Journal of Industrial and Management Optimization

doi:10.3934/jimo.2022223

DEEP LEARNING-BASED RECOMMENDATION METHOD FOR TOP-K TASKS IN SOFTWARE CROWDSOURCING SYSTEMS

Zhanglin Peng✉, Dequan Wan✉, Anning Wang✉,* and Xiaonong Lu✉

School of Management, Hefei University of Technology
Key Laboratory of Process Optimization and Intelligent Decision-making, Ministry of Education
Hefei 230009, China

Panos M. Pardalos✉

Department of Industrial and Systems Engineering, University of Florida
Gainesville, 32611, USA

(Communicated by Kok Lay Teo)

Abstract. The task personalized recommendation problems in software crowdsourcing systems have unique characteristics, i.e., large task flow, high task complexity, long development cycle, winning task competitions, professional ability requirements, etc. However, existing software crowdsourcing recommendation mechanisms do not consider the contextual information of crowdsourcing tasks. In particular, the effects of crowdsourcing workers' interest changes and capability constraints on task selection are usually ignored. Therefore, this study proposes a new worker capability-correction long- and short-term attention network (CLSAN) recommendation framework. Firstly, explicit and implicit features are obtained from the feature data of crowdsourcing workers and historical crowdsourcing tasks. The long-term and short-term feature layers are extracted by adding LSTM to the historical tasks; the attention weight of user preference is calculated by integrating the attention mechanism to obtain each user's personalized preference. Secondly, this study considers that the capabilities of the worker need to be matched with the skills required for the software task. We design a worker capability correction model that uses Word2Vec software to obtain competency similarities between crowdsourcing workers and tasks, thereby determining the priority of tasks that satisfy the worker's capability. The experimental results show that CLSAN can accurately evaluate the changes of interest preferences of crowdsourcing workers, and effectively improve the quality and efficiency of crowdsourcing recommendations.

1. **Introduction.** The advancement and popularization of emerging Internet information technology have promoted the application and development of swarm intelligence in online virtual communities. Large-scale "amateur" users are being guided to participate in completing tasks in a cooperative and collaborative manner. In 2006, Jeff Howe first proposed the term 'crowdsourcing' in Wired Magazine [10].

2020 *Mathematics Subject Classification.* Primary: 90B50; Secondary: 62P20.

Key words and phrases. Software crowdsourcing, recommendation algorithm, deep learning, attention mechanism, user preference.

The work is supported by grants from the National Natural Science Foundation of China (No: 72071060, 72101078, 72171069, 71901086) and the Fundamental Research Funds for the Central Universities (No. JZ2021HGTA0131).

*Corresponding author: Anning Wang.

Under this socialized production of crowd intelligence, with regard to software engineering, software development tasks and developers are no longer limited to specific, isolated community groups. Instead, an increasing number of people are choosing to publish or participate in the completion of tasks on crowdsourcing platforms, gradually forming a new cross-time and cross-regional software crowdsourcing model [3]. For example, GitHub, the world's largest open source software platform, has more than 40 million developers worldwide. These developers not only have easy access to source code, program documentation and test cases, but they also have the freedom to create and manage open source software (OSS) projects. Developers can share ideas, experiences and source code with many other professional developers and hobbyists on the platforms [11]. In recent years, millions of software development tasks have been implemented by many successful crowdsourcing platforms, including but not limited to platforms like Yipinweike, Zhubajie, Amazon Mechanical Turk, Topcoder, and Local Motors.

Software crowdsourcing is mediated by platforms that connect requesters with workers [33], and is the accomplishment of specified software development tasks that have been undertaken on behalf of an undefined group of external people with the requisite specialist knowledge [22]. At present, software crowdsourcing has become a viable development paradigm for software-as-a-service (SaaS) ecosystems. Many virtual online communities enable enterprises to tap into their global talent pools and crowdsource a variety of SaaS development tasks, including requirement analysis, architecture design, coding, testing and so on [27]. Klaas-Jan Stol et al. [22] pointed that some potential benefits have been linked to the use of software crowdsourcing, such as cost reduction, faster time to market, and higher quality through broad participation, creativity and open innovation. However, at any point in time, the number of open software tasks and development workers can become quite large. The types of tasks are diverse, and the workers are heterogeneous. Workers are faced with the challenge and difficulty of evaluating a large number of tasks and identifying one in which they wish to participate. The selection of tasks is autonomously initiated by the workers. This results in most workers potentially compromising tasks that they decide to undertake but for which they may not be suitable. As such, they may fail in task competitions. Manually selecting a preferred task will cause problems, such as low task completion efficiency and the reduced motivation of workers participating in the tasks. In such circumstances, a personalized recommendation system can turn out to be an effective solution to matching software tasks with development workers. Personalized recommendation systems not only can significantly help the workers to reduce costs in searching for the appropriate tasks in which they can participate and for which they can provide high quality solutions; such systems can also give considerable help to requesters, saving time and increasing productivity.

Recommendation algorithms have already been proven to be very successful in e-commerce platforms and some entertainment platforms, such as Amazon, Alibaba and Netflix. However, compared with traditional e-commerce recommendation scenarios, software crowdsourcing systems have certain unique features, i.e., the huge flow of tasks and workers, the heterogeneous nature of tasks and workers, task competition winning, the importance of workers' capabilities, and the quality of tasks completed [20]. All of these characteristics mean recommendation features in software crowdsourcing systems face more challenges including: (1) the heterogeneity of software tasks and workers, including the value of the task, the knowledge and

skill required, the cognitive effort required and the complexity of the task [22]. In contrast to all other micro-tasks, software crowdsourcing tasks are often interdependent, complex, and heterogeneous. Similarly, tasks are assigned to heterogeneous developers with different skills, efforts, motivations, etc., which increases the complexity of task selection and allocation [30]. (2) Dependence on workers' capabilities: different from consumers on e-commerce platforms, workers are the value creators on digital software crowdsourcing systems. They also have value attributes similar to enterprise employees, and ability attributes similar to typical knowledge workers. Software tasks have strong knowledge barriers, which often require workers to have different types of software engineering theoretical knowledge, cognitive processing capabilities, code development, testing ability, and a willingness to spend a lot of time on tasks. Importantly, these abilities and knowledge require repeated training and cannot be acquired in a short period of time. (3) Context-aware of workers' preferences on tasks: workers are driven to participate in software tasks by extrinsic and intrinsic motivation factors, and task attributes have significant impacts on workers' intrinsic motivation. Thus, the intrinsic motivation of different workers when choosing different tasks is obviously different, and this has decisive significance [36]. Intrinsic motivation is continuous, and correlations will exist at different temporal positions, which are reflected in the context of workers' historical behavior sequences. Accordingly, software crowdsourcing recommendation systems must capture the intrinsic motivation of workers to engage in different tasks chronologically. This can be achieved through sufficient contextual order information. (4) Dynamic changes in workers' preferences and capabilities: the process of user participation in crowdsourcing initiatives can be abstracted as three stages consisting of solver intention, solver participation and continuous solver participation. Those three stages are dynamic, progressive and interrelated [34]. During the different stages, a worker can be stimulated by different task requirements, different motivations, and different complexities of the task when participating in software crowdsourcing initiatives. Furthermore, a worker's capabilities will change in line with time-series locations. The capabilities which the worker has shown in completing near-time tasks can more accurately reflect the essence of that worker's capabilities. These distinct characteristics and challenges make traditional recommendation methods unsuitable for software crowdsourcing systems. Therefore, a special recommendation algorithm should be designed that efficiently delivers personalized recommendation software tasks for workers.

In recent years, neural networks, such as recurrent neural networks (RNN) and attention mechanisms, have begun to be widely used in many recommendation systems. Recurrent neural networks have significant advantages in the application of recommendation systems. The RNN model records the user's behavior and the corresponding temporal context as a temporal behavior sequence, so as to make better use of the context relationship and obtain the temporal order dependency. The attention mechanism is able to adaptively integrate various features, specifically to capture user preferences from the entire sequence. However, existing crowdsourced recommendation methods structure worker models without considering time-series locations. This results in a lack of time-sequential information. These worker models are constructed with the assumption that workers' interests are stable and focused over time. As a result, shortcomings also exist in terms of describing and distinguishing users' dynamic and changing long-term preferences and short-term

needs, both of which are critical for enabling accurate software crowdsourcing task recommendations.

This article proposes a worker capability-correction long- and short-term attention network (CLSAN) for top-K task recommendations in software crowdsourcing systems. In practice, a CLSAN can effectively combine the features and correlations between workers and task interactions, in order to simulate user behavior sequence changes. This approach captures the user's long-term and short-term preferences and adaptively adjusts the degree of association between workers and tasks to achieve accurate recommendations. In the proposed CLSAN, this study introduces 'long short-term memory' (LSTM) to model long-term and short-term sequential behavior features between workers and tasks, thus preserving sufficient contextual order information. Workers' long-term and short-term preferences are captured from historical behaviors, and the attention mechanism is used to give different weights to workers' expressions of interest in different historical behaviors. This method can ignore irrelevant content in interests and deeply learn the dynamic characteristics of workers' interests. Furthermore, we propose a worker capability correction mechanism. The mechanism is that, for each task, consideration is given to how well the worker's capability assessment matches the task's required skills. Finally, federated learning training is applied, to obtain the worker's predicted recommendation top-K tasks list. On real crowdsourced datasets, extensive experiments demonstrate the effectiveness of this method.

In summary, the main contributions of this paper are as follows:

(1) This study proposes a novel framework, specifically a worker capability-correction long- and short-term attention network (CLSAN) for accurate recommendations related to software crowdsourcing tasks. The model can effectively utilize the feature information between workers and software tasks and deeply learn to mine the workers' dynamic interest changes of workers.

(2) A worker capability correction mechanism is introduced. This mechanism fully considers the perceived similarity between workers and software tasks, to ensure that the required task skills can match the worker's ability constraints, thus improving recommendation performance.

(3) An extensive experimental study on real crowdsourced datasets shows that the proposed model outperforms other baseline algorithms, e.g., collaborative filtering (CF), singular value decomposition (SVD), LSTM, and LSTM-attention. These experiments prove the effectiveness and efficiency of the proposed CLSAN model.

(4) Our research can further improve the accuracy of the recommendation algorithm, help the crowdsourcing platform further optimize its recommendation algorithm and recommendation mechanism, and improve the recommendation effect. It also helps the crowdsourcing platform further optimize its governance mechanism and improve user experience and satisfaction.

The rest of this article is organized as follows: First, some works related to this study are reviewed in Section 2. Then, in Section 3, the proposed research question is formulated, detailing the principle and framework of the CLSAN. Extensive experiments are carried out in Section 4, and the results are analyzed. Finally, the full text is summarized in Section 5.

2. **Related work.**

This section reviews some related works on crowdsourcing personalized recommendation systems, including traditional general recommendation

methods and deep learning based recommendation methods. Finally, the attention mechanism is introduced.

2.1. **General recommended.** Traditional recommendation systems are mainly based on collaborative filtering recommendation frameworks. One such system is memory-based collaborative filtering, which relies on similar users, who are likely to have similar interests. The system computes user-user or item-item similarity to generate recommendations. For example, Yuan et al. [31] inferred user ratings from the interaction behavior of crowdsourced workers and tasks, thereby learning a task recommendation method based on worker latent feature space and task latent feature space. Fu et al. [7] proposed a cluster based collaborative filtering classification model (CBC), which describes the winner prediction problem of software developers as a multi label classification problem. Liao et al. [14] divided a group of workers into several characteristic communities with similar behaviors, based on the workers' reputations, preferences and activity characteristics. Through interaction between multiple communities, top-N worker groups were selected for recommendation. User participation behavior is also affected by many factors, such as internal and external motivation and social environment [4]. Therefore, Wang et al. [25] considered using the reputation and participation frequency of mobile crowdsourcing workers to gain the trust of tasks, combined with the workers' dwell time, to predict potential tasks. Li et al. [12] further considered the social influences of social affiliation and social intimacy to identify suitable contributors, specifically those with high willingness to contribute and who were able to complete tasks with high-quality results. In addition, some studies have focused on the impact of geographic location [23], privacy protection [21] and expert recommendations [19] on task recommendation.

The other type of system is a model-based collaborative filtering recommendation. This system mainly uses machine learning or data mining methods (such as Bayesian network, clustering, and association rule mining) to train and learn data samples and complete the recommendation. For example, Li et al. [13] proposed using the k-medoids clustering algorithm to obtain similar workers, and to provide accurate and effective suggestions through the group selection and construction of low-will workers. Mao et al. [15] proposed a Crowdrex recommendation system, which will extract historical data and challenge data of developers as the input of their models, and automatically match tasks and developers through content. You et al. [1] used the crowdsourcing competition participation information, together with the features of workers and competitions, as auxiliary information. The study concluded that the combination of transfer learning and feature-based matrix factorization can effectively predict the winner of crowdsourcing competitions. The study of Yuan et al. [32] was based on a probabilistic matrix factorization (PMF) model, which considers workers' task choice preferences and workers' performance history. The study further described how the weight of worker task choice preference gradually decreases over time. The above research has made certain contributions to the field of crowdsourcing recommendation study. However, traditional recommendation methods often only consider modeling worker or task features; these methods assume that worker-task interactions are static. As such, they cannot effectively simulate the dynamic changes of workers' historical behavior context information over time, making it difficult to capture workers' long-term and short-term preferences.

2.2. **Deep-learning-based recommendation.** Recently, neural networks have made outstanding contributions in the fields of natural language processing, video and image understanding. Neural networks have also achieved good results in the application of recommendation systems. Many researchers have tried to introduce deep learning into the field of crowdsourcing recommendation algorithms. For example, Pan et al. [18] proposed a deep learning-based tag semantic task recommendation model, which calculated the similarity of word vectors through Word2Vec software. The study established a semantic tag similarity matrix database, as a means to realize personalized recommendations for crowdsourcing tasks. Yang et al. [29] constructed a dynamic crowd work decision support model (DCW-DS) using multiple influencing factors, which can predict the role of developers in a given challenge (registrant, submitter or winner). Zhang et al. [35] further proposed different meta learning adaptive prediction models for the registration, submission and winning stages of developers, and then used the meta characteristics of learning as a strategy to recommend topk developers for challenges issued by customers. Yi et al. [6] proposed a CRF based method to learn software crowdsourcing task requirements from task descriptions. This research uses potential Dirichlet assignment (LDA) topic modeling to obtain the topic distribution of crowdsourcing tasks, and quantitatively measures the matching between tasks and developers. Gao et al. [8] studied the assignment of tasks to users through expert knowledge, using tree decomposition techniques and heuristic depth-first search algorithm (DFS+HA) to rank the assigned tasks. In addition, the consideration of explicit and implicit relationship features between users and tasks is a key factor affecting the construction of recommendation models. He et al. [9] introduced the consideration of implicit feedback. The study used a multilayer perceptron to learn the user-item interaction functions, in order to model the key factors in collaborative filtering. Wang et al. [24] proposed a user-item graph collaborative filtering method (NGCF), based on a graph neural network, by exploring the potential collaborative relationship in user-item interactions. Xie et al. [26] considered the multiple implicit relationships of interactions between software developers and tasks. The study generated predictions based on a deep neural network architecture to make recommendations. The introduction of deep learning can effectively capture global context information. However, the modeling of workers' individual interests is still not adaptive enough and is unable to identify changes in workers' interests with different preferences and behavior sequence distributions. The attention mechanism was first proposed by Desimone [5]. The mechanism's principle is to simulate the attention of the human brain and selectively allocate limited attention to more important information. Specifically, the attention mechanism forces the model to focus on the most important parts of the target with different weights. Inspired by the attention mechanism, we first design our own CLSAN recommendation model to capture features that are more likely to attract user interest. The proposed model can correlate explicit and implicit features between crowdsourcing workers and tasks, for global optimization.

3. **The proposed system model.** This section introduces our CLSAN recommendation model. First, the software crowdsourcing task recommendation and the basic framework concept are defined. Then, we give the description of the preference feature extraction and worker ability correction methods defined by the CLSAN model, as well as the definition of parameters and the generation of the list of predicted recommendation probabilities. The goal of the CLSAN model is to

optimize the task selection scheme of crowdsourcing workers and help them get the top-K recommended task list that is interested and matches their abilities.

3.1. **Problem definition.** Let $U = \{U_1, U_2, ..., U_l, ..., U_n\}$ denote the set of n crowdsourcing workers, and U_l represents the feature vector of the l-th crowdsourcing worker; $V^l = \{V_1^l, V_2^l, V_3^l, V_4^l, ..., V_t^l\}$ represents the set of historical task data of the l-th crowdsourcing worker. Tasks are ordered by time series of interactions with crowdsourcing workers, and V_i^l $(i \in [1, t])$ represents the session of user U_l at time i. Specifically, the crowdsourcing worker's behavior sequence data V^l are divided into sessions, and each session V_i^l represents a crowdsourcing worker's behavior within a task. Obviously, the task behavior at different times represents the long-term or short-term preferences of the particular crowdsourcing worker. Then, we distinguish the data of crowdsourcing workers U_l and task behavior V^l into structured data and unstructured data. The features extracted from structured data are defined as explicit features C_e; those features extracted from unstructured data are defined as implicit features C_i. In summary, our task recommendation goal is to recommend the crowdsourcing worker U_l the next task V_{t+1}^l from the target task set V_T in which he is interested and has a probability of winning. This is achieved by combining the explicit and implicit features of the crowdsourcing workers and historical tasks. The key notations used in this paper are presented in Table 1.

TABLE 1. Key notations and their descriptions.

Notations	**Explanations**
U, V	Set of crowdsourcing workers and task data
V_i^l	Interaction task session of the user l at time i
C_e	Explicit features from structured data
C_i	Implicit features from unstructured data
V_T	Set of target task
e_i	Overall explicit feature vector of task session at time i
W^*, b^*	The weight matrix and bias vector
b_t, b_u	The vector of the target task requirement and the worker's ability
∂	The time decay coefficient
$C_{e,a}$, $C_{e,u}$	Explicit feature vector of target task and crowdsourcing worker

3.2. **Recommendation framework.** The architecture diagram of the proposed model is shown in Figure 1. Specifically, the CLSAN framework consists of an embedding layer, a preference extraction layer, a capability correction layer, and a prediction layer, as follows:

(1) Embedding layer: we encode the preprocessed structured data. Specifically, numericalize, clean and convert the information that the model needs to process, and input in the form of data. Since the input data is high-dimensional and sparse, the high-dimensional sparse input data are converted into a low-bit dense hidden representation through the embedding layer. The original sparse features are mapped into multiple fixed-length representation vectors, and all representation vectors are then concatenated, to obtain the overall explicit feature vector e of the session.

(2) For the preference extraction layer, we propose a feature extraction model based on LSTM-attention (as shown in Section 3.3). This model deals with explicit features obtained from structured data. Focus on the long-term and short-term sequential behavior characteristics of crowdsourcing workers under different

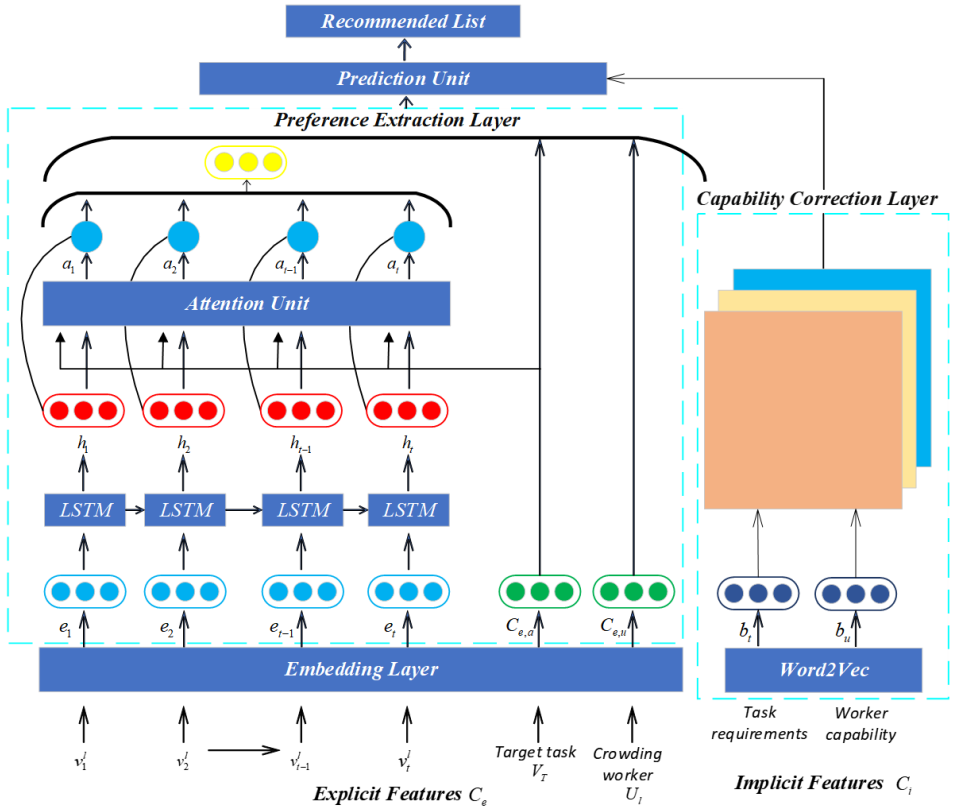

FIGURE 1. Worker capability-correction long- and short-term attention network (CLSAN) recommendation framework

time sequence information, so as to extract the interest preference of crowdsourcing workers.

(3) For the capability correction layer, we propose a capability corrected method, based on Word2Vec (as shown in Section 3.4). This method obtains implicit features from unstructured data. By matching the capability of crowdsourced workers with the task requirements, we can obtain a modified matrix about the comprehensive capability of workers. It can help us to select tasks that crowdsourcing workers are more capable of completing.

(4) Prediction layer: The task prediction layer (as shown in Section 3.5) predicts the interest features of the crowdsourced workers from the preference extraction layer. Then, it is combined with the comprehensive capability of workers in the capability correction model for common learning, so as to obtain a more accurate recommended task list.

3.3. **Worker preference feature extraction.** On the crowdsourcing platform, different crowdsourcing workers will choose different tasks, and they are more inclined to choose tasks that interest them. Therefore, a worker's choices of different tasks reflect all his/her interest preferences. Importantly, user interests are not fixed. Temporal changes and contextual information between different tasks reflect changes in the interests of crowdsourcing workers. Therefore, the proposed model

combines LSTM and an attention mechanism as a preference extraction model to focus on sequence information at different times. This way, the interesting features of these data can be captured, thus obtaining dynamic changes about long-term and short-term interests.

3.3.1. *LSTM model.* LSTM is a variant of RNN suitable for processing serialized data. Specifically, LSTM adds a memory unit and three control gate structures, consisting of a forget gate, an input gate and an output gate. The structure of a memory cell is illustrated in Figure 2.

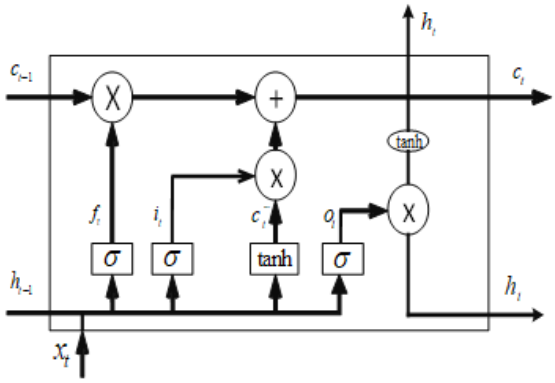

FIGURE 2. Structure of LSTM

To process sequence information, the three-gates structure will use the input x_t at the current time t, the output h_{t-1} at the previous time $t-1$, and the cell state c_{t-1} at the previous time $t-1$ to achieve each LSTM unit update. Therefore, each gate acts as a filter, and each filter fulfills a different function. Specifically, the forget gate f_t determines what information needs to be discarded at present. It will first check h_{t-1} and x_t, and then use the generated f_t to control those in the previous state c_{t-1} that need to be retained or forgotten. The input gate i_t determines how much new information in x_t and h_{t-1} will be added to the internal state and used to synchronously complete the update of cell state c_{t-1}. The output gate o_t determines which information will be used as the output h_t of the current state and outputs it to the next LSTM unit.

The specific calculation formulas of LSTM are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{3}$$

$$\tilde{c_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t}, \tag{5}$$

$$h_t = o_t \odot tanh(c_t), \tag{6}$$

where W and b refer to the weights and deviations of the raining matrix, respectively; $tanh$ refers to the hyperbolic tangent activation function; $\odot$ is the dot product operation; $\tilde{c_t}$ is the candidate value of the cell unit, and the c_t state at the current time t. σ is the sigmoid activation function. Its calculation formula can be seen in

(1)-(3); the sigmoid function scales the activation value between 0 and 1. Here, 1 represents completely retaining the stated value, whereas 0 represents completely discarding the value. The state C of the memory unit is used as the accumulator of state information, and the state c_{t-1} is updated to the state c_t using the Formula (5). In this way, h_t and c_t will be transferred to the next memory unit at each time-step, in order to realize the repetition of the information transfer process.

For crowdsourcing workers, workers' task choices change dynamically over time; these tasks are also contiguous and finite in nature, rather than being an isolated set of points. Therefore, LSTM is used to simulate the contextual information relationship of crowdsourcing workers' task sequence changes. The long-term and short-term sequence-related information of the task is learned and stored by using the LSTM gated structure to control the path of information transmission. The dependencies among tasks can be more comprehensively captured, thereby selectively preserving contextual information. The model has a prominent auxiliary role in the mining and analysis of user interest preferences.

3.3.2. *Attention mechanism.* Different tasks may have different amounts of information when chosen by crowdsourcing workers. Therefore, it is necessary to assign higher weights to some tasks and lower weights to others, in order to obtain crowdsourcing tasks that are more important in terms of historical behavior. The attention mechanism can well capture the correlation between each task selected by the workers and can also obtain the factors that change the task interest characteristics of the crowdsourced workers, thereby learning more representative crowdsourced worker representations.

First, the attention mechanism causes each intermediate output result of the LSTM layer modeling the input sequence to be associated with the output sequence value. The mechanism also trains a model to learn how to selectively focus on input data, which in turn assigns higher weights to more relevant input vectors. Note that the calculation formula of the weight a_t is:

$$a_t = \frac{\exp(h_t W C_{e,a})}{\sum_{j=1}^{T} h_j * W C_{e,a}}, \tag{7}$$

where $C_{e,a}$ is the explicit feature vector of target task, $W \in R^{n_H \times n_A}$, n_H is the dimension of the hidden state, n_A is the dimension of the target task vector, and $*$ represents the scalar-vector product.

Then, the attention weight configuration is performed. Attention resources are allocated to the feature vector h_t, according to the degree of association between tasks, thereby generating a weighted representation of the feature vector and enhancing the feature expression of the task. Attention score ${i_t}'$ can be calculated as follows:

$${i_t}' = \sum_{j=1}^{T} a_{t,j} * h_{t,j}, \tag{8}$$

where $a_{t,j}$ is the weight value of the j-th feature at time t; $h_{t,j}$ is the j-th feature at time t, ${i_t}'$ represents the feature vector with attention weight. The attention score can reflect the relationship between the target task and the input information h_t, a strong correlation can also make the attention score higher.

3.4. **Capability correction model.** Software crowdsourcing tasks have strong capability barriers, specifically with regard to professional knowledge and technical capabilities. As such, crowdsourcing workers are often required to have the corresponding skills required to undertake the task. The knowledge and skills possessed by different crowdsourcing workers may be different and even inadequate. This can lead to crowdsourcing workers choosing to abandon tasks of interest, due to a lack of corresponding technical capabilities. The belief with regard to perceived ease of use in a technology acceptance model (TAM) is that, when the public perceives their own ability to complete an activity more easily, the possibility of their participation is greater. Naturally, then, crowdsourcing workers will give priority to undertaking tasks at which they are good. Therefore, this study corrects the recommendation model based on interest preference extraction, by measuring the matching degree between worker capabilities and task requirements.

In this article, the ability of crowdsourced workers is measured through two data sources: (1) Platform registration information, which includes the skills and project descriptions that workers are good at performing. (2) Historically completed software crowdsourcing tasks: each task has a descriptive text that describes the task requirements, functional purpose, etc., thereby providing more personalized information. Here, worker skills are defined as the task requirements needed for workers to participate in completing those tasks. For example, if a worker participates in a task that requires Java to be developed, it is natural to assume that the worker is good at Java technology. Therefore, Java-related tasks can be recommended to the worker.

A popular deep learning-based word vector similarity calculation tool is Word2Vec. The basic idea of Word2Vec is to judge the semantic similarity between words, according to the distance between them (such as cosine similarity, and Euclidean distance). This tool makes full use of the context of words to make semantic information richer. Therefore, we compute the similarity between worker capabilities and target tasks by introducing the Word2Vec semantic training model. The text description of the target task and worker ability is converted into word vectors, thus obtaining worker capability weight vectors and target task weight vectors under each historical task. The cosine similarity $s(b_t, b_u)$ between target tasks and worker capabilities is calculated as:

$$s(b_t, b_u) = \frac{\sum_{i=1}^{m} b_{ti} * b_{ui}}{\sqrt{\sum_{i=1}^{m} (b_{ti})^2} \sqrt{\sum_{i=1}^{m} (b_{ui})^2}}, \tag{9}$$

where b_t represents the vector representation of the target task requirement, b_u represents the vector representation of the worker's ability, and m represents the dimension of the vector. When constructing the ability similarity matrix, the larger the value of $s(b_t, b_u)$ is, the better will be the ability of the worker to meet the needs of the target task.

In the process of crowdsourcing workers bidding for tasks, the workers' preference for certain tasks will gradually be attenuated over time [28]. The recent behaviors of crowdsourcing workers are also more meaningful than their early behaviors. Therefore, when calculating the task-ability similarity, the time decay function $f(T - t_i)$ that is suitable for the application scenario of the test set in this paper is introduced

as follows:

$$f(|T-t_i|)=\frac{1}{1+\partial\,|T-t_i|}, \tag{10}$$

where T represents the current time node, t_i represents the time node when the i-th task occurs, and ∂ is the time decay coefficient.

The introduced time decay function $f(T-t_i)$ reflects the degree of worker preference decay for different tasks. Therefore, the correction matrix for the comprehensive capability of crowdsourced workers is obtained as:

$$s'=\sum_{i=1}^{n} s(b_t,b_u)f(|T-t_i|), \tag{11}$$

where n represents the total number of tasks with which the crowdsourcing workers interact, and s' represents the matching value between the target task and the worker's comprehensive capability.

3.5. **Recommendation prediction.** Through joint learning of the vectors output by the model, the predicted probability scores of the crowdsourcing workers for the tasks to be recommended are obtained.

Specifically, the crowdsourcing workers explicit feature vector $C_{e,u}$, the target task vector $C_{e,a}$, and the crowdsourcing workers interest preference feature vector ${i_t}'$ are all combined into the overall feature vector v:

$$v={i_t}'\oplus C_{e,a}\oplus C_{e,u}. \tag{12}$$

Then, v is input to the fully connected layer, and the excitation function of each neuron in the fully connected layer adopts the *ReLU* function, in order to obtain the weighted feature vector v':

$$v'=ReLU(vW'+b'), \tag{13}$$

where W' and b' represent the network parameters of the fully connected layer.

The output value of the last fully connected layer is passed to an output, which in turn is associated with the capability feature vector. The function is used as the network objective function to guide the classification. The vector v' and the capability correction vector s' are normalized to obtain the probability value y' as follows:

$$y'=tanh((v',s')W''+b''), \tag{14}$$

where W'' and b'' represent the network parameters that the model can learn. Thus, a top-K recommendation is performed on y' in descending order, and the final task recommendation list is obtained.

4. Experiment and results.

4.1. **Experiment.** In order to verify the performance of the proposed model in this paper, we conducted extensive experiments on the real data set.

We crawled the information of software crowdsourcing workers and interactive historical tasks on the "Yipingweike" crowdsourcing website as an experimental data set. Yipingweike is one of the most widely used online crowdsourcing platforms in China. The data mainly include crowdsourcing worker information, task information, bidding and bid winning records, etc. Data were divided into structured data and unstructured data according to type, and the explicit and implicit features defined in this paper were extracted from that data. The specific data items obtained are shown in Table 2 and Table 3:

TABLE 2. Experimental data explicit feature.

Feature Data		Feature Type	Feature Description
Crowdsourcing worker structural data	Worker ID	Text	Name information
	Type	Category	Individual, studio or business
	Rating	Category	1st rate, 2nd rate, ..., 9th rate
	Reputation score	Number	Reputation score, integrity level of a worker
	City	Category	City which the worker is located in
	Identity information	Number	Time of the worker updates his/her certificate information
	Total transaction amount	Number	Total bonuses earned by worker for completing historical tasks
	Completed total tasks	Number	Total number of historical tasks completed by worker
	Employer positive rating	Number	Task posters' evaluation metrics for a crowdsourcing worker
	Total winning bids	Number	Total number of historical tasks won by a crowdsourcing worker
Crowdsourcing task structural data	Type of task	Category	The type attribute which the task belongs to
	Sub-task type	Category	The Sub-type attribute which the task belongs
	Task price	Number	A reward for completing the task
	Task followers	Number	Number of workers that the task has been followed
	Number of bidders	Number	Number of workers that the task has been biden
	Number of successful bidders	Number	Number of workers who have won the bidding on the platform
	Time of posting the task	Number	Time of posting the task
	Submission deadline	Number	Submission deadline
	Transaction mode	Category	Tender, Hire, Single bounty, Multiplayer bounty, Piece count

The data acquisition and preprocessing process mainly includes the following:

TABLE 3. Experimental data implicit feature.

Feature Data		Feature Type	Feature Description
Crowdsourcing worker unstructured data	Personal description	Text	Self-introduction information of crowdsourcing worker
	Good at skills	Text	The expertise of crowdsourcing worker
Crowdsourcing task unstructured data	Task name	Text	Task name information
	Task requirements	Text	Detailed description of the task requirements

TABLE 4. Dataset statistics.

Statistics Category	**Statistics**
Number of workers	471
Number of tasks	38896
Number of task categories	284
Maximum number of task interactions for workers	1045
Minimum number of task interactions for workers	20
Average number of worker interactions	61.35
1st quartile (25th percentile) of worker task interactions	40
2nd quartile (50th percentile) of worker task interactions	61
3rd quartile (75th percentile) of worker task interactions	85.25

(1) Firstly, Python, Octopus and other crawler tools were used to perform large-scale real-time incremental crawling of the crowdsourcing workers and crowdsourcing tasks of the Yipinweike platform. By crawling the data of 2637 software crowdsourcing workers on the platform, and then locating and tracking the data of the historical interactive tasks of these crowdsourcing workers, a total of 53443 interactive tasks and task information were combined to form the crowdsourcing worker-task behavior sequence data.

(2) Secondly, in order to improve the quality of the data, the historical task data of the crowdsourcing workers was cleaned, and the meaningless tasks were eliminated. The meaningless tasks include unreal tasks, duplicate tasks and task testing. In addition, the data with 20 or more bidding records for a certain crowdsourcing worker were screened out. In order to ensure the activeness of crowdsourcing workers, it was necessary to delete workers that have not been updated for a long time in the bidding records of crowdsourcing tasks. We set the time threshold for the bid update time at 180 days. After the above data processing and cleaning, the data set of 471 crowdsourcing workers and 38896 software tasks were finally obtained. Table 4 provides statistical information about the dataset.

(3) Thirdly, in order to ensure the expressive ability of the original features of the data, and to improve the convergence speed and accuracy of the model, label encoding was performed on categorical features in the dataset. Word vector training was used on text features to convert text-like data into vectors, and normalization of numerical features was conducted to convert data values in the range [0, 1]. The

normalization processing formula is as follows:

$$x_i = \frac{x_i - \min}{\max - \min}, \tag{15}$$

where min is the minimum value under the corresponding numerical feature, and max is the maximum value under the corresponding numerical feature.

4.2. **Experimental methods.** This paper uses the historical task data set of crowdsourcing workers. For any one crowdsourcing worker, we randomly selected n tasks of the same scale with which the crowdsourcing workers had not interacted as negative samples. Meanwhile, the time when the negative samples occurred is guaranteed to be within the positive sample time interval. In the data set, a total of $2n$ tasks were used as data sets, which are arranged in the order of task release time. The first 80% of the data set was obtained as the training set, and the remaining 20% of the data set was used as the test set.

4.3. **Evaluation metrics.** This article utilizes precision, mean average precision (MAP), mean reciprocal rank (MRR) and normalized discounted cumulative gain (nDCG) to evaluate models' abilities to capture users' preferences and to perform time-sensitive next-item recommendations. These metrics are widely used in other recommendation areas.

Precision is commonly used to evaluate the recommendation results. 'Precision rate' refers to the ratio of the number of positive samples predicted by the model to the total number of positive samples. The precision method is shown in (16):

$$Precision = \frac{\sum\limits_{u \in U} |R(u) \cap T(u)|}{\sum\limits_{u \in U} |R(u)|}, \tag{16}$$

where $R(u)$ is the list of recommendations made to the crowdsourcing worker u based on his/her behaviors on the training set, and $T(u)$ is the list of real behaviors of the crowdsourcing worker u on the test set.

Next, MAP is a comprehensive recommendation evaluation index that focuses on sequence weights. In practice, MAP is used to calculate the arithmetic average on the basis of the average accuracy of each recommendation result. The top-K of MAP can be expressed as:

$$AP = \frac{1}{k} \sum_{j=1}^{k} Precision_i \times rel_i, \tag{17}$$

$$MAP = \frac{1}{|U|} \sum_{i=1}^{|U|} AP, \tag{18}$$

where k represents the length of the recommendation list, $Precision_i$ is the precision of a cut-off rank list from 1 to i, and rel_i represents the correlation of the prediction results at position i. The value of a positive sample will be 1; otherwise, it is 0. Finally, $|U|$ is the number of users.

Next, MRR is an internationally used measurement tool used for evaluating the accuracy of lists. In practice, MRR mainly evaluates the performance of a

recommendation algorithm in time series prediction, which can be summarized by the following equation:

$$MRR = \frac{1}{|U|}\sum_{i=1}^{U}\frac{1}{rank_i}, \tag{19}$$

where $rank_i$ is the rank of the first correct answer for the i-th task.

Finally, nDCG is used to measure the quality of recommendation ranking and consider the impact of recommendation tasks at different positions in the recommendation list on the overall recommendation results. Formally, nDCG is defined as:

$$DCG = \sum_{i}^{k}\frac{2^{rel_i}-1}{\log_2(i+1)}, \tag{20}$$

$$IDCG = \sum_{i}^{|REL|}\frac{2^{rel_i}-1}{\log_2(i+1)}, \tag{21}$$

$$nDCG = \frac{DCG}{IDCG}, \tag{22}$$

where $|REL|$ represents an ideally sorted list of correlations from largest to smallest.

4.4. **Experimental results and analysis.**

4.4.1. *Performance comparison.* In order to further verify the superiority of the proposed CLSAN model for software crowdsourcing task recommendations, CLASN was compared with the traditional collaborative filtering algorithm, SVD and LSTM-based algorithm used in our research. When top-K recommendation was carried out in the experiment, the length of the recommendation list took different values for the experiment, and the experimental results when was 5, 10, 15, and 20 were selected as the analysis samples. We repeated the experiment for all users collected in the experiment and selected the average value of each evaluation index (such as precision rate, MAP, MRR, and nDCG) in the recommendation test as the experimental result. Table 5 shows the experimental results of all the methods:

Table 5. Performance comparison of CLSAN with other algorithms on crowdsourced datasets.

Methods	Top-5				Top-10			
	P	MAP	MRR	nDCG	P	MAP	MRR	nDCG
CF	0.3602	0.1464	0.1097	0.3199	0.3637	0.1825	0.0914	0.3073
SVD	0.4816	0.4375	0.3075	0.6018	0.5750	0.4041	0.1940	0.6147
LSTM	0.3903	0.1522	0.1122	0.3577	0.5333	0.2527	0.0997	0.4351
LSTM-A	0.6667	0.4044	0.2100	0.5458	0.6904	0.4708	0.1536	0.6143
CLSAN	0.7333	0.6122	0.3544	0.7571	0.7971	0.6595	0.2333	0.7958
Methods	Top-15				Top-20			
	P	MAP	MRR	nDCG	P	MAP	MRR	nDCG
CF	0.3999	0.1854	0.0722	0.3295	0.4263	0.1984	0.0599	0.3405
SVD	0.5500	0.3721	0.1423	0.5888	0.5625	0.3669	0.1151	0.5907
LSTM	0.6222	0.3266	0.0870	0.5164	0.6500	0.3653	0.0757	0.5533
LSTM-A	0.6667	0.4561	0.1178	0.6107	0.6833	0.4794	0.0986	0.6311
CLSAN	0.8667	0.7211	0.1815	0.8417	0.8333	0.6971	0.1465	0.8240

Among different recommendation list lengths, the CF algorithm had the most unfavorable performance under nearly all metrics. On the one hand, many traditional CF algorithms consider user-task feature modeling, but they have difficulty modeling implicit features. Therefore, we considered introducing implicit features into the CLSAN algorithm for recommendation. On the other hand, a CF algorithm does not consider contextual information between tasks and cannot accurately identify users' dynamic interest preferences. However, the LSTM-based models achieved better results, which indicates that LSTM can effectively obtain the long-term and short-term preferences of workers. Moreover, the sequential information of user interaction behavior also plays an important role in crowdsourcing recommendations.

As shown in Table 5, we can find that, as the recommended list length changes, the proposed CLSAN model significantly outperformed SVD and other algorithms in almost all cases. In addition, SVD achieves similar but poorer performance than CLSAN, because a SVD algorithm can effectively pay attention to the display and implicit feedback information of a user-task interaction matrix. Compared with the baseline algorithm, both LSTM and LSTM-attention use deep neural networks for recommendation systems. This finding further demonstrates the importance of considering contextual information and time-order recommendation as a means to provide more information about user preferences. We attribute the performance enhancement of the CLSAN to the effect of the relevant context based on the attentional mechanism on the user's dynamic interest conversion. This shows that the application of an attention mechanism can capture the influence of historical interaction on user interest transformation.

In particular, one can find that both the precision and nDCG show a trend of first increasing and then decreasing. With an increase in the K value, the model can get more and more information regarding the potential interests of users at the beginning; this makes the recommendation effect better and better. However, when the recommendation list exceeds a certain range, the indications are that possible over-fitting will affect the final recommendation result. Therefore, in parameter comparisons, we set the recommended length of CLSAN to 15. The precision and nDCG models reached the highest value, 86.6% and 84.1% respectively, to ensure good performance.

4.4.2. *Model component analysis.* To further ensure the effectiveness of the algorithm, we provided optimal hyperparameter settings for the LSTM algorithm. As shown in Figure 3, several important hyperparameters were taken as examples. 'Dropout rate' is the retention rate of neurons in the training process, and is used to reduce the over-fitting problem of the neural network. Experimental results show that the best performance can be achieved when the dropout rate is 0.1. The learning rate can provide guidance on how to adjust the weight of the network through the gradient of the loss function. When the learning rate is 0.005, the performance is optimal.

In order to analyze the advantages of each computing component, we compared CLSAN, LSTM-attention, and LSTM for different metrics, under K=15. As shown in Figure 4 significant improvement can be observed in recommendation performance by adding the attention mechanism. This is because the attention mechanism can highlight relevant important features and reduce the interference of irrelevant features, in order to capture more accurate user interest features. Therefore, the results verify that an attentional mechanism can further optimize the ability to learn important situational information in the dynamic interaction of behavioral data.

In addition, CLSAN takes into account the embedment of the worker competency correction model and achieved 19.8% precision and 23.1% nDCG improvement on average, compared with LSTM-attention. In the real software crowdsourcing recommendation process, a crowdsourcing worker's choice of tasks will be restricted by the worker's professional competence. The staff competency calibration model can effectively identify tasks that better match the competencies of crowdsourced workers. The model ensures that crowdsourcing workers can first find and then select the tasks that are most relevant to them. Due to the embedding of the capability correction model, the next recommendation function of personalization in the proposed CLSAN model is further improved. This finding also shows that CLSAN not only can model the dynamics and different interest preferences of employees through attention networks, but can also fully simulate the behavioral patterns of employees interacting with real recommendation system.

In CLSAN, the time decay coefficient affects the influence of the association relationship between workers and tasks on the recommendation results, to a certain extent. As can be seen from formulas (9)-(11), CLSAN uses to balance the similar correlations between workers' abilities and tasks, considering that different workers have different levels of time sensitivity to task preference. When is 0, this is equivalent to completely ignoring the influence of preference time sensitivity. As increases, task-time relationships can be used to make higher priority recommendations. As shown in Figure 5, when ranges from 0.01 to 0.5, CLSAN's precision,

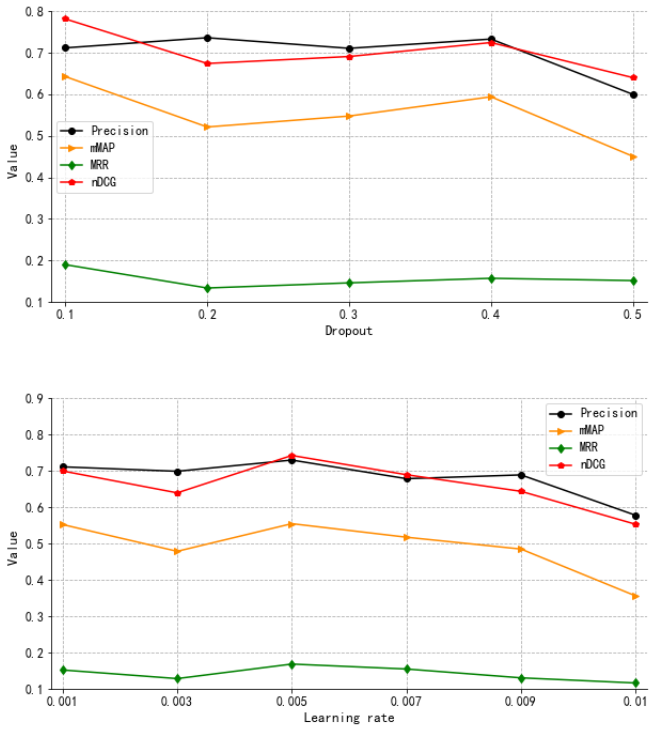

FIGURE 3. Parameter optimization process in datasets, K=15

and the values of MAP, MRR and nDCG decrease; when exceeds 0.5, these values gradually increase.

5. **Conclusions and discussions.** Traditional recommendation approaches are not suitable for software crowdsourcing systems that require timely, personalized recommendations of tasks/workers. Software crowdsourcing brings new features

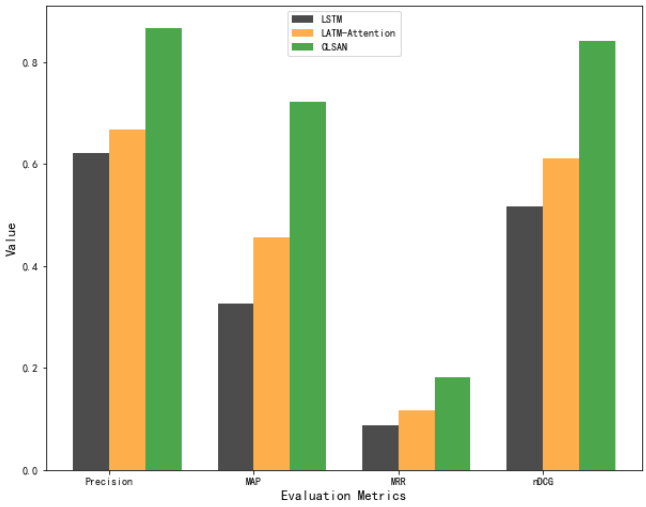

FIGURE 4. Performance comparison of different LSTM algorithms, K=15

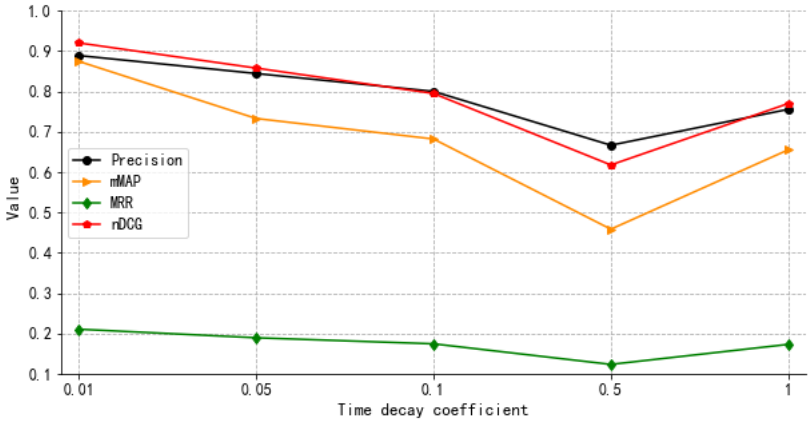

FIGURE 5. Performance comparison of CLSAN at different time decay coefficient, K=15

that do not exist in traditional recommendation scenarios, i.e., large task flow, high task complexity, long development cycle, winning task competitions, the importance of workers' professional capabilities, and so on. When designing recommendation systems, these properties not only need to be fully considered, but one must also note that the selection of tasks by crowdsourcing workers is driven by intrinsic interests. Personalized recommendations for crowdsourcing workers are made by excavating the interests and preferences of workers, and proactively recommending tasks that workers are interested in or that they need. Based on the research on the dynamic changes of the individual interests and preferences of crowdsourcing workers and the ability factors, this article proposes a worker capability-corrected long- and short-term attention network recommendation model (CLSAN). First of all, this method introduces the LSTM model, which is used to learn the context information in a large number of historical tasks and which combines the attention mechanism to apply different attention weights to the obtained information features. On the basis of retaining effective information features, the focus on key information features is increased, in order to obtain the personalized preferences of crowdsourcing workers. Finally, a worker capability correction model is introduced. Considering the actual recommendation situation, there are strong knowledge and skills barriers between crowdsourcing workers and tasks. Workers will often reject a task, due to a lack of corresponding abilities. The proposed model implements the refined sorting of the recommendation list by revising the worker's ability attributes. The experimental results show that, compared with traditional CF, SVD, LSTM, and LSTM-attention algorithms, the CLSAN algorithm proposed in this paper has an average increase of 30.6% in precision, 38.6% in MAP, 7.7% in MRR, and 33.0% in nDCG. The evaluation results showed the potential benefits of the CLSAN model in recommending appropriate tasks to crowdsourcing workers. CLSAN model can help the crowdsourcing platform to achieve a more accurate and personalized recommendation mechanism, quickly help the crowdsourcing workers find the most suitable task for them, and reduce the time cost and economic cost lost by the crowdsourcing workers when looking for task matching. In addition, it can help the platform optimize the service and management mechanism for crowdsourcing employees, formulate corresponding service strategies for different types of crowdsourcing employees, and improve user satisfaction and enthusiasm. Our work has contributed to solving the problem of heterogeneity between task requirements and software crowdsourcing workers, focusing on personalized task recommendation and providing inspiration for task recommendation in other crowdsourcing scenarios.

In the future, we plan to explore several possible directions to extend our work. First, we are interested in integrating auxiliary information, such as user reviews and trust relationships, to further enhance the performance of crowdsourcing recommendations. Second, crowdsourcing recommendation systems have a large number of complex scenes and heterogeneous data, which will cause a lot of noise interference to our recommendation results. It is worth mentioning that some advanced algorithm models have achieved remarkable success in solving scenario uncertainty and data uncertainty [17, 2, 16]. We will use these algorithm models to provide more effective suggestions and make our models more effective.

Acknowledgments. The work is supported by grants from the National Natural Science Foundation of China (No: 72071060, 72101078, 72171069, 71901086) and

the Fundamental Research Funds for the Central Universities (No. JZ2021HGTA01 31).

REFERENCES

[1] Y. Baba, K. Kinoshita and H. Kashima, Participation recommendation system for crowdsourcing contests, *Expert Syst. Appl.*, **58** (2016), 174-183.

[2] İ. Batmaz, F. Yerlikaya-Özkurt, E. Kartal-Koç, G. Kösal and G.-W. Weber, Evaluating the CMARS performance for modeling nonlinearities, in *AIP Conference Proceedings*, **1239** (2010), 351-357.

[3] A. Begel, J. Bosch and M.-A. Storey, Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange, and TopCoder, *IEEE Softw.*, **30** (2013), 52-66.

[4] L. Chen, A. Baird and D. Straub, Why do participants continue to contribute? Evaluation of usefulness voting and commenting motivational affordances within an online knowledge community, *Decision Support Syst.*, **118** (2019), 21-32.

[5] R. Desimone and J. Duncan, Neural mechanisms of selective visual attention, *Ann. Rev. Neuroscience*, **18** (1995), 193-222.

[6] Y. Fu, B. Shen, Y. Chen and L. Huang, TDMatcher: A topic-based approach to task-developer matching with predictive intelligence for recommendation, *Appl. Soft Comput.*, **110** (2021).

[7] Y. Fu, H. Sun and L. Ye, Competition-aware task routing for contest based crowdsourced software development, 6th International Workshop on Software Mining (SoftwareMining), Urbana, IL, USA, 2017.

[8] L. Gao, Y. Gan, B. Zhou and M. Dong, A user-knowledge crowdsourcing task assignment model and heuristic algorithm for Expert Knowledge Recommendation Systems, *Engrg. Appl. Artif. Intell.*, **96** (2020).

[9] X. He, L. Liao, H. Zhang, L. Nie and X. Hu, et al., Neural collaborative filtering, in *Proceedings of the 26th International Conference on World Wide Web*, 2017, 173-182.

[10] J. Howe, *The Rise of Crowdsourcing*, *Wired Magazine*. Available from: `https://www.wired.com/2006/06/crowds/`.

[11] W. Li, W.-J. Wu, H.-M. Wang, X.-Q. Cheng and H.-J. Chen, et al., Crowd intelligence in AI 2.0 era, *Frontiers Info. Tech. Electron. Engrg.*, **18** (2017), 15-43.

[12] Y.-M. Li, C.-Y. Hsieh, L.-F. Lin and C.-H. Wei, A social mechanism for task-oriented crowdsourcing recommendations, *Decision Support Syst.*, **141** (2021).

[13] Z. Li, B. Cheng, X. Gao, H. Chen and G. Chen, A unified task recommendation strategy for realistic mobile crowdsourcing system, *Theoret. Comput. Sci.*, **857** (2021), 43-58.

[14] Z. Liao, X. Xu, X. Fan, Y. Zhang and S. Yu, GRBMC: An effective crowdsourcing recommendation for workers groups, *Expert Syst. Appl.*, **179** (2021).

[15] K. Mao, Y. Yang, Q. Wang, Y. Jia and M. Harman, Developer recommendation for crowdsourced software development tasks, 2015 IEEE Symposium on Service-Oriented System Engineering, San Francisco, CA, USA, 2015.

[16] A. Özmen, G. W. Weber, İ. Batmaz and E. Kropat, RCMARS: Robustification of CMARS with different scenarios under polyhedral uncertainty set, *Commun. Nonlinear Sci. Numer. Simul.*, **16** (2011), 4780-4787.

[17] S. Özöğür-Akyüz and G.-W. Weber, Infinite kernel learning via infinite and semi-infinite programming, *Optim. Methods Softw.*, **25** (2010), 937-970.

[18] Q. Pan, H. Dong, Y. Wang, Z. Cai and L. Zhang, Recommendation of crowdsourcing tasks Based on Word2vec semantic tags, *Wirel. Commun. Mob. Comput.*, **2019** (2019).

[19] B. Rashidi, C. Fung and T. Vu, Android fine-grained permission control system with real-time expert recommendations, *Pervasive Mobile Comput.*, **32** (2016), 62-77.

[20] M. Safran and D. Che, Efficient learning-based recommendation algorithms for top-N tasks and top-N workers in large-scale crowdsourcing systems, *ACM Trans. Inf. Syst*, **37** (2019), 1-46.

[21] J. Shu, X. Jia, K. Yang and H. Wang, Privacy-preserving task recommendation services for crowdsourcing, *IEEE Trans. Services Comput.*, **14** (2018), 235-247.

[22] K.-J. Stol, B. Caglayan and B. Fitzgerald, Competition-based crowdsourcing software development: A multi-method study from a customer perspective, *IEEE Trans. Softw. Engrg.*, **45** (2019), 237-260.

[23] D. Sun, K. Xu, H. Cheng, Y. Zhang and T. Song, et al., Online delivery route recommendation in spatial crowdsourcing, *World Wide Web*, **22** (2019), 2083-2104.
[24] X. Wang, X. He, M. Wang, F. Feng and T.-S. Chua, Neural graph collaborative filtering, in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, 165-174.
[25] Y. Wang, X. Tong, K. Wang, B. Fan and Z. He, et al., A novel task recommendation model for mobile crowdsourcing systems, *Int. J. Sens. Netw.*, **28** (2018), 139-148.
[26] X. Xie, B. Wang and X. Yang, SoftRec: Multi-relationship fused software developer recommendation, *Appl. Sci.*, **10** (2020).
[27] X. Xu, W. Wu, Y. Wang and Y. Wu, Software crowdsourcing for developing Software-as-a-Service, *Front. Comput. Sci.*, **9** (2015), 554-565.
[28] L. Yang, Y. Hu and G. Shao, Preference prediction method based on time attenuation and preference fluctuation, *Optim. Method Softw.*, **36** (2016), 2011-2015.
[29] Y. Yang, M. R. Karim, R. Saremi and G. Ruhe, Who should take this task? Dynamic decision support for crowd workers, in *ESEM'16: Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, **2016** (2016).
[30] X. Yin, J. Huang and W. He, H. Yu and L. Cui, Group task allocation approach for heterogeneous software crowdsourcing tasks, *Peer-to-Peer Netw. Appl.*, **14** (2021), 1736-1747.
[31] M.-C. Yuen, I. King and K.-S. Leung, TaskRec: A task recommendation framework in crowdsourcing systems, *Neural Process. Lett.*, **41** (2015), 223-238.
[32] M.-C. Yuen, I. King and K.-S. Leung, Temporal context-aware task recommendation in crowdsourcing systems, *Knowl.-Based Syst.*, **219** (2021).
[33] A. L. Zanatta, L. S. Machado and G. B. Pereira, Software crowdsourcing platforms, *IEEE Softw.*, **33** (2016), 112-116.
[34] X. Zhang, Z. Peng, Q. Zhang, X. Tang and P. M. Pardalos, Identifying and determining crowdsourcing service strategies: An empirical study on a crowdsourcing platform in China, *J. Ind. Manag. Optim.*, **18** (2022), 1809-1833.
[35] Z. Zhang, H. Sun and H. Zhang, Developer recommendation for Topcoder through a meta-learning based policy model, *Empir. Softw. Engrg.*, **25** (2020), 859-889.
[36] H. Zheng, D. Li and W. Hou, Task design, motivation, and participation in crowdsourcing contests, *Int. J. Electron. Commer.*, **15** (2011), 57-88.

Received August 2022; revised October 2022; early access November 2022.