specific channel have no access to the corresponding ledger. For ease of presentation, we assume that each developer deploys only one peer and a total of P peers are deployed. The channel-peer access matrix CP of size $P \times P$ is built. $CP_{ij} = 1$ representing peer j has the access to channel i. CP_{ii} = 1 and CP is a symmetric matrix. Through special filling of CP, multiple consortia can be set up to fit complex and diverse FL production scenarios.

Definition 4. A collection of developers D can set up a consortium when satisfying the following two conditions:

$$CP_{ij} = 1, \forall i, j \in D$$

$$CP_{ij} = 0, \forall i \in D, j \notin D$$

Figure 5 is a case study with five developers. Local word-preference counting matrix N_p are shared within each consortium. Setting up multiple small consortia is reasonable in real production scenario due to two commercial reasons. First, it may be unfair for developers with large dataset size to endure those with small one to participate in FL with them. Therefore, developers with extremely large (or small) dataset size can build an isolated consortium (e.g., developers 4 and 5 in Figure 5). Second, there may exist

1	1			
1	1	1		
	1	1		
			1	1
			1	1

Figure 5 A case study with five developers. In this case, we set up three small consortia, con_{1-2}, con_{2-3}, con_{4-5}.

conflicts of interest among some developers, so that they cannot share data with each other but are willing to participate in FL with third-party developers. (e.g., developers 1 and 3 in Figure 5, and they both communicate with developer 2).

(2) **Data writing authority control based on endorsement strategy.** In HF, chaincodes have to be instantiated at specific channel before peers can invoke them. We can specify several peers as endorsement ones when instantiating chaincodes. Endorsed peers are granted read-write permissions, while others only have read authority. In MCACS, for a specific channel p, only peer p is endorsed. Therefore, only developer p has the authority to submit local N_p to ledger p which cannot be falsified by other developers. By configuring the appropriate parameters, the latest N_p waiting for aggregation can be packaged into a block. Each new block is linked to its predecessor as FL training is going on. *Ledger World State*, a Level-DB or Couch-DB based database, is adopted by HF to record N_p in the form of key-value pairs. Other developers with read authority can obtain N_p, by querying the latest key-value from ledger p to implement global aggregation while completing local GS.

From a blockchain perspective, the process of setting up a consortium is a five-step workflow: (1) Deploy CA node. (2) Provide legal PKI-based identity certifications to each developer. (3) Deploy orderer nodes and peer nodes. (4) Create multiple channels and involve peers as members for each channel. (5) For each channel, deploy chaincodes at joining peers and select specific endorsement peer as MCACS requires while instantiating chaincodes.

7.2 Federated LDA Perspective

From a federated LDA perspective, the major steps of DAFed-LDA with the consortium built include five consecutive steps. (1) Training preparation: align vocabulary and initiate hyper parameters. (2) Model initialization: assign random preferences as an initial state, calculate the initial user-preference distribution θ_p and word-preference counting matrix N_p. Invoke chaincode and make a transaction proposal to submit the initial N_p to ledger p. N_p will be packaged into the zeroth block (genesis block). (3) Global aggregation: developer p queries the latest $N_{\neg p}$ from other ledgers and aggregates N_p with $N_{\neg p}$ to calculate $N_{\sum}$. (4) Local GS: use θ_p and $\Phi_{\sum}$ to renew the conditional distribution and perform local GS. After finishing sampling for every word, submit the updated N_p to ledger p. (5) Repeat steps 3 and 4 until the model converges. Each developer saves the final updated θ_p and $\Phi_{\sum}$. The training detail can be seen in Algorithm 3.

Algorithm 3: Training process of DAFed-LDA

Input: LDA hyper parameters α β K , local user-word vector $\{w_p\}$, maximum iterative number $maxIter$;

Output: Global preference-word distribution $\Phi_{\sum}$, local user-preference distribution $\{\theta_p\}$;

initialization;
randomly initialize $\{z_p\}$, count the initial $\{N_p\}$, submit the initial N_p to ledger p and create genesis block for ledger p; **for** *iteration steps = 1,2, …,* $maxIter$ **do**
  for *developer-id p = 1,2, …, P* **do**
    global aggregation;
    query the latest $N_{\neg p}$ from other ledgers;
    $N_{\sum} \leftarrow$ aggregate $N_p, N_{\neg p}$;
    sum global $\Phi_{\sum}$ with $N_{\sum}$;
  end
  local sampling;
  for *user-id i = 1,2, …,* M_p **do**
    for *word-id j = 1,2, …,* L_p^i **do**
      sample a new preference $z_{p,i}^j$ for word $w_{p,i}^j$;
    end
    update θ_p;
    calculate N_p;
  end
  send the latest N_p to ledger p;
end
return $\{\theta_p\}$, $\Phi_{\sum}$;

There are three major features for DAFed-LDA to be differentiated from CSFed-LDA.

First, with downloaded $N_{\neg p}$ from other developers, each developer implements global aggregation at a local site, while in CSFed-LDA, the aggregation process is carried out at the server's site.

Second, at asynchronous scenarios, the latest $N_{\neg p}$ downloaded from other developers may be lagging calculation results, since training speeds are inconsistent among developers. For example, at a certain moment, developer p has completed t iterations, but other developers may only have completed $t-1$ iterations or even fewer. Developer p would not wait but use N_p and lagging $N_{\neg p}$ to make global aggregation and then start the next iteration. With this strategy, computing resources could be fully utilized.

Third, Since no central server is included, we do not have to worry about the leakage of model privacy to the server. Instead, the threat comes from the

developers excluded by the consortium, and we will discuss this threat in the next section.

Note that sparse representation of transmission data N_p and multi-step training strategy can be similarly applied in DAFed-LDA to reduce communication cost.

7.3 Analysis of Model Inversion Attack

With MCACS, developers included in the same consortium share parameters with each other, and they are the builders of the trained model. Only when other developers excluded by the consortium try to get model parameters does the model inversion attack occur. However, such an attack cannot obtain any privacy information in the HF based environment. Therefore, model privacy can be well protected through our MCACS.

In the experiments of this research which will be discussed in detail later, we set aside commercial factors and consider all developers are in a consortium.

8 Experiments and Analysis

In this section, we first explain our experimental settings, and then present our series of experimental results on model performance, security, and model training speed.

8.1 Experimental Setting

Dataset: We crawled the metadata of mashups from ProgrammableWeb.com, which is by far the largest online repository of Web services and their reported mashups. Each mashup contains metadata such as name, category and description. In order to make the experimental results more significant, we added category into the text description, and applied word stemming and removed stop words. These mashup descriptions were treated and analyzed as user requirements for our experiments as a typical setting [7, 19, 40]. After the preprocessing process, an overview of our dataset is summarized in Table 2. About 75% user requirements were split as a training set, which were assigned to each developer as local training data. We used the other 25% to validate the trained FL model.

Environment: All the experiments were performed on six computers with CPU i7-6700, five for simulating developers where up to 20 virtual nodes are

Table 2 Dataset from ProgrammableWeb.com

Total # of user requirements	7936
Length # of word vocabulary	13,155
Average # of words in each user requirement	20.08
Average # of words for each word in vocabulary	12.11
Length # of training set	6000
Length # of test set	1936

simulated, and one for simulating the server in CSFed-LDA. To establish the HF structure in DAFed-LDA, we deployed a total of three orderer nodes to run Raft consensus. The HF version is v1.4.

Parameter setting: We set the hyper parameters $\alpha = 50/K$ and $\beta = 0.01$ according to the empirical formula [18], and the maximum number of iterations was set to 1000. Some parameters have a direct effect on model performance and are important to our research, including the encryption ratio δ, the number of preferences K and the number of steps. We'll analysis their effect on model performance, privacy security and training speed through a series of experiments.

Evaluation metrics: Perplexity was adopted in our experiments to evaluate the performance of LDA models. Generally, lower perplexity stands for better generation performance. For M user requirements, the perplexity is described as:

$$perplexity = \exp\left(-\frac{1}{\sum_{i=1}^{M} L_i}\sum_{i=1}^{M}\sum_{w\in D_i} ln\left(\sum_{k=1}^{K} p(w|z_k)p(z_k|D_i)\right)\right) \tag{16}$$

where D_i is the collection of words for the ith user, L_i is the number of documents, w represents the word and z_k represents preference k.

Decentralized settings: Some settings were specifically configured for DAFed-LDA. (1) A random delay was added to every iteration at each developer's site to create an asynchronous training environment. (2) As long as the developer with the fastest training speed had finished the set maximum number of iterations, the training ended.

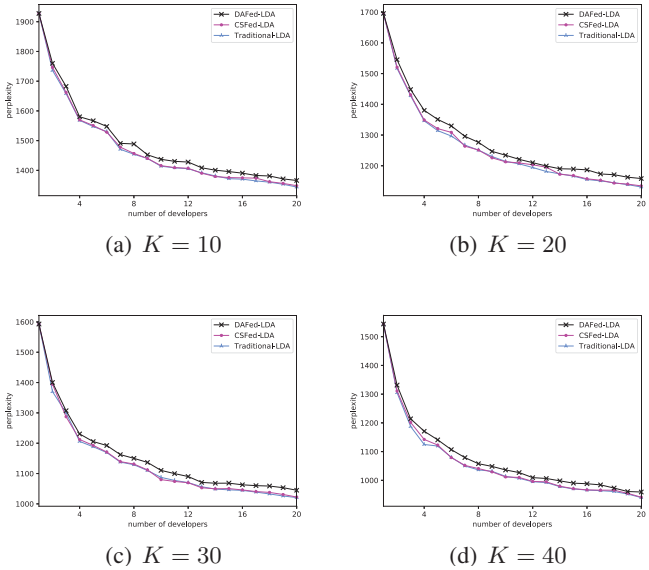
Other default settings: If without special instructions, we set the following configures as default. (1) A total of 20 developers were included in our experiment, and the training set was randomly split evenly. (2) Every result was the average of three repeated experiments.

8.2 Analysis of Model Performance

We designed a collection of experiments to evaluate the performance of our framework.

8.2.1 Impact of Federated Learning

Figure 6 shows the performance comparison of traditional-LDA (collect all developers' data at a device to perform LDA training) and our CSFed-LDA and DAFed-LDA, with a given K ($K = 10$, 20, 30, 40). It can be seen that the perplexity declines significantly as more developers participate in the training process. Therefore, it is meaningful to utilize multi-party data to train the federated model. Another important observation is that CSFed-LDA performs almost as well as the traditional-LDA, which indicates that model performance will not be affected with CSFed-LDA. Although DAFed-LDA shows a slight performance loss, because the results may converge into sub-optimal values under asynchronous scenarios, it still outperforms significantly the stand-alone LDA training (training with only one developer's data at local site). We can see from Figure 6 that the performance of DAFed-LDA with 20 developers is almost equal to that of CSFed-LDA with 18 developers.



(a) $K = 10$ (b) $K = 20$

(c) $K = 30$ (d) $K = 40$

Figure 6 Model performance as the number of developers increases when given different K.

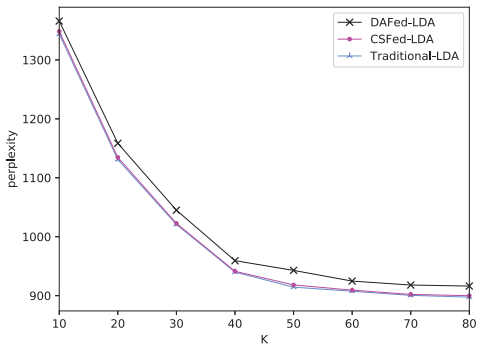


Figure 7 Model performance as K increases.

8.2.2 Impact of *K*

We designed experiments to identify an optimal number of preferences K. Figure 7 shows that with K increasing, the perplexity of all three approaches decreases. It can be found that $K = 40$ leads to the common points of inflection on these curves. When $K > 40$, the perplexity only drops slightly, but we will have to endure linearly increasing time cost as K increases. Therefore, the optimized number of preferences based on this dataset is 40, which is the default setting in the subsequent experiments.

8.2.3 Impact of Data Heterogeneity

We designed experiments to evaluate the effect of data heterogeneity on the performance of federated learning. In the earlier experiments, the total training dataset was equally divided among 20 developers. Such an assignment was labeled as scheme (a). In this experiment, we re-partitioned the training dataset in two new ways as scheme (b) and scheme (c). In scheme (b), we divided the total training dataset to 20 developers in a non-uniform manner instead of equal division, so as to investigate the effect of unbalanced dataset sizes. In scheme (c), we partitioned all the data according to the original category label information, each developer maintaining one or more categories. Scheme (c) aimed to investigate the effect of different data types.

Under the three schemes, the model performance results are illustrated in Table 3. It can be seen that the performance of CSFed-LDA and DAFed-LDA are almost unaffected by the data heterogeneity. The reason is that the data had been fully trained at each developer's site. Even though the data distribution among developers was heterogeneous, the operation of global aggregation could almost eliminate the effect of heterogeneity.

Table 3 Model performance with different dataset assignment schemes

	(a)	(b)	(c)
CSFed-LDA	941.36	940.07	941.71
DAFed-LDA	959.37	961.32	960.03

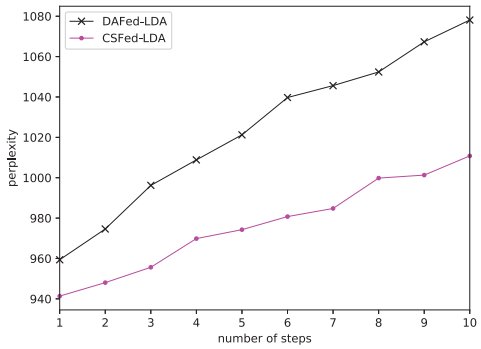


Figure 8 Model performance as the number of steps increases with multi-step strategy.

8.2.4 Impact of Multi-steps

We further designed experiments to study the multi-step training strategy and analyzed the effect of the number of steps on the performance of federated learning. Figure 8 shows that model performance may be affected by the multi-step strategy. This shows that it is a strategy to sacrifice performance for training speed. Taking a further analysis, when steps $= 5$, perplexity increases by 3.5% for CSFed-LDA compared with that in single-step training, and increases by 6.5% for DAFed-LDA. This implies that the multi-step strategy has a stronger impact on DAFed-LDA. Compared with the results in Figure 6(d), it can be seen that the performance of CSFed-LDA when steps $= 8$ is close to that in single-step training with 12 developers. For DAFed-LDA, the closest point appears when steps $= 4$. As a result, if the multi-step strategy is chosen to reduce communication cost so as to accelerate training process, the loss of model performance should be taken into account as well.

8.3 Analysis of Security

We designed a series of experiments to investigate the relationship between the δ value contained in δ-IPHE and the protection level of model's privacy.

Although δ-IPHE could prevent δ important data from being leaked to the central server, the other $1 - \delta$ non-encrypted data still bear the risk of

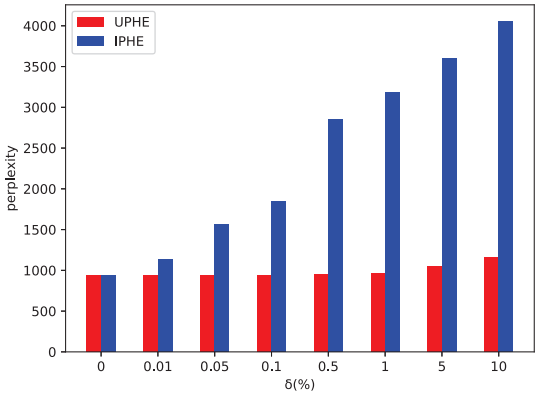


Figure 9 The performance of inversed model as δ increases in UPHE and IPHE. The larger perplexity of inversed model means the better protection of model privacy.

being leaked. Based on these data, the untrustworthy server can inverse a new model, resulting in the leakage of model's privacy. As a contrast, we proposed δ-UPHE in this experiment, where δ unimportant data were encrypted.

Figure 9 shows the performance of an inversed model using the non-encrypted $1-\delta$ data with IPHE and UPHE, respectively. The larger perplexity of the inversed model represents the worse performance of an inversed model, which means the better protection of model privacy. Note that there exists a long-form of IPHE in Figure 9, which represents the large gap between the two inversed models in IPHE and UPHE with the same δ. The server could not inverse a well performing model with low perplexity under the protection of IPHE, while the server could do it under UPHE. Therefore, IPHE protects model privacy far better than UPHE with the same δ. As δ increases, more data are encrypted, and the ability to protect model privacy is enhanced. However, the enhancement for UPHE is not obvious. For example, when δ increases to 0.5% from 0, the perplexity for UPHE only increases by 1.17%, corresponding to (0.01)-MP. At this occasion, the inversed model still has good performance. By contrast, the perplexity increases up by 203% with IPHE when $\delta = 0.5\%$, corresponding to (1.11)-MP. This inversed model has lost its functionality completely. Two conclusions can be drawn from the results. First, the important data have a more significant impact on the model performance. In LDA, high-frequency words can reflect users' preferences better than low-frequency words. Second, even a small δ can protect model privacy well in IPHE. Thus, in our model, we select δ to be 0.5% and δ-IPHE could reach model privacy (1.11)-MP.

8.4 Analysis of Training Speed

Furthermore, we designed experiments to investigate the training speed of CSFed-LDA and DAFed-LDA. Because the entire training cycle is long, we used the average time consumption of each iteration to describe the training speed. The average time consumption mainly comprises local GS time, communication time and encryption time. Table 4 records the time consumption of traditional-LDA, CSFed-LDA and DAFed-LDA. δ was 0.5% in IPHE and the number of steps was 4 in the multi-step strategy.

As to local GS time, since traditional-LDA has to sample every word for the entire dataset, the average time consumption of each iteration is long. By contrast, CSFed-LDA and DAFed-LDA both show higher efficiency in Table 4. The reason is that they inherit the advantage of distributed LDA and accelerate the training process through parallel training.

As to encryption time (including decryption time), Table 4 shows that the encryption time is the training bottleneck of CSFed-LDA with original homomorphic encryption (HE). And our approach uses IPHE to optimize the process. As analyzed in Section 6.3, model privacy can be well protected with $\delta = 0.5\%$. Under this setting, the encryption time drops by more than 95%.

As to communication time, since the encrypted cipher data consume more storage space, CSFed-LDA has to bear the transmission time of large-capacity cipher data during communication. The communication time for DAFed-LDA consists of the time to operate blockchain, including submitting and downloading the block. As shown in Table 4, such time is significantly reduced under a permissioned blockchain structure.

Finally, we studied the effect of a multi-step strategy, where the data were encrypted and transmitted after multiple iterations. The time consumption for encryption and communication was allocated to each iteration. As shown in Table 4, as the number of steps increases, the average time consumption

Table 4 Average time consumption of each iteration

	Setting		Running Time (Seconds)			
	Multi-step	Encryption Methods	Local GS	Encryption	Communication	Total
Traditional-LDA	\	\	139.31	\	\	139.91
CSFed-LDA	False	HE	6.91	1374.21	4.57	1385.69
	False	IPHE		7.03	1.61	15.55
	True	HE		343.56	1.14	351.61
	True	IPHE		1.76	0.40	9.07
DAFed-LDA	False	\		\	3.24	10.15
	True	\		\	0.81	7.72

drops. In the conditions when network connection is limited, the advantage of multi-step strategy will be further reflected.

9 Conclusions and Future Work

In this paper, we have investigated LDA training under an FL setting to mine user preferences in the service ecosystem. This is a new secure way to utilize multi-party data to train an optimized federated model. On the basis of a user-level distributed LDA framework, we present two federated LDA implementation techniques and prototypes towards different training scenarios: CSFed-LDA for synchronous scenarios and DAFed-LDA for asynchronous scenarios. We have proved that our models are secure against reconstruction attack and model inversion attack. Extensive experiments also show the advantages of our models as to model performance and training speed while protecting privacy.

Our future work will focus on the following two aspects. First, we aim to build a federated mashup recommendation system based on the extracted user preferences to create and recommend Web service compositions for users. Second, we plan to extend our federated learning prototypes to other machine learning algorithms, so as to solve more Web services computing problems [2, 38] in a secure way.

Acknowledgements

This research has been partially supported by the National Natural Science Foundation of China (No.62173199). Yushun Fan is the corresponding author.

References

[1] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. Big data and cloud computing: new wine or just new bottles? *Proceedings of the VLDB Endowment*, 2010.

[2] Shereen H Ali, Rana A El-Atier, Khaled M Abo-Al-Ez, and Ahmed I Saleh. A gen-fuzzy based strategy (gfbs) for web service classification. *Wireless Personal Communications*, 113:1917–1953, 2020.

[3] Vasilios Andrikopoulos, Salima Benbernou, and Michael P Papazoglou. On the Evolution of Services. *IEEE Transactions on Software Engineering*, 38(3):609–628, 2012.

[4] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A Reuter, and Martin Strand. A guide to fully homomorphic encryption. *Cryptology ePrint Archive*, 2015.

[5] Sampathkumar Arumugam, Shishir Kumar Shandilya, and Nebojsa Bacanin. Federated learning-based privacy preservation with blockchain assistance in iot 5G heterogeneous networks. *Journal of Web Engineering*, pages 1323–1346, 2022.

[6] B. Bai, Y. Fan, W. Tan, and J. Zhang. SR-LDA: Mining effective representations for generating service ecosystem knowledge maps. In *Proceedings of IEEE International Conference on Services Computing (SCC)*, pages 124–131, 2017.

[7] B. Bai, Y. Fan, W. Tan, and J. Zhang. Dltsr: A deep learning framework for recommendations of long-tail web services. *IEEE Transactions on Services Computing*, 13(1):73–85, 2020.

[8] Kailash Chander Bhardwaj and RK Sharma. Machine learning in efficient and effective web service discovery. *Journal of Web Engineering*, pages 196–214, 2015.

[9] M. Blake and Y. Wei. Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, 14(06):72–75, 2010.

[10] David M Blei, Andrew Y Ng, Michael I Jordan, and John Lafferty. Latent Dirichlet Allocation. *J. Mach. Learn. Res*, 3:993–1022, 2003.

[11] D. Chai, L. Wang, K. Chen, and Q. Yang. Secure federated matrix factorization. *IEEE Intelligent Systems*, 2020.

[12] Jianfei Chen, Kaiwei Li, Jun Zhu, and Wenguang Chen. Warplda: a cache efficient o(1) algorithm for latent dirichlet allocation. *Proceedings of the Vldb Endowment*, 9(10):744–755, 2016.

[13] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 2021.

[14] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7): 1366–1385, 2018.

[15] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[16] Z. Gao, Y. Fan, C. Wu, W. Tan, J. Zhang, Y. Ni, B. Bai, and S. Chen. SeCo-LDA: Mining service co-occurrence topics for composition recommendation. *IEEE Transactions on Services Computing*, 12(3):446–459, 2019.

[17] Zhenfeng Gao, Yushun Fan, Xiu Li, Liang Gu, Cheng Wu, and Jia Zhang. Discovery and analysis about the evolution of service composition patterns. *Journal of Web Engineering*, 18(7):579–626, 2019.

[18] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Supplement 1):5228–5235, 2004.

[19] Y. Hao, Y. Fan, W. Tan, and J. Zhang. Service recommendation based on targeted reconstruction of service descriptions. In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pages 285–292, 2017.

[20] H. Kim, J. Park, M. Bennis, and S. Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2020.

[21] Jakub Konen, H. Brendan Mcmahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. 2016.

[22] C. Li, R. Zhang, J. Huai, X. Guo, and H. Sun. A probabilistic approach for web service discovery. In *Proceedings of IEEE International Conference on Services Computing (SCC)*, pages 49–56, 2013.

[23] X. Liu and I. Fulia. Incorporating user, topic, and service related latent factors into web service recommendation. In *IEEE International Conference on Web Services (ICWS)*, pages 185–192, 2015.

[24] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[25] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10(12):1801–1828, 2009.

[26] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of USENIX Annual Technical Conference*, pages 305–319, 2014.

[27] Abdelaziz Ouadah, Allel Hadjali, Fahima Nader, and Karim Benouaret. Sefap: an efficient approach for ranking skyline web services. *Journal of Ambient Intelligence and Humanized Computing*, 10:709–725, 2019.

[28] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology*, pages 223–238, 1999.

[29] Nhat Hai Phan, Xintao Wu, and Dejing Dou. Preserving differential privacy in convolutional deep belief networks. *Machine Learning*, 106(9-10):1681–1704, 2017.

[30] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

[31] Jin Qi, Bin Xu, Yu Xue, Kun Wang, and Yanfei Sun. Knowledge based differential evolution for cloud computing service composition. *Journal of Ambient Intelligence and Humanized Computing*, 9:565–574, 2018.

[32] Paritosh Ramanan and Kiyoshi Nakayama. Baffle : Blockchain based aggregator free federated learning. 2019.

[33] F. Sattler, S. Wiedemann, K. R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9): 3400–3413, 2020.

[34] Gustavus J. Simmons. Symmetric and asymmetric encryption. *Acm Computing Surveys*, 11(4):305–330, 1979.

[35] CB Sivaparthipan, Bala Anand Muthu, G Fathima, Priyan Malarvizhi Kumar, Mamoun Alazab, and Vicente García Díaz. Blockchain assisted disease identification of covid-19 patients with the help of ida-dnn classifier. *Wireless Personal Communications*, 126(3):2597–2620, 2022.

[36] Hongbing Wang, Bin Zou, Guibing Guo, Danrong Yang, and Jie Zhang. Integrating trust with user preference for effective web service composition. *IEEE Transactions on Services Computing*, 10(4):574–588, 2017.

[37] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu. Collecting and analyzing multidimensional data with local differential privacy. In *Proceedings of IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649, 2019.

[38] Ronghan Wang and Junwei Lu. Qos-aware service discovery and selection management for cloud-edge computing using a hybrid meta-heuristic algorithm in iot. *Wireless Personal Communications*, 126(3):2269–2282, 2022.

[39] Yansheng Wang, Yongxin Tong, and Dingyuan Shi. Federated latent dirichlet allocation: A local differential privacy based framework. In *Proceedings of AAAI*, pages 6283–6290, 2020.

[40] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu. Category-aware api clustering and distributed recommendation for automatic mashup creation. *IEEE Transactions on Services Computing*, 8(5):674–687, 2015.

[41] Peichen Xie, Bingzhe Wu, and Guangyu Sun. Bayhenn: Combining bayesian deep learning and homomorphic encryption for secure dnn inference. In *Proceedings of The 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4831–4837, 2019.

[42] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 13–23, 2019.

[43] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

[44] Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the International Conference on World Wide Web (WWW)*, page 1351–1361, 2015.

[45] J. Zhang, Y. Fan, J. Zhang, and B. Bai. Learning to build accurate service representations and visualization. *IEEE Transactions on Services Computing*, 2020.

[46] Jia Zhang. A mobile agent-based tool supporting web services testing. *Wireless Personal Communications*, 56:147–172, 2011.

[47] Y. Zhang, Y. Qian, and Y. Wang. A recommendation algorithm based on dynamic user preference and service quality. In *IEEE International Conference on Web Services (ICWS)*, pages 91–98, 2018.

[48] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu. Privacy-preserving blockchain-based federated learning for iot devices. *IEEE Internet of Things Journal*, 2020.

Biographies

Xing Wu received his BS degree in control theory and application from Tsinghua University, China, in 2017. He is currently working toward a Ph.D. degree in the Department of Automation, Tsinghua University. His research interests include services computing, service recommendation, federated learning and blockchain.

Yushun Fan received his Ph.D. degree in control theory and application from Tsinghua University, China, in 1990. He is currently a professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. From September 1993 to 1995, he was a visiting scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He has authored 10 books and published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process re-engineering, workflow management, system integration, object-oriented technologies and flexible software systems, petri nets modeling and analysis, and workshop management and control.

Jia Zhang received her PhD degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, Professor of Department of Computer Science at Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graphs, and their interdisciplinary applications. Dr. Zhang has co-authored one textbook "Services Computing" and has published over 170 refereed journal papers, book chapters, and conference papers. Dr. Zhang has served as an associated editor of the IEEE TSC since 2008. She served as Program Committee Chair for IEEE SCC (2020), ICWS (2019), CLOUD (2018), and BigData Congress (2017). She is a senior member of the IEEE.

Zhenfeng Gao received his PhD degree in control theory and application in 2018 from Tsinghua University, China. He is currently working as a postdoctor at the Graduated school at shenzhen, Tsinghua University as well as the postdoctoral research center at Sangfor Technologies Inc. His research interests include services computing, service recommendation, big data and blockchain technology.