北京交通大学

Lancaster University College
at Beijing Jiaotong University

基于缓存驱动联邦学习的个性化文本分类研究

Empowering Knowledge Cache-driven Federated Learning to Natural Language Processing

Major: Computer Science

Student′ Name (In Pin-Yin): SHEN, Pengyuan

Student′ Name (in Chinese): 申鹏远

Student′ ID No.: 20722061

Supervisor′ Name: GAO, Bo

22 April 2024

Declaration

I certify that the material contained in this dissertation is my own work and does not contain unreferenced or unacknowledged material.

Regarding the electronically submitted version of this submitted work, I consent to the work being stored electronically and copied for assessment purposes, including the School's use of plagiarism detection systems in order to check the integrity of assessed work.

I agree to my dissertation being placed in the public domain, with my name explicitly included as the author of the work.

Date: 29 April 2020

Signed: Shen Peng yuan

学士论文版权使用授权书

本学士论文作者完全了解北京交通大学有关保留、使用学士论文的规定。特授权北京交通大学可以将学士论文的全部或部分内容编入有关数据库进行检索\提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

学位论文作者签名：申鹏远　　指导教师签名：

签字日期：2024 年 4 月 8 日　　签字日期：2024 年 4 月 8 日

详细中文摘要

随着边缘智能的广泛应用，为了保护数据隐私，联邦学习得到了越来越多的关注。然而，目前流行的联邦学习方法并不能保证模型在异构客户端上的泛化能力，在实际应用中往往具有一定的性能限制。为了解决该问题，个性化联邦学习逐渐成为了边缘智能领域的研究热点。与此同时，现有的个性化联邦学习方法虽然能够支持异构的边缘设备并提高模型性能，但它们大多基于参数交互的联邦学习架构。这种架构需要在边缘设备与服务端之间大量传输模型参数，导致其通信开销过大，并严重影响联邦学习系统性能。与此相比，基于缓存驱动的 FedCache 架构通过传输 Logits 来更新模型参数，这种方式具有更高的通信效率和客户端异构支持性，能够在保证高模型精度的同时降低联邦学习系统的通信开销。然而，由于 FedCache 架构中客户端采用的 ResNet 模型、生成哈希向量的编码器以及数据集的划分方法仅支持部分图像数据集，因此该架构仅适用于计算机视觉领域的部分场景。此外，在边缘智能领域中，自然语言处理同样具有重要的地位，并面临着严峻的数据泄露风险。因此，为了在确保自然语言处理模型性能的同时保护文本数据的隐私安全，将 FedCache 架构应用于自然语言处理领域具有重要的研究价值。

目前为止，在自然语言处理场景下还没有针对 FedCache 架构的研究。因此，本文重新实现了 FedCache 架构，并基于该架构进行了充分的实验，使其适用于计算机视觉与自然语言处理场景下的所有个性化联邦学习任务。

首先，本文调整了 FedCache 架构中客户端的本地模型以及哈希编码器，设计了 IMDB 和 SST-2 等自然语言处理数据集的划分方法。接着，本文充分展开对比实验，证明了通过上述改进，FedCache 架构在自然语言处理场景下取得了相较于 pFedMe、FedAvg 和 MTFL 三种经典联邦学习架构更良好的性能，并在通信效率方面提高了一个数量级。最后，本文基于 Android Studio，将 FedCache 训练的自然语言处理模型应用在了电影评论情感分类场景中。该应用程序主要包含四个功能模块，分别是数据查看功能、数据采集功能、电影推荐功能和模型文本分类功能。经过测试，本文验证了 FedCache 架构在自然语言处理场景下的性能优越性。

关键词：知识蒸馏；个性化联邦学习；自然语言处理；通信效率

Abstract

With the wide application of Edge Intelligence (EI), federated learning has gained more and more attention in order to protect data privacy. However, the currently popular federated learning methods do not guarantee the generalization ability of the model over heterogeneous clients and often have certain performance limitations in practical applications. To solve the problem, Personalized Federated Learning (PFL) has gradually become a research focus in the field of EI. Meanwhile, although existing PFL approaches can support heterogeneous edge devices and improve model performance, most of them are based on parameter-interactive federated learning architectures. This architecture requires a large number of model parameters to be transmitted between the edge device and the server, leading to its excessive communication overhead, which seriously affects the performance of the federated learning system. In contrast, the architecture of cache-driven FedCache updates model parameters by transmitting Logits, which has higher communication efficiency and client-side heterogeneity support, and is able to reduce the communication overhead of federated learning systems while ensuring high model accuracy. However, the architecture is only applicable to some scenarios in Computer Vision (CV) because the ResNet model, the encoder for generating hash vectors, and the method of dividing the dataset used by the client in the FedCache architecture only support some image datasets. In addition, Natural Language Processing (NLP) has an equally significant place in the field of EI and faces a serious risk of data breaches. Therefore, to ensure the performance of NLP models while protecting the privacy and security of text data, the application of FedCache architecture to the field of NLP has significant research value.

So far, no research has been done on the FedCache architecture in NLP scenarios. Therefore, this dissertation re-implemented the FedCache architecture and performed adequate experiments to make it applicable to all PFL tasks in CV and NLP scenarios.

Firstly, this dissertation adjusts the client's local model as well as the hash encoder in the FedCache architecture and designs a division method for NLP data-sets such as IMDB and SST-2. Then, this dissertation fully develops comparative experiments to demonstrate that with the above improvements, the FedCache architecture achieves better performance in NLP scenarios compared to the three classical federated learning architectures, pFedMe, FedAvg,

and MTFL, and improves by an order of magnitude in terms of communication efficiency. Finally, this dissertation applies the NLP model trained by FedCache to the movie review sentiment classification scenario based on Android Studio. The application mainly contains four functional modules, which are the data viewing function, data collection function, movie recommendation function, and model text classification function. After testing, this dissertation verifies the performance superiority of FedCache architecture in NLP scenarios.

KEYWORDS: Knowledge Distillation, PFL, NLP, Communications Efficiency

Contents

1 Introduction

Firstly, this chapter describes the research background and significance of this dissertation, followed by an in-depth discussion of the research history and current situation of knowledge distillation-based federated learning architectures from the relevant studies of federated learning and knowledge distillation respectively. Finally, the main work and organization of this dissertation are explained.

1.1 Background and Significance of the Research

In recent years, big data-driven Artificial Intelligence (AI) technology has been increasingly widely used. With the emergence of massive amounts of data and the powerful ability of machine learning models to fit big data, a new generation of AI technology represented by machine learning has penetrated various industries[7]. Today, the main driver of the explosion of AI applications is deep learning technology, which has produced a large number of phenomenal applications that make life and work more efficient. Despite the rapid development of deep learning technology in the era of big data, there are still many problems with existing technologies. First, existing technologies and applications are dominated by centralized architectures centered on cloud computing platforms, where data is stored and processed centrally. In addition to the high latency and computational overhead of centralized architectures, such architectures have serious data privacy and security issues. If the privacy issues involved are not fully considered and resolved, centralized systems will face risks such as sensitive data leakage and cyber-attacks. Secondly, machine learning algorithms represented by deep learning require a large amount of training data, and in reality, due to industry competition, privacy, and security constraints, the data needs to be stored separately in different organizations, and the integration of distributed data is a huge obstacle for centralized deep learning architectures. This data isolation not only prevents effective collaboration between organizations but also restricts the full use of data value, which constitutes a certain obstacle to the development of the industry and the progress of society[2].

Because of the strict restrictions of data security regulations, machine learning application solutions based on the device side are gradually gaining widespread attention. In this area, federation learning, as an emerging distributed machine learning method, has become a research focus. This training mechanism could protect the user's private data from

being uploaded, reduce the large communication overhead generated by data transmission, and utilize the data generated by device interactions. In addition, the model performance statement is comparable to the performance of models obtained by training raw data from all parties. Thus, this distributed nature makes federated learning more advantageous in terms of privacy protection and data utilization. However, it also makes federated learning face communication inefficiency as well as client heterogeneity in practical applications: Firstly, the clients involved in learning are usually distributed in different geographical locations and limited by the network bandwidth of their respective devices. Especially in practical application scenarios, the number of devices participating in federated learning may be very large, and a large number of devices and the central server need to transmit parameters frequently, which not only occupies a large number of network resources, but also may lead to a significant decrease in communication efficiency. Secondly, in traditional distributed machine learning, the data of each client is generated independently, namely, the data presents a Non-Independent Identical Distribution (Non-IID) data model. This will lead to the accuracy of the model gained from the federated learning architecture will be small or even decreased compared with the local individually trained models.

Aiming at the client heterogeneity problem of federated learning, PFL has gradually become a research focus in the field of federated learning as a good solution to this problem. Personalized federated learning utilizes non-IID data in heterogeneous clients in many different ways, and the mainstream method is based on the parameter balancing algorithm to construct a global model on the server side and update the personalized model on the client side through parameter interaction. Although PFL can improve the model performance of federated learning on heterogeneous clients to some extent, it still faces performance challenges in practical applications because of the large communication overhead.

To make federated learning achievable with lightweight communication overhead while satisfying the good client heterogeneity supportability of federated learning, a knowledge distillation approach, known as federated distillation, is usually added to federated learning. Knowledge distillation is an implementation of knowledge migration that uses the output of a trained teacher model to train a student model so that its classification performance can approach that of the teacher model. This approach applied to federated learning enables Logits-based interaction between server and client while allowing model training to be performed disregarding the characteristics of heterogeneous clients. Thus, federated distillation can satisfy both communication lightness and better client heterogeneity support.

1.2 Research History and Current Status

Federated learning, as an advanced algorithm framework that can effectively protect the privacy of users' original data, needs to integrate the heterogeneous model parameters from each client to a centralized server, but this traditional PFL architecture faces many performance and application problems. In this dissertation, the personalized federated distillation architecture is chosen to tackle the drawbacks of traditional architectures such as inefficient communication and difficulty in handling heterogeneous data. In the following sections, the current development status of PFL architecture and federated distillation technology are introduced respectively.

1.2.1 Related research of personalized federated learning

In federated learning frameworks, the server side, as the training organizer, has access to a global model for use. However, for the client side, the global model does not apply to the client data when the data distribution is too different. In this condition, the concept of PFL gradually emerges. Compared with traditional federated learning that pursues training a global model with generalization performance, PFL provides each participating client with a personalized model adapted to its local data from the client's perspective. There are multiple approaches to PFL to achieve the goal of client model personalization, which can be divided into two broad categories: global model personalization and local model personalization.

(1) Global model personalization

Global model personalization is a method for transforming a global model into a personalized model adapted to the client. Specifically, this class of methods first trains a well-performing global model through traditional federated learning training methods and then provides a personalized model for the client by performing additional local adaptive updating steps to the global model with local data on the client. It can be viewed as an extension of traditional federated learning plus adaptive updating, and this two-stage scheme is regarded as a personalization strategy for federated learning[11; 17].

To train a well-trained global model, most studies utilize data augmentation to mitigate data distribution differences between clients. Zhao, Y., et al[30] proposed a data-sharing strategy that distributes a small amount of global data balanced by class to each client. Their experiments show that by adding a small amount of data, a significant improvement in accuracy may be achieved. Duan, M., et al[5] propose Astraea, a self-balancing federated

learning framework that handles class imbalance by using Z-score-based data expansion and down-sampling of local data. The server requires statistical information about the local data distribution of the clients. In addition, many methods also use regularized loss functions to avoid the model deviating too far from the global model during client-local updates. FedCL[6] uses Elastic Weight Consolidation (EWC) in the field of continuous learning to consider parameter importance in regularized local loss functions[12]. The importance of weights to the global model is estimated on agent data-sets in a central server. They are then transmitted to the client, where parameter update restrictions are enforced to prevent the global model from drastically modifying the important parameters while adapting to the client's local data. This reduces the difference in weights between the local and global models while preserving the knowledge of the global model to improve model generalization performance[28].

Other methods related to rapid model adaptation to the domain can also be borrowed to solve the problem of federated learning domain personalization. Jiang, Y., et al[10] explored the similarity between the classical federated learning algorithm FedAvg and the meta-learning domain algorithm Reptile and pointed out that the federated training process can be understood as the meta-learning training process to a certain extent[23]. Per-FedAvg built a set of meta-learning updating processes under the framework of the federation in the federation learning framework and utilizes the classical meta-learning algorithm MAML to build a set of meta-learning updating processes under the federated framework[6]. This approach is suitable for learning better global model initialization for stronger personalization on local data clients with heterogeneous distributions. However, this approach is computationally expensive. In addition, there are various approaches in the field of migration learning that have been applied to federated learning to make literate global models better adapted to data of clients[3; 13; 26].

(2) Local model personalization

Different from the two-stage training approach of global model personalization, the category of local model personalization directly modifies the objective function of traditional federated learning by no longer searching for only one global model, but providing suitable personalized models for all clients participating in federated learning training. A variety of methods such as multi-task learning and parameter decoupling are included in this category.

Multi-task learning is a task similar to federated learning in terms of setup, which improves generalization by leveraging domain-specific knowledge in the learning task to train a model capable of federating multiple related tasks. Smith, V., et al[23] proposed the MOCHA algorithm for extending the distributed multi-task learning approach to federated learning setups. MOCHA uses the primal-dual formulation to optimize the learning model. The

algorithm solves the communication and systemic problems prevalent in federated learning. Instead of traditional federated learning, which learns a single global model, MOCHA learns a personalized model for each client. While MOCHA improves personalization, it is not suitable for cross-device applications because all clients need to attend each round of federation model training. In addition, FedCurv uses elastic weight curing to prevent catastrophic forgetting when moving across learning tasks[22]. The significance of parameters is estimated using Fisher's information matrix, and a penalty step is performed to retain important parameters. FedAMP is an approach that utilizes an attention mechanism, which enables stronger peer-to-peer collaboration between clients with similar data distributions to enhance performance[8].

The core idea of the parameter decoupling scheme is to divide the model into a shared representation part, which needs to be globally trained for federated learning, and a private personalization part, which is kept locally at the client. The shared representation part acquires the generalization ability and is spliced with the personalization part to obtain a complete personalized model. Arivazhagan, M. G., et al[1] first proposed to use of a "base layer" plus a "personalization layer" scheme for model layering and achieved good results. Similarly but differently, the LG-FedAvg approach keeps the global representation part local and shares the parameters of the learned classifier part[14]. This method improves the generalization ability of the client model, but the overall performance of the model is more constrained by the local data.

1.2.2 Related research of federated knowledge distillation

Although PFL can improve the accuracy of models trained by federated learning, most PFL architectures are based on parameter interactions incur huge communication overhead, and don't support the case of complete client heterogeneity. Federated knowledge distillation, on the other hand, can support both communication lightness and complete client heterogeneity, making this area a current research focus. Federated Knowledge Distillation treats the local client model as the student model, the server model or the sum of clients other than the local client as the teacher model, and the categorical scores or logical vector outputs as knowledge. Federated knowledge distillation extracts knowledge from a complex teacher network into a simple student network without the need for the teacher network and student model to be aligned, which can provide a greater degree of flexibility to the client.

A typical approach is to use sample data information or agent data-sets to minimize the

logical vector outputs from the teacher model and the student model. FD computes the average of the outputs of each category on each client and uses the global average of the outputs of each category to guide the learning of the local client[9]. The federated learning server in FedDF constructs several different prototypical models, each of them representing the clients with the same model architecture. For each iteration, the client first updates the model parameters using local data and sends the updated parameters to the server[15]. The server side averages the received model parameters and generates logical vector outputs for each student model used to train the server on the unlabeled public data-set. One integrated distillation is performed for each model of each architecture to get the aggregated model parameters. The main limitation of this approach is its dependence on proxy datasets, making the choice of proxy datasets play a key role in distillation performance. FEDGEN avoids proxy datasets and uses a data-free approach for knowledge distillation[30]. The algorithm requires the client to be able to provide the server side with the distribution of its sample labels, that is, the client shares the sample labels. The server side learns a mapping from sample labels to samples based on the global distribution of sample labels so that the generated samples are consistent with the global data distribution. After that, each client can use the newly generated samples for knowledge distillation.

The most representative of the current research on FedCache knowledge distillation is the FedCache architecture proposed by Wu, Z., et al[24], which first establishes the relationship between all client sample data in the knowledge cache, then stores the logits output by the clients and assigns the logits most relevant to each client to it. Finally, each client performs knowledge distillation based on the assigned logits to optimize the local model. Although the FedCache architecture can better support client heterogeneity and keep communication lightweight, the architecture is only suitable for CV application scenarios, and there is still a large research space in extending its generalization.

1.3 Main Contents of This Research

Considering the limitation of the single application scenario of FedCache, the research goal of this dissertation is to build a knowledge cache-driven PFL architecture generalized to NLP scenarios based on FedCache using text sentiment classification as a research scenario and utilizing the relevant theories and techniques of deep learning and knowledge distillation. Then, the Internet Movie Database (IMDB) data-set and Stanford Sentiment Treebank (SST-2) data-set are used as examples to simulate the client heterogeneity in real applications, and

based on this, comparative experiments are made to verify the superiority of the new PFL architecture implemented in this dissertation in terms of communication efficiency and support for client heterogeneity in comparison with the traditional PFL architectures, and meanwhile, to provide the users with the corresponding system services that are applicable to all other NLP data-sets.

Combining the problems of current research on federated learning architectures described in 1.2 and the research objectives of this dissertation, the research work of this dissertation consists of the following four main aspects:

(1) Partitioning of example datasets for data heterogeneous federal learning scenarios

Data heterogeneous federated learning requires the data on the clients to follow a non-independent distribution, namely, the datasets on each client are different in terms of quantity, feature distribution, label distribution, and so on, to more realistically simulate the real-world scenarios where edge devices with different arithmetic power participate in federated learning with text datasets with different features and perform the text categorization task. The PFL architecture implemented in this dissertation can satisfy all the data-sets under the general NLP scenario, in order to revert to the actual application scenarios, and enable the subsequent experiments to realistically reflect the actual performance of the PFL architecture, this dissertation selects the IMDB data-set and the SST-2 data-set as the example data-sets, and based on the Dirichlet probability distribution algorithm, assign these two data-sets randomly to different clients. To ensure that the data-set assigned to each client is not too little, this dissertation specifies the minimum sample size for each client, and if the sample size is smaller than this minimum limit, all the clients will re-randomly assign the data until each client meets the basic requirements of data assignment. In addition, the IMDB data-set has large differences in the length of the original movie reviews, the distribution of movie titles, and the sources of the reviews, which may lead to poor detection performance of models trained in the same client in movie review scenarios. Therefore, this dissertation carries out data cleaning and other work based on the original IMDB data-set to improve the training and testing effect of the text classification model in the same client.

(2) Construction of text classification models

Based on the relevant theoretical knowledge of the pre-trained language model RNN, this dissertation builds text classification models LSTM and GRU for each client. These two models are evenly distributed in the overall architecture, and each client is assigned one of them to reach the purpose of heterogeneous federated learning under real applications. This dissertation will first define the neural network hierarchies of the LSTM and GRU models as

the long and short-term memory network and the gated recurrent unit network, respectively, and these network layers determine how the models process the input data as well as the dimensionality of the hidden states. The input data is pre-processed and fed into the LSTM as well as GRU layers respectively. Then, the hidden state of the last time step is extracted from the output and transformed through the fully connected layer, and the final output is obtained. This output is not processed by the soft-max function and is the predicted score for positive and negative comments, the logits generated by each client after the training process will be uploaded to the server during the communication phase.

(3) Implementation of a PFL architecture based on knowledge cache

In this dissertation, by constructing a knowledge cache for federated distillation, the communication overhead required for federated learning is drastically reduced, and the training effect of federated learning in data heterogeneity as well as model heterogeneity scenarios is effectively improved. The knowledge cache plays a key role in the architecture by first collecting the hash values of the individual client samples and using this to build relationships between the sample indexes. After that, the knowledge cache collects the logits and corresponding sample indexes uploaded by each client after the training round, and the client downloads the R closest logits of its samples from the knowledge cache to its instructor's knowledge cache according to the HNSW algorithm as a way to distill knowledge and complete the optimization of the client's local model[16]. Throughout the process, the architecture achieves lightweight federated learning because of the knowledge distillation approach to model training, and in doing so outperforms the training results of model parameter interaction-based federated learning architectures. At the same time, the architecture supports clients to use non-independent and co-distributed datasets, which allows each client to communicate with the knowledge cache asynchronously, a feature that not only enables the PFL architecture to achieve better model training in heterogeneous scenarios but also improves the efficiency of the communication between the client and the server.

(4) Implementation of movie review sentiment classification application

In order to systematically integrate the work of this dissertation, and in order to facilitate the presentation of the experimental results and data in a visual way, and to provide the corresponding film review classification service for users in need, this dissertation adopts the Kotlin language and Android Studio to develop a film review classification system, which is equipped with the functions of data collection, data viewing, data processing, model prediction, and so on.

1.4 Structure of This Dissertation

This dissertation consists of six chapters, which are described below:

The first chapter is the introduction. Firstly, the research background and significance of this topic are introduced in detail, and secondly, the current research status of federated learning and knowledge distillation in related fields so far are introduced respectively. The main sequence of presentations on the state of the art of federated learning research is as follows: practical applications of federated learning in industry, background, and significance of federated learning as a research focus, classical architectures and open-source platforms for federated learning, research related to solving the problem of high communication overheads in federated learning, and research related to solving the problem of heterogeneity of data and models on the client side in federated learning. The main order of presentation of the current state of knowledge distillation research is research on the application of knowledge distillation in federated learning and research on the application of knowledge distillation in the field of NLP. Finally, the main research work and chapter arrangement of this dissertation are briefly described.

The second chapter is about related theories and key technologies. About NLP, firstly, the basic background knowledge of text categorization is introduced, and secondly, the relevant theories about recurrent neural networks are briefly described. Next, the federated learning architecture targeted by this topic is introduced, firstly, a general overview of federated learning is given, then the system architecture of federated learning and the workflow of each component are sorted out, followed by the introduction of the optimization problem and federated averaging algorithms in federated learning, then the non-independent homogeneous distribution problem in federated learning investigated by this topic is introduced and demonstrated, and finally a brief overview of the process of partitioning non-IID data sets based on the Dirichlet to divide the non-IID data-set. On the content of knowledge distillation, firstly, model response-based knowledge distillation is introduced, and then feature-based knowledge distillation techniques are summarized.

The third chapter is based on a knowledge cache-driven PFL architecture. Firstly, it introduces the overall system design idea, then analyses the design method of knowledge cache architecture in the system, and introduces the cache-driven personalized knowledge distillation technique based on the cache. Finally, it is based on the core of the cache-driven PFL system architecture, which theoretically and intuitively analyses the FedCache architecture algorithms against the general FedAvg algorithms, and demonstrates the

algorithmic process of personalized model knowledge distillation and optimization by a single client in federated learning using local data.

The fourth chapter is the experimental design and performance analysis. Firstly, it introduces the basic settings of the experiment in terms of data-set selection, data-set preprocessing methods, non-independently and identically distributed data-set construction, and model structure, and analyzes the impact of different parameter configurations on the model training effect. Secondly, the FedCache architecture is compared with FedAvg, MTFL, and pFedMe architectures through experiments, highlighting the superiority of FedCache architecture in solving the problem of non-independent and homogeneous distribution of client data, etc., and demonstrating the significant role of FedCache in improving the performance and communication efficiency of PFL architectures. Finally, it is demonstrated that the FedCache architecture applies to the image processing domain, but is also universal in other domains.

The fifth chapter is the design and implementation of a film review sentiment classification application. Firstly, it presents the requirement analysis of the film review classification application, the background of the application's requirements as well as the functional requirements. Next, the architectural design of the application and the design of each functional module are presented to highlight the superior performance of the application in performing text classification tasks. Then it introduces the implementation method of the application, and finally elaborates the functions of each module and shows the test results of the corresponding functions of the module.

The last chapter is the conclusion and prospects for future work. This chapter mainly concludes and analyzes the research situation and research results of this dissertation, points out the shortcomings of this dissertation, and finally, based on the results already achieved, looks forward to the future where the research can be continued and where it can be improved.

2 Relevant Theories and Key Technologies

The introduction later in this chapter provides an overview of the technical background, including text classification, deep learning fundamentals, text representation (word vectors and pre-trained models), and recurrent neural networks. It makes this chapter clearer.

2.1 Introduction

This chapter introduces the technical background knowledge, firstly, it introduces the relevant basics of text classification and deep learning, including text representation methods and recurrent neural networks, and describes the word vector model and pre-trained model of language in text representation methods. Next, this chapter introduces the basic concepts of federated learning, the federated learning system, and the implementation process, and takes the FedAvg architecture and pFedMe architecture as examples to address PFL architecture specifically. Then, this chapter will describe the non-IID problem in federated learning and introduce the method of dividing non-IID data-sets with Dirichlet. Finally, the chapter will introduce two knowledge distillation methods and conclude the chapter.

2.2 Text Representation and Deep Learning Techniques

This section first introduces two text representation methods, namely the word vector model and the pre-trained model of language, and describes the characteristics of the above methods from the aspects of both usage and principle, respectively. Then, this section will specifically introduce the features of recurrent neural networks and their usage.

2.2.1 Methods of text representation

In this section, it will introduce the features of the word vector model and pre-trained model in terms of usage and principle, respectively.

(1) Word vector model

The main goal of NLP is to enable computers to handle various language-related tasks like human beings, such as analyzing the emotions contained in the content of a certain text, correcting typos in a certain text, and predicting what is missing in the middle of a given context. However, computers cannot understand text, and by directly inputting text into a

computer, the computer cannot be informed of the meaning of its references. Therefore, before using computers to accomplish tasks related to NLP, it is necessary to convert the input textual data into digital symbols that can be understood by computers in advance so that the computers can understand them. For the input text data, the computer will first use the relevant word splitter to cut the text into characters or words and then map each character or word into a real number vector that the computer can understand, this vector form is called a word vector, also known as word embedding. With word vectors, computers can accomplish tasks related to NLP. There are many different ways to represent word vectors, and the commonly used methods are the bag-of-words model, TF-IDF, and Word2vec.

Bag-of-words modeling is one of the most commonly used methods in NLP to transform text into real number vectors. It is a model based on word frequency, which maps each word appearing in a collection of text into a specific numerical ID, obtaining a vocabulary of size N; after that, for each text, it is viewed as a vector of size N, counting the frequency of each word appearing in the text, searching for the ID number of the word in the corresponding vocabulary, and marking the frequency of the word in the corresponding mark of the vector, and marking 0 in the other positions. Bag-of-words modeling can simply and quickly process document data, as well as missing data, by transforming text of different lengths into vectors of the same dimensions. After being transformed into the corresponding vectors, the computer can then perform subsequent operations. However, the disadvantages of this method are also obvious, one of the disadvantages is that as the number of documents increases, the size of the vocabulary list will also increase, so the dimension of the final generated vector will also increase, and thus the sparsity of the vector increases. Another and its biggest drawback is that it cannot represent the semantic and syntactic relationships between words, for example, it does not take into account word collocations, word roots, word derivations, etc., so the accuracy of using this method to carry out tasks related to NLP is not very high. But it is - a fast feature extraction method that can effectively convert text into feature vectors for tasks such as text categorization and sentiment analysis.

TF-IDF is a weighting technique commonly used in the field of text mining and information retrieval, which can be used to compute the relative importance of words in all documents in a collection of documents. TF denotes the word frequency, namely, the number of times a word occurs in a document; IDF denotes the inverse document frequency, namely, the fewer documents that contain the word if it is used as a measure of how well a word or phrase is differentiated in the whole set of documents. Its basic idea is that the importance of a word or phrase in a document is directly proportional to its frequency of occurrence in the

document, but at the same time will be inversely proportional to its frequency of occurrence in the document collection, if a word or phrase occurs in a document with a particularly high frequency, and at the same time almost does not occur in other documents, then the word or phrase has a great ability to distinguish when converted into numerical features to change the words which have a greater value of the weights.

Both the bag-of-words model and TF-IDF are based on the statistical characteristics of the text to vectorize the text, and they do not consider the contextual relationship of the words in the text and the textual semantics when encoding the vectors of the text, so the obtained vectors lack the semantic information of the words. Therefore, a neural network is utilized to generate a model of word vectors, namely Word2vec. It learns words in the text by using neural networks and maps these words into fixed low-dimensional dense vectors that are not only recognizable by the computer but also contain semantic features of the words. Word2vec has two models, CBOW and Skip-Gram, and the network architecture diagrams for both models are shown in Figure 2-1 below.

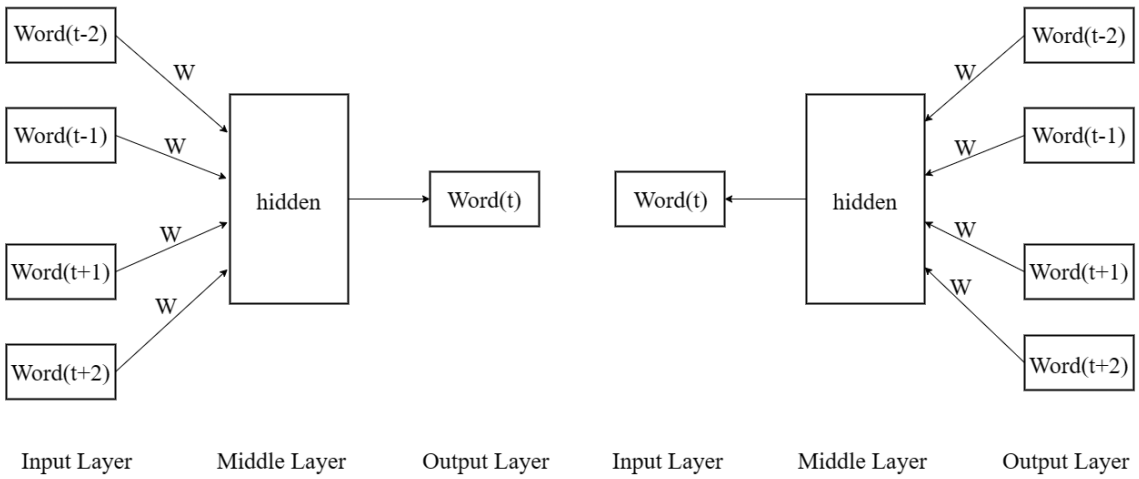


Figure 2-1 Network architecture of CBOW and Skip-Gram

With the word vectors generated by CBOW and Skip-Gram, it can effectively capture the relationship between words, which improves the computer's ability to understand the text and provides great help for text analysis and semantic understanding.

(2) Pre-trained language models

In recent years, supported by deep learning and big data, NLP technology has been developing rapidly, and pre-trained language modeling has brought NLP into a new stage, and has also gained widespread attention in the industrial community. By pre-training on a large amount of unlabeled data and then fine-tuning the dataset for a given task, NLP tasks can be

solved without relying on a lot of manual parameter tuning. Pre-training belongs to the research category of migration learning, compared to existing neural networks in the training, generally first random initialization of the parameters in the network, and then use stochastic gradient descent and other optimization algorithms to continuously optimize the model parameters, and then use the back-propagation algorithm to learn the model. The idea of pre-training, on the other hand, is that the model parameters are no longer randomly initialized, but rather a set of model parameters are obtained by pre-training unsupervised learning on a large-scale corpus, using which the model is initialized and then fine-tuned on a given dataset by adding a network structure at the top of the model for a specific NLP task.

BERT is a new model in NLP proposed by Google in 2018, which is considered to be one of the most influential research in NLP. BERT has achieved very strong performance in NLP tasks. The architecture of BERT is based on two main training modes, namely, Mask Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM training, for each sentence in the training corpus, the model randomly replaces 15% of the words in the input sentences with the MASK flag and then trains the model to predict the words that have been replaced with MASK, forcing the model to predict the words that have been replaced with MASK through contextual word information due to the use of a bi-directional Transformer encoder in the model architecture. This is in contrast to ordinary recurrent neural networks or pre-trained language models generated based on unidirectional Transformer encoders such as GPT, which can only predict with unidirectional information, whereas BERT allows the model to learn a bidirectional representation of the sentence[19]. In NSP training, the input to the model is two spliced sentences, which may be coherent sentences or incoherent sentences randomly pieced together from different texts, and the training goal of the model is to predict whether the two spliced sentences are coherent or not.

With the two stages of training, MLM and NSP, BERT can learn not only the deep semantics at the word level but also semantics at the sentence level. The pre-trained model is then applied to downstream tasks. Because the model is initially already highly capable of language learning and then fine-tuned using feature data-sets relevant to the downstream task, the model can achieve advanced performance in the relevant task. Utilizing advanced transfer learning methods, the BERT model has achieved a breakthrough in the field of natural language processing, which not only improves the performance and accuracy of the model, but also provides powerful support for a variety of language processing tasks, and greatly promotes the research and application in this field[21].

2.2.2 Recurrent neural network

The recurrent neural network is a kind of deep neural network, which uses the recurrent structure to process the input sequence to simulate the human memory function. The basic principle of RNN is to save the features of the previous input sequence through the recurrent structure, and then combine the information of the current input sequence, add the current input information into the saved features of the previous input sequence, and do the updating of the features retained in the recurrent structure, to realize the effect of memorizing the input information, and thus be able to deal with the dependency relationship between variable length information. In NLP, text is viewed as a string of discrete words, and the back-and-forth relationship between words contains semantic information, and a model that learns the relationship between words can learn the semantic information between words to a large extent. This cyclic structure of RNN makes it the most commonly used traditional deep learning model for NLP tasks. The structure of the RNN is shown in Figure 2-2 below.

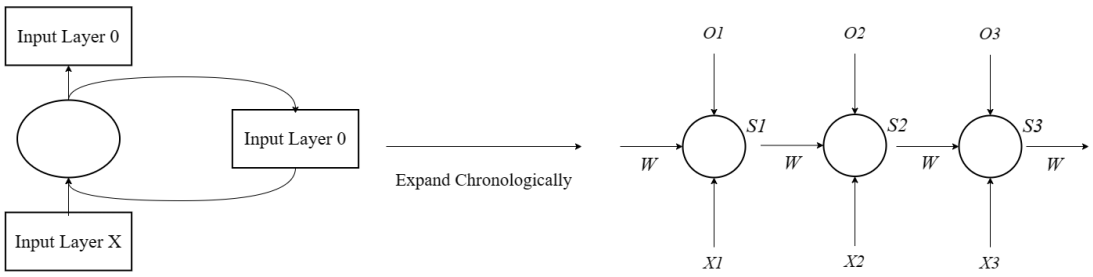


Figure 2-2 RNN structure

Since the RNN retains the information of its input sequence through a cyclic structure while updating the retained information in conjunction with the input of the current moment, as a way to ensure the memorization of the information. However, when the length of the input sequence is too large, the previous input information may be forgotten, and at the same time, when using the backpropagation algorithm to do gradient updating of the network, the model may also face the problem of disappearing gradient, which leads to the network parameters not getting updated and learning cannot take place. In order to solve the gradient vanishing problem of RNN, models such as LSTM have become the focus of current research on RNN. The LSTM model implements its memory mechanism through input gates, forgetting gates, output gates, and memory cells, where input gates are mainly used to control which of the information input at the current moment will be inputted; forgetting gates are mainly used to decide which information should be retained in the memory cells and which

should be forgotten; and input gates determine the content of the output of the memory cells. The memory updating steps of LSTM can be categorized into: the input step, forgetting step, output step, and saving step. In the input step, the network receives new inputs through input gates; in the forgetting step, the network controls the saving and forgetting of memories through forgetting gates; in the output step, the network decides the outputs through output gates; and in the saving step, the network saves the new memories to the internal state. LSTM solves the problem of gradient vanishing by its unique memory updating step.

2.3 Federated Learning

This section will introduce the usage and implementation of federated learning, focus on the characteristics and significance of PFL architecture with FedAvg and pFedMe as examples, and specify the optimization process and performance strength of PFL. Finally, this section will introduce the classification and segmentation methods of non-IID data in federated learning architecture.

2.3.1 Overview of federated learning

Various industries have already witnessed the great value brought by the development of AI, but with the increasing volume of data, numerous data flows on the Internet, which brings convenience but also increases the probability of danger. In 2016, Facebook, which has a huge customer base, did not have a comprehensive and reasonable protection policy in terms of security and privacy, and privately shared user data with third parties, which ultimately led to the misuse of as many as 87 million user profiles. Facebook was heavily fined for this incident and promised to sound its privacy system within the enterprise, and this incident once became the focus of the Internet industry, and user data This incident once became the focus of the Internet industry, and the issue of user data privacy protection was pushed into the limelight.

The global emphasis on user privacy has increased dramatically, with regulators introducing detailed laws and regulations to govern the management and use of data. For example, the General Data Protection Regulation (GDPR), which was implemented in the European Union in 2018, has imposed strict constraints on the collection and processing of data. Strictly regulating the use and dissemination channels of data protects user privacy on the one hand, but on the other hand, creates difficulties for the integration of data resources.

The AI industry faces a data dilemma, and how to enable federated training of data while adhering to privacy-protecting regulations is the primary challenge for AI development today.

Federated learning is a distributed machine learning algorithm built on a multi-party secure computing framework, and its powerful application value lies in its ability to break down data barriers between enterprises due to data privacy protection, enabling private and secure collaboration on multi-party data. The concept of federated learning was first introduced by Google in 2016 and was originally used to solve the problem of updating models locally by end users of devices such as mobile phones or tablets. Mobile devices are equipped with powerful sensors and allow for rich user interaction, thus mobile devices can capture a large amount of data and information. This information can provide a powerful impetus for intelligent optimization of applications in the device. As the amount of user data increases, the model becomes more and more usable, but storing multiple user data-sets for training in a centralized manner is difficult to achieve due to privacy protection constraints. With the ability to train a shared model by aggregating updates from each user's local computation through a centrally coordinated server while keeping the training data on the mobile device side, federated learning separates model training from the need for direct access to the raw training data, significantly reducing privacy and security risks. Google designed a secure distributed computing environment for business-to-consumer (B2C) federated learning applications. Subsequently, federated learning was further developed in China, and in 2019, Yang, Q., et al[27] defined federated learning more comprehensively and well, and introduced the application of federated learning on business-to-business (B2B) cross-border cooperation.

2.3.2 Non-IID data in federated learning

In this section, the main types of non-IID data and the reasons for their generation are described, and based on this, the process of generating non-IID data and the methods of dividing them are carried out.

(1) Categories of non-IID data

The concept of Independent Identical Distribution (IID) is often used as a premise in major learning and training scenarios, but the actual data are in many non-IID ways. This subsection will first briefly classify the commonly used non-IID data due to different clients. The most common reason this happens is because the data to be used for federated learning is generated by different clients, which in turn are used by each user based on different times

and locations. Each of these elements is a sufficient reason for non-IID data to be generated.

To facilitate the following exposition, this paragraph starts with some basic facts, which can be described by assuming that there is a supervised task with feature x and label y. The following is an illustration of the statistical model of federated learning. A general sampling of statistical models on federated learning is as follows: a data distribution $(x, y) \sim P_i(x, y)$ is described based on the local data distribution of this client. To better distinguish the different cases of non-IID, $P_i(x, y)$ can be written $P_i(x \mid y) P_i(y)$ and $P_i(y \mid x) P_i(x)$, while facilitating the interpretation of the different distributions that follow. Based on the different distributions of features and labels, they can be classified into the following four cases:

- Deviation of feature distribution: It refers to the situation that even though the conditional distribution $P_i(y \mid x)$ of different nodes is the same, the feature distribution $P_i(x)$ is different because of the different nodes. For example, in the field of handwriting recognition, even if it is the same character, the feature distribution will be different due to the thickness of the font, the compactness or looseness of the structure, and the correctness or skewness of the font.
- Label distribution bias: This refers to the situation that even if $P_i(y \mid x)$ of each node is the same, the label distribution $P_i(y)$ is different because of different nodes. Such as training related to unique geographical features, for example, in the feature recognition system related to protected plants, Banksia in Australia and Peony in China are unique data with extremely uneven label distribution.
- Same labels, different features: Specifically, the $P(y)$ of each node is the same, but the $P_i(x \mid y)$ of the conditional distribution of each node is different. This situation is actually very common, there are many cultures on earth, even the same thing will have many different forms.
- Same features, different labels: Specifically, even if the characteristic distribution of each node $P(x)$ is the same, there is its conditional distribution $P_i(y \mid x)$ due to different nodes. Here is a popular example, the most common example in life is actually reading. Different people will often give different interpretations after reading the same article.

Real-life federated learning data-sets are not independently and identically distributed and may be one of the above or a mixture of many. To test the algorithms and make a specific measure of the degree of heterogeneity of the client, and to better understand the nature of real-world non-IID data, it is necessary to carry out a controllable but realistic non-IID data

reconstruction for it[29]. However, the current experiments on federated learning for non-IID data often focus on non-IID data reconstruction or label distribution as a scenario.

(2) Non-IID data-set partitioning based on Dirichlet distribution

The Dirichlet distribution, first proposed by statistics, is a typical high-dimensional continuous distribution, generally denoted as $Dir(\alpha)$, where α denotes a vector of positive real numbers. The Dirichlet distribution is called "the distribution of probability distributions" because its parameter α also obeys a probability distribution[20].

The formalization of the Dirichlet distribution is defined as follows: Make $Y_k = [y_1, y_2, ..., y_k]$ a vector of order k, where for $\forall i \in [1, 2, \cdots, k]$, there is $y_i > 0$ and $\sum_{i=1}^{k} y_i = 1$; And for k-order vector $a^k = [a_1, a_2, ..., a_k]$, there is always $a_i > 0$ in $\forall i \in [1, 2, \cdots, k]$. Thus, the probability distribution of Dirichlet is determined as:

$$f(y^k) = \frac{\Gamma(a_0)}{\Pi_{i=1}^{k}\Gamma(a_i)} \prod_{i=1}^{k} y_i^{a_i - 1} \tag{1}$$

Where, $a_0 = \sum_{i=0}^{k} a_i$, $y_i > 0$ and $y_k = 1 - y_1 - \cdots - y_{k-1}$, $\Gamma(.)$ is Gamma function, its general formula can be expressed as:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} \exp(-t) dt. \tag{2}$$

Where, in the data heterogeneous federated learning scenario, one can sample the public IID data-set by Dirichlet distribution to obtain the non-IID data-set. In the original data-set containing the N class labels, the training set of each local model passes through a probability parameter vector $q(q_i \geq 0, i \in [1, N])$ to control the sampling of raw data samples. The parameter profile, the rate vector q satisfies the Dirichlet distribution, can be written as:

$$q \sim Dir(\alpha\rho) \tag{3}$$

Where, the probability vector ρ represents the prior knowledge of the N class label class, while the parameter α is a continuous scalar that controls the consistency of the training data sample of each local model, equivalent to controlling the degree of generated data distribution skewed. When $\alpha \to \infty$, the distribution of the training data of the generated local model is consistent with the prior distribution of the original data-set; when $\alpha \to 0$, the sample data obtained by each local model will only be randomly sampled from a certain class in the original data-set, thus ensuring that the data distribution of each local model satisfies the non-IID distribution.

2.3.3 Federated learning system and implementation process

The main members of the federated learning system are the participant clients and servers. In the federated learning system, the participants and the servers can communicate directly without the intervention of a third party, thus enhancing the security of the whole system. The system architecture of horizontal federated learning is described in detail in the following, and the system architecture of horizontal federated learning is shown in Figure 2-3, taking a server with N client as an example.

Usually, the server in federated learning is called the cloud, and the whole of the participant's client is called the edge device. As can be seen in Figure 2-3 below, the difference between federated learning and traditional data "centralized" co-training systems is that the original data-set is always local to the client, and only the encrypted gradient local to the client is involved in the process of transmitting information to the edge of the cloud.

The implementation process of the federated learning system can be summarized in four main phases, as shown in Figures 2-3, from labels 1 to 4:

(1) Each client updates the local model;

(2) Each client uploads an encrypted gradient to the server;

(3) The server performs a secure aggregation of the encryption gradients across clients and subsequently updates the global shared model;

(4) The server sends down the global shared model to each client.

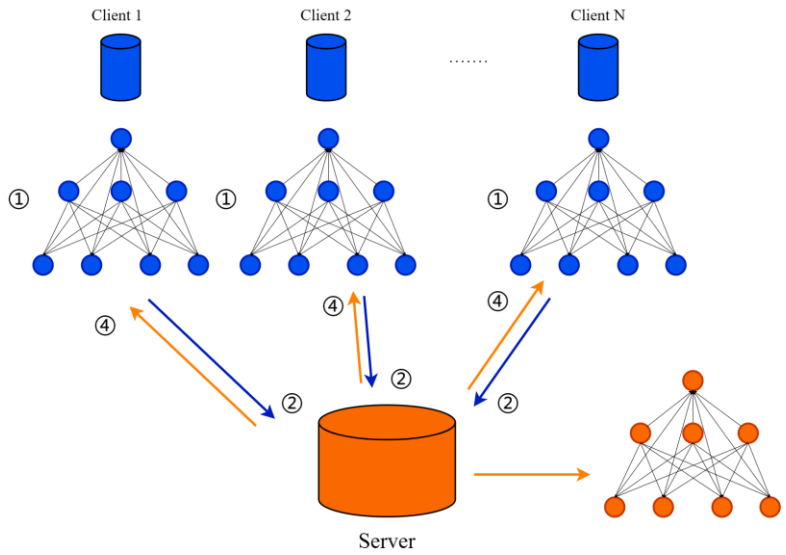


Figure 2-3 Horizontal federated learning architecture

The processes are repeated until the stopping condition of the system is satisfied, and

finally, the server gets a sharing model of all clients and sends it to each client.

From the implementation process of the federated learning system, it can be seen that federated learning can guarantee that the participants achieve the sharing of gradient information. The server security aggregation process is usually performed by calculating a weighted average of the parties, so the client encryption gradient uploading, server encryption gradient aggregation, and global shared model downstream involved in the horizontal federated learning process are collectively referred to as gradient averaging methods. At the same time each participant can share the encryption model in addition to the encryption gradient, namely, the method of model averaging is used, and the above two methods are collectively known as the federated averaging algorithm.

2.3.4 Personalized federated learning architectures and their features

Modern edge devices have access to large amounts of data suitable for training their local models, and trained and optimized local models can improve the user experience on the device. For example, mobile devices can obtain a rich corpus through the network and train a high-performance language model to improve speech recognition and text input on mobile devices; not coincidentally, an image model can automatically recognize the target to be processed in an image, and this rich data often belongs to the sensitive information of individual users, which can hinder the training of models in data centers. To address such problems; in 2017, Smith, V., et al[23] proposed to enable each node to adopt a knowledge-sharing MTFL architecture to directly train heterogeneous models for PFL; in 2020, Dinh, T., et al[4] built on FedAvg for device, data and model-level personalization to reduce heterogeneity and obtain high-quality personalized models for each device and proposed the pFedMe PFL architecture. The following section specifies the implementation and characteristics of federated learning architectures.

(1) Mean-based framework for federated learning: FedAvg

FedAvg was the first architecture formally proposed in the federated learning field and laid the foundation for federated learning. McMahan, H. B., et al[18] of Google proposed the Federated Averaging Algorithm (abbreviated as FedAvg) that can be applied to horizontal federated learning systems. The goal of the FedAvg architecture is to solve the data leakage and privacy and security problems associated with traditional centralized model training based on data transmission, but at the same time, this architecture is also limited by the high communication overhead, which is also one of the issues that this dissertation focuses on. The

algorithmic flow of the FedAvg architecture will be given below, and its pseudo-code is shown in Algorithm 1[18].

Table 2-1 Algorithm 1[18]

Algorithm 1: Federated Averaging

Server executes: initialize w_0 **for** each $t = 1, 2, \ldots$ round **do**
  S_t = (random set of $\max(K,1)$ clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow$ ClientUpdate(k, w_t)
  end for
  $w_{t+1}^k \leftarrow \sum_{t=1}^{\frac{k}{n}} w_{t+1}^k$
end for
ClientUpdate(k, w): // *Executed on client k*
for each local epoch i from 1 to **do**
  batches $\leftarrow$ (data k split into batches of size)
  for batch b in batches **do**
  $w \leftarrow w - \eta(w; b)$
    end for
end for
return w to server = 0

The architecture of FedAvg is based on the FedSGD client model training and optimization algorithm, namely, gradient computation is performed once per round on a randomly selected client. This approach is computationally efficient but requires a lot of training to obtain a good model. In the federated learning architecture setup, in order to involve more clients, FedAvg uses a large number of synchronized SGDs, and in each round, FedAvg selects the clients that account for a proportion C of the total number of clients and calculates the gradient of the data loss on this batch of clients. Here, C refers to the ratio of the number of clients participating in the federated aggregation compared to the total number of clients, and also B refers to the batch size of the client's local training setup, E refers to the number of times local training between two federation training. Meanwhile, C controls the global batch size, and when $C = E = 1$, $B = \infty$, the FedAvg architecture algorithm is equivalent to the FedSGD algorithm, namely, all the data-sets of the client are used for

training each time, with the number of local training being 1, and then aggregated.

The specific algorithm is: The FedAvg architecture selects a small random fraction of clients with proportion C at the beginning of each training round, and the server sends the current global model parameters to each of the selected clients, based on the global model parameters as well as the local data-set, each selected client starts to perform the local computation, and then sends the updates to the server. Next, the server then assigns the received updates to the global parameters and repeats the process over and over again. A formal description of the FedAvg algorithm is provided below, specifically, a schematic of the FedAvg architecture algorithm is shown in Figure 2-4.

- The server initializes the training model ω_0;
- To perform the t-th round of model fusion, the server selects the number of clients from all clients randomly and broadcasts the model to the selected clients;
- The client takes the accepted model as an initialized model, trains it using local data, and then uploads the new parameters ω_{t+1}^{k} to the server;
- The server aggregates the received models and averages the k models to get ω_{t+1};
- Perform the next round of model fusion and repeat the process in turn.

Numerous current federated learning-related works are based on the federated averaging algorithm, which, as the earliest classical algorithm for federated optimization, has been studied and applied in a large number of ways, including the discussion of the convergence of the federated averaging algorithm and the proposal of some improved algorithms.

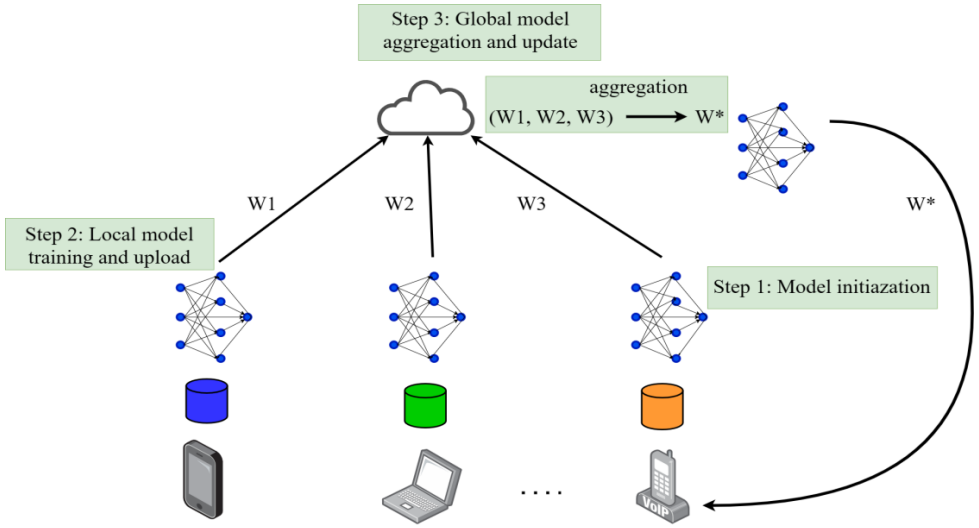


Figure 2-4 Schematic flow of FedAvg algorithm

In data center optimization, communication costs are relatively small, and computation

costs dominate, but modern edge devices have relatively fast processors, so the system is trained efficiently. In contrast, the communication cost of FedAvg is dominated by the fact that the server side in real scenarios usually needs to coordinate hundreds of edge devices for federated learning, while the communication cost of FedAvg is positively correlated with the number of clients in the system, which will make the communication cost of the whole model training process grow rapidly and lead to the system's training efficiency being limited by the network bandwidth[25]. Meanwhile, FedAvg faces the problem of client heterogeneity, and since FedAvg focuses on obtaining high-quality global models by learning local data from all participating clients in a distributed manner, it is unable to capture personal information about each device, which leads to degradation of inference or classification performance. In addition, FedAvg requires all participating devices to agree on a common model for collaborative training, which is unrealistic in real-world complex IoT applications.

In summary, although FedAvg has the advantage of data privacy protection, its parameter interaction-based training approach will incur significant communication costs. At the same time, there are challenges of data heterogeneity and model heterogeneity in each client, which need to be further improved to meet the needs of real-world application scenarios.

(2) Architecture with parametric equalization for federated learning: pFedMe

To address the heterogeneity challenge in federated learning, Dinh, T., et al[4] proposed the pFedMe architecture, a PFL scheme that introduces Moreau envelopes optimization based on client-side loss functions. With this solution, clients can not only build global models like FedAvg, but also optimize their personalized models using global models. Analyzed from a geometric point of view, the global model in this scheme can be viewed as a central point agreed upon by all clients, while the personalized model is a point constructed by clients following different directions based on their heterogeneous data distribution. In summary, the pFedMe algorithm updates the global model in a similar manner to standard federated learning algorithms such as FedAvg, while optimizing the personalized model in parallel with lower complexity, providing an effective solution to the challenge of federated learning heterogeneity.

Unlike the classical federated learning algorithm FedAvg, the pFedMe algorithm uses a regularized loss function with a two-layer structural paradigm for each client. The algorithm determines the global model ω by using data aggregation from multiple clients in the external layer and optimizes the personalized model ω for the data distribution of client i at a distance from ω in the internal layer. Further, the two-layer structural paradigm loss function can be expressed as follows:

$$\min_{w\in\mathbb{R}^d}\{F(w) \coloneqq \frac{1}{N}\sum_{i=1}^{N} F_i(w)\}, \text{where } F_i(w) = \min_{\theta_i\in\mathbb{R}^d}\{ f_i(\theta_i) + \frac{\lambda}{2} || \theta_i - w ||^2\} \quad (4)$$

Where, $F_i(\omega)$ is defined as the Moreau envelope function in Equation (4), also known as the neighborhood operator $\hat{\theta}_i(\omega)$, which represents the loss function on client i for model parameter ω and is defined as follows:

The loss function $f_i(\theta)$ is used to measure the performance of the model on the i-th client's data, and the regularization term is used to control the gap between θ_i and global model parameters ω. λ is the regularization factor, which is used to balance the effects of the loss function and the regularization term, with a larger λ allowing for unreliable data from rich data aggregation and a smaller λ helping the client to get enough data for personalized prioritization. The purpose of the whole $F_i(\omega)$ function is to find an optimal ω such that the local loss plus the regularization term is minimized at the i-th client. By minimizing $F_i(\omega)$, clients can find an optimal model parameter θ_i that both adapts to the local data features and maintains a certain degree of heterogeneity with the global model parameter ω, so that each client is able to train a personalized model based on its data distribution, while the regularization term ensures that these models do not deviate too far from the global consensus.

This form of optimization helps to keep the model parameters dispersed while driving the model parameters closer to the global model parameters ω, as a way to improve the overall performance of the model on all client data. Table 2-2 shows the complete pseudo-code for the pFedMe algorithm.

The pFedMe algorithm, although advantageous in PFL, is still deficient in many ways: first, although the pFedMe algorithm allows different clients can have different model structures or configurations, it still needs to maintain consistency to a certain extent to be able to interact and update with the global model, and thus the pFedMe algorithm only allows partial heterogeneity. Secondly, the pFedMe algorithm is based on parameter interaction, which requires the client to send the locally updated model parameters to the central server for aggregation in each iteration, which may lead to more data to transfer and thus reduce communication efficiency. Finally, pFedMe needs to update models synchronously, i.e., all clients need to complete their local updates at some point in time and then send them to a central server, which also reduces the efficiency of model optimization.

Table 2-2 Algorithm 2[4]

Algorithm 2: pFedMe: Personalized Federated Learning using Moreau Envelope Algorithm

Input: $T, R, S, \lambda, \eta, \beta, \omega^0$

for $t = 0$ to $T-1$ **do**

  Server sends w_t to all clients

  for all $i = 1$ to N **do**

    $w_{i,0}^t = w_t$

    for $r = 0$ to R-1**do**

      Sample a mini-batch D_i ; with size $|D|$ and minimize $\tilde{h}_i(\theta_i; w_{i,r}^t; D_i)$, defined in (7), up to an accuracy level under (8) to find a δ -approximate $\tilde{\theta}_i(w_{i,r}^t)$

      $w_{i,r+1}^t = w_{i,r}^t - \eta\lambda(w_{i,r}^t - \tilde{\theta}_i(w_{i,r}^t))$

    Server uniformly samples a subset of clients S^t with size S , and each of the sampled client sends the local model $w_{i,R}^t$, $\forall i \in S^t$, to the server

  Server updates the global model: $w_{t+1} = (1-\beta)w_t + \beta\sum_{i\in S^t}\frac{w_{i,R}^t}{S}$

In conclusion, the pFedMe algorithm still has many problems in supporting model heterogeneity and improving communication efficiency. Therefore, this dissertation implements the FedCache architecture with NLP features to cope with the complex scenario of NLP models performing distributed text categorization tasks in the form of federated learning, which provides an effective solution for NLP-based federated learning to solve practical application problems.

2.4 Knowledge Distillation

In this section, we will investigate the distinctive features of knowledge distillation, the categorization methods, and the implications of their practical applications in the real world. To illustrate this concept more concretely, this dissertation will use multi-teacher knowledge distillation as a typical case to analyze the working principles, the design of the system architecture, as well as its effectiveness, advantages, and potential challenges in a variety of application scenarios. By analyzing this area in depth, we hope to provide readers with a comprehensive and in-depth understanding to better grasp the important role and application prospects of knowledge distillation technology in modern science and technology.

2.4.1 Feature-based knowledge distillation

Deep neural networks are mainly good at learning multiple layers of feature representation by increasing abstraction, so the outputs of both the last and intermediate layers of the neural network can be used as instructional messages from the "teacher" to the "student". Feature-based knowledge distillation is an extension of logits-based knowledge distillation and is especially effective when the "student" network is deeper than the "teacher" network. The feature representation of the middle layer provides guidance information for student training through the features extracted from the teacher's implicit layer. The main idea is to directly match the implicit layers of the "teacher" and the "student". This knowledge distillation is completed in two stages, firstly the feature information extracted from the middle layer of the "teacher" network guides the "student" to train the first half of the model, and only in the second stage is the knowledge distillation carried out on the whole network.

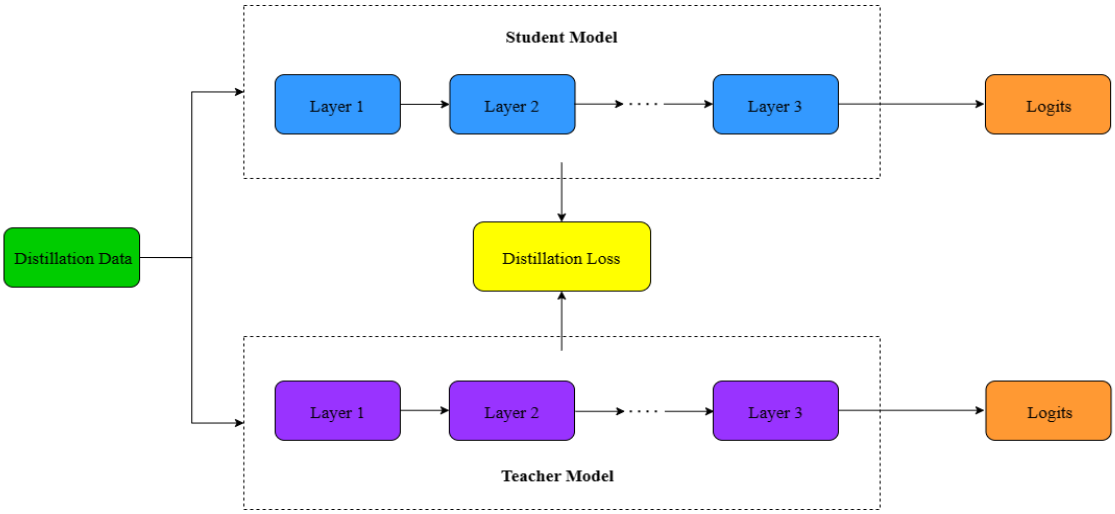


Figure 2-5 Schematic of feature-based knowledge distillation

The schematic diagram of feature-based knowledge distillation is shown in Figure 2-5, where the feature information comes from the information contained in the middle layer of the neural network, and the loss function of the feature-based knowledge distillation method is shown in Equation 5 below. Although the neural network implicit layer can provide a lot of helpful information for "student", how to select the guidance layer that provides feature information from the "teacher" model and the guidance layer used for matching in "student" remains to be investigated, and how to correctly match the feature representations of "teacher" and "student" due to the difference in the sizes of the cueing layer of "teacher" and guidance layer of "student" needs further research.

$$L_{FeaD}(f_t(x), f_s(x)) = \ell_F(\Phi_t(f_t(x)), \Phi_s(f_s(x))) \quad (5)$$

Where the detailed explanation of it can be found in the research on federated distillation[9].

2.4.2 Multi-teacher based knowledge distillation

One simple and effective distillation method makes direct use of logits to match the knowledge information output from the neural network, and another method extracts the knowledge by matching the intermediate layer features of the neural network. In order to cope with more complex scenarios, different variants of knowledge migration methods have been derived, among which are knowledge distillation based on adversarial learning, knowledge distillation based on the attention mechanism, and multi-teacher knowledge distillation, this dissertation focuses on multi-teacher knowledge distillation, and the following is an introduction to the knowledge distillation method based on multi-teacher.

Different “teachers” are able to convey different useful knowledge to “student”, and a multi-teacher network can be used to distill the “student” network individually or as a whole. Different “teachers” can convey different useful knowledge to “student”, and a multi-teacher network can be used to distill the “student” network individually or as a whole. A schematic of knowledge distillation for multi-teachers is shown in Figure 2-6. Each “teacher” can have a different network structure and can be trained from different data, and different “teacher” convey different knowledge information to the “student” model.

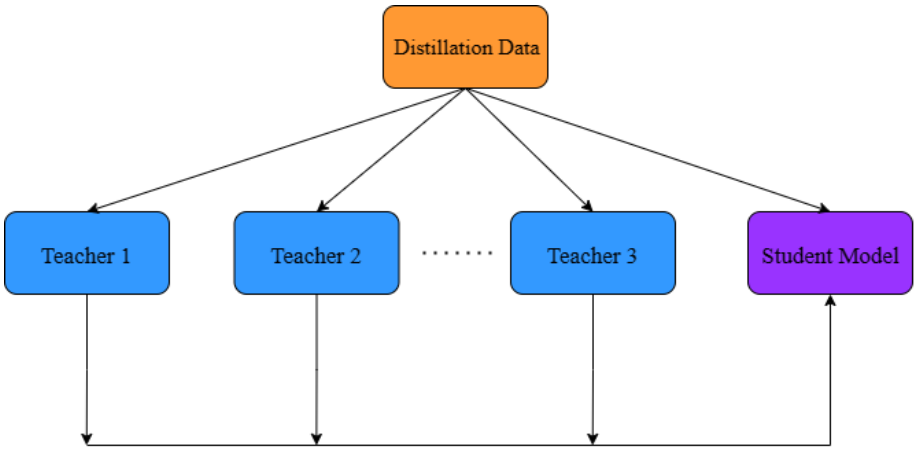


Figure 2-6 Schematic of knowledge distillation with multiple teachers

In knowledge distillation of multi-teacher networks, logits or network middle layer

features are usually used as knowledge. In order to extract knowledge from multiple “teacher” models, the simplest way is to average the outputs of all “teacher” networks and use them as guidance information for “student” distillation[30]. To migrate knowledge from multiple “teachers” to “student”, training can be done using a mechanism based on multi-step migration, that is, the “student” from the previous step is used as the “teacher” for the training in the later step. Since knowledge distillation can utilize multiple different “teachers”, many deep models trained on public datasets can also be repurposed as “teacher” models.

2.5 Summary of Chapter

This chapter provides an introduction to the background related to the fields of NLP, knowledge distillation, and federated learning. Federated learning, as a distributed machine learning framework, allows data to be distributed across mobile devices for collaborative training of machine learning models. Federated learning needs to deal with a large amount of non-IID data in real-world application scenarios, and this chapter provides an introduction to sub-independent identically distributed data and accounts for the methods used to generate non-IID experimental data. In addition, this chapter provides an introduction to recurrent neural networks and knowledge distillation, where recurrent neural networks can be applied to NLP scenarios, and knowledge distillation can be used for model compression and model fusion, which are also the techniques that form the basis of the subsequent work in this dissertation.

3 Personalized Federated Learning Driven by Knowledge Cache

This chapter will specifically introduce the theoretical research work of this dissertation, that is, to extend the original FedCache architecture in the NLP scenario, which is also one of the core contents of this dissertation.

3.1 Introduction

This chapter describes the main elements of this dissertation and the core ideas of the implemented architecture. Inspired by the traditional PFL architecture, this dissertation implements an innovative knowledge cache-driven PFL architecture. The architecture addresses the scenario of independent and homogeneous distribution of client-side data in horizontal federated learning by using knowledge caches to store and compute logits, and by using the client-side refinement of the transmitted data at the server-side to guide the personalized training and optimization of client-side models. This chapter first illustrates the characteristics and limitations of the traditional federated learning architecture and then demonstrates the new system structure, implementation method, operation process, and performance advantages of the extended FedCache.

3.2 Extension of the FedCache Architecture

The schematic diagram of the FedCache module is shown in Figure 3-1. This dissertation extends the original FedCache architecture based on the original FedCache architecture by adding the NLP module. The architecture consists of a server that contains three functional modules, namely the server-client communication module, the knowledge integration module, and the knowledge caching module; the system also has several clients, each of which contains four functional modules, namely the server-client communication module, the knowledge refinement module, the data module, and the NLP module. The NLP module contains the hash encoder and the NLP model, and the clients with an odd number of IDs and the clients with an even number of IDs have different NLP modules. In this example, the integration module is responsible for efficiently integrating knowledge into the knowledge cache, while the modeling module extracts knowledge from local data and updates the model. In addition, the schema encoder encodes specific hash data to initialize the knowledge cache. These three modules work together to build an efficient knowledge-processing system.

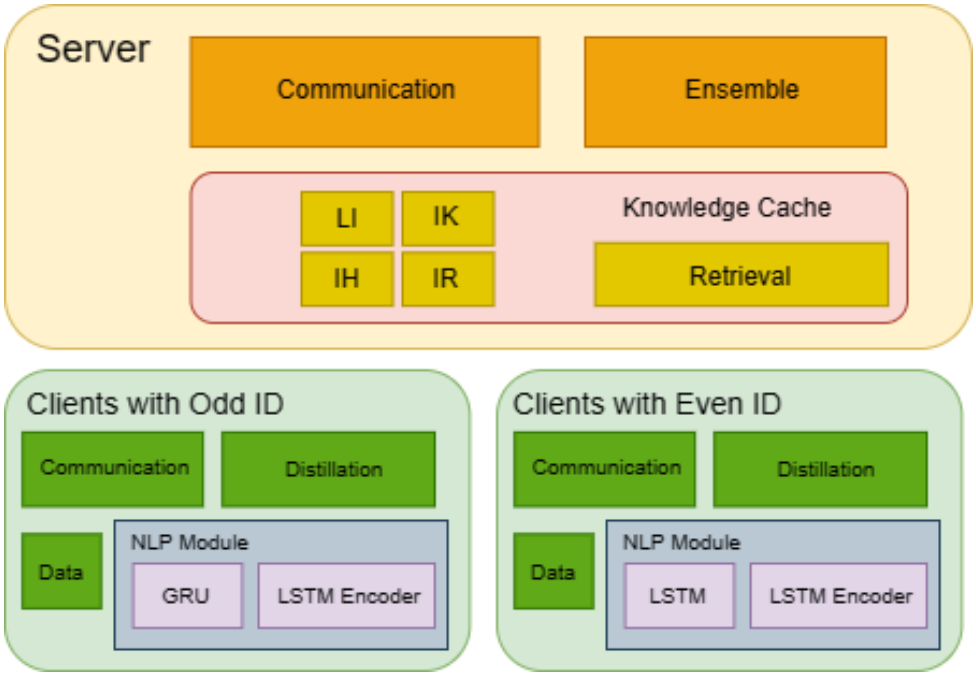


Figure 3-1 FedCache functional modules

In the initialization phase, the client generates an exact local hash value corresponding to each text sample based on the encoder and immediately loads the generated hash value into the server cache. Next, the HNSW algorithm is run on the server-side cache and retrieves the most relevant R samples for each sample by calculating the cosine similarity of the hash values. The HNSW algorithm matches each measured sample with text data that is very similar to the original sample so that the information extracted from it is favorable for the client to optimize the original sample. Meanwhile, locally accurate hashing can quickly find similar text in large-scale text data by mapping the text to the hash space and using the properties of the hash function, so that similar text is more likely to be mapped to the same bucket in the hash space, making the FedCache architecture highly efficient at matching text and allowing the HNSW algorithm to be completed in less time[24].

During each connection in the learning phase, the algorithms and indices of the privacy samples are sent to the server. The system then retrieves the best knowledge R with the highest hash similarity for each sample in the knowledge cache based on predefined similarity relationships and then performs knowledge integration and knowledge transfer to the corresponding client for local refinement. Since the refinement phase relies only on highly relevant knowledge from the data of the respective clients, the generated models are locally customizable and perform well in processing personalized tasks. In the following subsections, the main FedCache operations will be described.

3.3 Principle of Knowledge Cache

Knowledge cache is used to store and compute knowledge from each client, providing a knowledge-sharing platform for each client. Different from FedAvg and pFedMe architectures, knowledge caching provides hash-based similarity matching for each sample uploaded by the client on the server side, and transmits the knowledge related to the samples to the corresponding client, which makes its communication overhead lighter; at the same time, FedCache based on knowledge caching doesn't require that the client models are structurally the same, and only requires that their outputs have the same dimension of logits can be used, this mechanism can realize the complete heterogeneity of the model in the client; in addition, the knowledge cache also asynchronously supports the client. The following section describes the design methodology of the knowledge cache and the fundamentals of its working and explains the superiority of the knowledge cache.

3.3.1 Knowledge cache architecture implementation methods

The knowledge cache on the server side can support asynchronous access to the relevant knowledge of any local sample of controlled computational complexity, where the hash value of the corresponding sample of the relevant knowledge must be the hash value of the R closest original samples. Guided by the above design, it keeps multiple data pairs in the knowledge cache, including sample label-index pairs (LI), sample index-knowledge pairs (IK), sample index-hash pairs (IH), and sample index-relationship pairs (IR), where each of these pairs can map the first element to the second element. Based on this, there are two main phases of knowledge caching namely the initialization phase and the training phase.

The initialization phase consists of the following two main steps:

(1) Data pair initialization: hash value h_i^k is updated corresponding to each sample index (k,i) is stored in IH. Additionally, pointers are added to the LI based on their tag classes, and the knowledge corresponding to each particular pointer is initialized to zero in the IK, as shown in the following equation:

$$IH(k,i) \leftarrow h_i^k \tag{6}$$

$$LI(y_i^k) \leftarrow LI(y_i^k) \cup \{(k,i)\} \tag{7}$$

$$IK(k,i) \leftarrow \underbrace{(0,...,0)}_{Czeros}. \tag{8}$$

The fact that LI only allows pointers to instances in the same labeled class to be bound is supposed to reduce the number of candidate instances used for matching and improve the efficiency of the next computation step.

(2) Establishing relations: each sample index (k,i) , this architecture relates the R indexes $\{(l_1, j_1),(l_2, j_2),...,(l_R, j_R)\}$ of the candidate hash values that have the maximum cosine similarity to the hash value of the given sample as shown in the following equation:

$$\begin{aligned} &\arg\max_{(l_1,j_1),(l_2,j_2),...,(l_R,j_R)} \sum_{m=1}^{R} \cos(IH(k,i), IH(l_m, j_m)), \\ &s.t. \begin{cases} l_{n1} \neq l_{n2} \vee j_{n2}, \forall n_1, n_2 \wedge n_1 \neq n_2, \\ (k,i) \in LI(y^*) \wedge (l_m, j_m) \in LI(y^*), \exists y^* \\ n_1, n_2, m \in \{1,2,...,R\}, \\ y^* \in \{1,2,...,C\} \end{cases} \end{aligned} \tag{9}$$

FedCache implements R-nearest neighbor retrieval using the HNSW algorithm to compute and associate these indexes[24]. Then, the R nearest neighbor indexes associated with each sample index are retrieved and these results are stored in IR, namely:

$$IR(k,i) \leftarrow \{(l_1, j_1),(l_2, j_2),...,(l_R, j_R)\} \tag{10}$$

(3) Knowledge acquisition: Each client will traverse all its samples in this step to acquire as much knowledge as possible in the knowledge cache KC that is most relevant to that sample. For the index (k,i) uploaded, the relevant knowledge is acquired and returned in the following steps, firstly, make sure that IR has completed the relationship establishment of the sample index (k,i), then obtain other indexes related to the sample index (k,i) from IR, and finally obtain the relevant knowledge according to the result of the previous step in IK, which is expressed by the formula as follows:

$$KC(h_i^k; k, i) = IK(IR(k,i)). \tag{11}$$

Since the knowledge cache on the server side can accept access from single or

multiple clients at the same time during the knowledge acquisition process, it is only necessary that the client requesting the knowledge is online, and that the client is able to optimize the acquired knowledge asynchronously. This approach improves the efficiency of model training for federated learning and saves a significant amount of synchronous training overhead.

(4) Knowledge update: After the next round of client model training, the knowledge $IK(k,i)$ in IK will be updated by the knowledge z_i^k of the corresponding sample index (k,i) so that the clients accessing IK can get the latest knowledge, namely:

$$IK(k,i) \leftarrow z_i^k \tag{12}$$

In summary, knowledge caching accomplishes the server side of the federated learning knowledge distillation process through the above stages.

3.3.2 Knowledge distillation driven by knowledge cache

The federated learning architecture implemented in this dissertation has significant advantages in improving communication efficiency and supporting client model heterogeneity, due to the fact that client model optimization is accomplished by personalized federated distillation driven by knowledge caching. After the knowledge cache completes knowledge acquisition in the training phase, it transmits the acquired relevant knowledge to the corresponding client for knowledge distillation for its local model. In this distillation process, logits from the server side will be treated as knowledge output from the teacher model and passed to the client's local model as a reference. The local model of the client as the student model will be optimized by reducing the loss rate by using the knowledge of the teacher model as a reference.

Specifically, the FedCache architecture acquires sample knowledge from the knowledge cache that is similar to each client's private data, followed by each client distilling its acquired knowledge to complete the optimization of its local model. First, the client will generate the hash value of the local samples based on the encoder $f^h(.)$, and this dissertation selects the LSTM pre-trained model to encode the hash value of the local samples, and uploads the generated hash value with the index and label of the corresponding samples to the IH module of the knowledge cache, namely:

$$h_i^k = f^h(X_i^k) \tag{13}$$

Where, after initializing the knowledge cache, the client will train on each local sample $z_i^k = f^k(X_i^k)$. In the training process, the client k first extracts the knowledge z_i^k from the X_i^k, and then uploads the knowledge z_i^k with the corresponding sample index (k,i) to the server, namely:

$$z_i^k = f^k(X_i^k) \tag{14}$$

In the above process, the sample hash encoder is obtained from the pre-trained model by removing the fully connected layer to be able to obtain vectors that represent the features of the samples in the client. Since the sample data processed is text, this dissertation chooses the LSTM pre-trained model, and the feature vector, namely, the hash value, of the sample data is obtained by this encoder. Meanwhile, the server, for each category among all the samples, can get the similarity between all the data of that category by calculating the distance or cosine value between the feature vectors of the samples, so the server can select the nearest R samples to represent the current sample and obtain the knowledge of these R samples from the knowledge cache KC, namely:

$$(zr_i^k)_1, (zr_i^k)_2, ..., (zr_i^k)_R = KC(h_i^k; k, i) \tag{15}$$

Where, $(zr_i^k)_S$ stands for the s-th sample knowledge acquired for a given indexed sample, and then the acquired knowledge is averaged in order to perform the calculation with the following formula:

$$\overline{zr}_i^k = \frac{1}{R}\sum_{s=1}^{R}(zr_i^k)_S \tag{16}$$

Where, the calculated average cognitive value calculated for customer k is then assigned to optimize the teardrop model, which is defined as follows:

$$\begin{aligned} &\arg\min_{W^k} J^k(W^k) \\ &= \arg\min_{W^k} \sum_{(X_i^k, y_i^k)\in D^k} [L_{CE}(\tau(f^k(X_i^k)), y_i^k) + \beta.KL(\tau(f^k(X_i^k)) \,\|\, \tau(\overline{zr}_i^k))] \end{aligned} \tag{17}$$

Where the definition of calculating the average cognitive value for customers is shown above.

In summary, this dissertation constructs an encoder based on the LSTM pre-trained model and deploys a different RNN model on each client, associates the sample knowledge of all clients by the hash value generated by the encoder on a sample-by-sample basis, and finally, each client obtains the relevant knowledge of the local samples and performs knowledge distillation locally to complete the model optimization. Based on knowledge distillation, FedCache architecture avoids the large communication cost due to model parameter interaction and the heterogeneity limitation to ensure the successful transmission of model parameters and has a clear advantage over traditional federated learning architectures in terms of communication overload and handling model heterogeneity.

3.4 Principles of Cache-Driven PFL Architecture

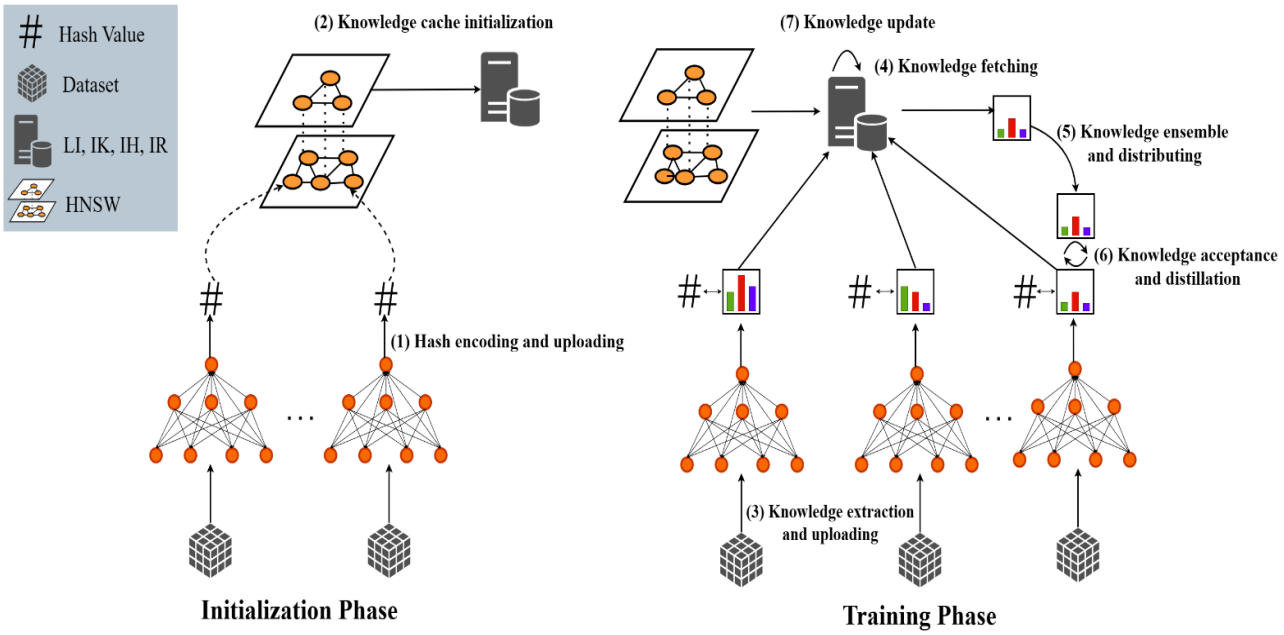


Figure 3-2 Implementation process overview of FedCache

The execution of the FedCache algorithm is shown in Figure 3-2, and the execution of FedCache on the client and server is shown in Algorithm 3[24] and Algorithm 4[24], respectively. Overall, FedCache allows NLP local models on the device to extract integrated knowledge on samples similar to its data with the help of a server-side knowledge cache and use that integrated knowledge as a teacher model for the distillation optimization of client models.

In particular, FedCache consists of the steps below:

(1) Hash encoding and uploading: For each sample from a given client, a hash value is needed to represent the features of that client sample, and each hash value is

computed by the pre-trained model LSTM that has been set up globally according to Eq. (14). Then, this hash value will be uploaded to the server along with the corresponding label and sample index, as shown in line 4 of Algorithm 3. Since the encoder is a pre-trained model with a large number of superimposed nonlinear mappings, the dimension of its output encoding is much smaller than that of the original sample data, and thus the confidentiality of the process of sharing the hash value with the server side is very high.

(2) Knowledge cache initialization: The server receives information such as hash values and sample indexes generated by the previous step from each client and establish a relationship between the sample indexes in the knowledge cache so that each sample can find its closest R samples based on the pre-established mapping indexes.

(3) Knowledge extraction and uploading: The client first performs local model training, uploads information such as model output logits and their corresponding sample indexes to the server, and updates the IK module in the knowledge cache, a step based on sample granularity instead of the traditional communication based on parameter interaction. As the size of the algorithms and sample lists is several times smaller than the size of the model parameters or attributes, and the size of the data does not increase linearly with the number of clients, the communication load can be significantly reduced.

(4) Knowledge acquisition: After receiving the sample index uploaded by the client, the the server will find and obtain the R-nearest-neighbor-related sample knowledge in the knowledge cache based on the sample index.

(5) Knowledge distribution: After obtaining the R-nearest neighbor sample knowledge in the knowledge cache, the server will average this knowledge based on Eq. (16) and distribute the processed knowledge to the corresponding clients for knowledge distillation.

(6) Knowledge reception and distillation: The client will receive the corresponding sample integration knowledge distributed from the server and optimize the client's local model according to Eq. (17), it improves the communication efficiency.

(7) Knowledge updating: The client's local model optimization will be followed by the next round of model training and knowledge uploading, and according to Eq. (12), the newly uploaded knowledge will update the old knowledge that was originally stored in the knowledge cache to ensure that the next round of knowledge distillation uses the knowledge acquired in this round.

The following Algorithm 3 and Algorithm 4 are the pseudo-codes involved in the above process. Specifically, lines 3 and 4 of the code in Algorithm 3 implement the encoding work and the uploading work in Step 1; lines 3 and 4-6 of the code in Algorithm 4 implement the data receiving work and the associating samples work in Step 2, respectively; lines 9 and 10 of the code in Algorithm 3 implement the client's work of extracting the logits and uploading the information, such as the indexes of the corresponding samples, in Step 3; lines 10 and 11 of the code in Algorithm 4 together implement the knowledge acquisition work of Step 3, according to Eq. (16). server-side work; lines 10 and 11 of code in Algorithm 3 together implement the knowledge acquisition in Step 3; in Step 4, line 12 of code in Algorithm 4 implements the knowledge integration according to Eq. (16), and line 13 of code implements the distribution of the integrated knowledge. In step 5, lines 11 and 12-13 of the code of Algorithm 3 implement the integrated knowledge distribution and the client-side local model knowledge distillation, respectively; in step 6, lines 10 and 14 of the code of Algorithm 4 jointly implement the updating of the knowledge in the server according to Eq. (12).

Table 3-2 Algorithm 3[24]

Algorithm 3: FedCache on Client k.

1: // Initialized phase
2: **for** $(X_i^k, y_i^k) \in D^k$ **do**
3: $h_i^k \leftarrow f^h(X_i^k)$
4: Upload h_i^k with index (k,i) and label y_i^k to the server
5: **end**
6: // Training phase
7: **repeat**
8: **for** $(X_i^k, y_i^k) \in D^k$ **do**
9: $z_i^k \leftarrow f^k(X_i^k)$
10: Upload z_i^k with index (k,i) to the server
11: Download averaged ensemble knowledge $\overline{zr}_i^k$ from the server
12: $w^k \leftarrow w^k - lr.\nabla_{w^k} J^k(w^k)$
13: **end**
14: **until** Training stop;

Table 3-3 Algorithm 4[24]

Algorithm 4: FedCache on the Server.

repeat
  Receive h_i^k with index (k,i) and label y_i^k from client k
  Update LI, IK, IH with according to Eq. (6) (7) (8)
until Receive all indexes (k,i) from K clients
Bulid realtions via HNSW according to Eq. (10) and Eq. (12)
repeat
  for (k,i) **do**
    Receive h_i^k with index (k,i) and label y_i^k from client k
    Fetch R related knowledge from the knowledge cache according to Eq. (15), Eq. (11)
    Obtain ensembled fetched knowledge $\overline{zr}_i^k$ according to Eq. (16)
    Send $\overline{zr}_i^k$ to client k
    Update knowledge cache according to Eq. (12)
  end
until Training stop;

3.5 Summary of Chapter

This dissertation begins with an introduction to some of the current improved approaches to federated learning and then leads to the focus of this dissertation: a cache-driven framework for communication-efficient PFL.

Firstly, this dissertation introduces the work on extending the FedCache architecture, specifically, it presents the idea of updating the client model based on parameter interactions to the way of updating the client model based on knowledge caching. The client generates logits by training the local model and uploads the relevant information to the server-side knowledge cache, the additional communication and computational overhead associated with logits and their relevant information is almost negligible compared to parameter interactions; moreover, the knowledge distillation implemented in this chapter does not require exchanging parameters between homogeneous models, thus allowing federated learning to be carried out on the premise that the client-side models are completely heterogeneous.

The client generates logits by training the local model and uploads the relevant information to the server-side knowledge cache, the additional communication and computational overhead associated with logits and their relevant information is almost

negligible compared to parameter interactions; moreover, the knowledge distillation implemented in this chapter does not require exchanging parameters between homogeneous models, thus allowing federated learning to be carried out on the premise that the client-side models are completely heterogeneous.

Then, this chapter explains the knowledge cache-based federated distillation process in detail, including the initialization phase and the training phase, and describes the specific steps and computational methods executed by the client and the server in the federated distillation process. Different from the application of FedCache in other scenarios, FedCache in the NLP scenario adopts an LSTM pre-trained model as encoder and different RNN models as local models for different clients, aiming to simulate the heterogeneity of the client models during the actual application in the NLP scenario.

Finally, this chapter provides a brief description of the basic flow and algorithmic pseudo-code for the operation of the FedCache architecture based on the previous two points, and the advantages of the framework are introduced and explained accordingly.

4 Experimental Design and Performance Analysis

The chapter begins with an introduction, followed by an introduction to the experimental environment settings, which describes the comparison methods and principles of the experiments. The PFL architecture implemented in this dissertation is generalized to all data-sets of NLP scenarios, but to carry out the experiments, sample data-sets are selected in this chapter to compare and verify the innovations of this dissertation.

4.1 Introduction

This chapter experiments with the cache-driven FedCache architecture for personalized learning based on cache-driven federated learning in the context of an NLP application scenario and compares it with the traditional parameter interaction-based federated learning architecture. Firstly, the preparation process of the non-IID data-set in this experiment is introduced, and secondly, the effectiveness of the FedCache framework for solving the federated learning communication overhead problem is verified through experiments, while the NLP-based FedCache framework is implemented. Finally, performance comparison experiments are conducted between the FedCache framework the traditional FedAvg architecture and the pFedMe architecture to demonstrate the superiority of the FedCache architecture implemented in this dissertation in NLP application scenarios through comparison and analysis.

4.2 Basic Settings of Experiment

In this section, the basic parameter settings of the experiment are specified. Firstly, the basic settings of the experiment are introduced, including the basic configuration of the development platform, the introduction of the content of the data-set, the construction method of the non-IID data-set, and the parameter settings of the experimental model. Then, this section will specify the reasons for choosing the above methods and configurations, explain the effects of different parameter configurations on the experimental results, and prove the validity of the experimental results. Finally, this section will present tables with specific configuration values that specify the parameter configurations chosen for the experiment.

4.2.1 Experimental platforms and tools

This dissertation uses Python language and Pytorch framework to implement GRU and LSTM models and build FedCache architecture to accelerate the model training with GPU. Use libraries such as torch text and spacy to load data-sets and preprocess text. The experiment uses the matplotlib library to visualize the training process and training results in order to better understand the learning situation and performance of the model. At the same time, this experiment does rapid iterative development based on PyCharm IDE to improve development efficiency and code quality. Table 4-1 lists the specific hardware configurations and software versions of the experimental platform. These configurations and software versions provide the necessary support and guarantee for the experiment. Through the organic combination of these tools and the environment, it can perform experiments more efficiently and obtain reliable experimental results.

Table 4-1 shows that the development environment adopted for the experiment is Python 3.7.5, the deep learning framework adopted is Pytorch, the dataset loading tool adopted is Torchtext, the tool for data preprocessing is Spacy, the tool for graph drawing and data visualization is TensorBoard, and the development platform is PyCharm 2023.3. In addition to this, the table contains specific hardware configurations.

Table 4-1 Statistics for the IMDB and SST data-set

Experimental tools	Software version / Hareware configuration
Language environment	Python 3.7.5
Deep-learning architecture	PyTorch 1.3.1
Data-set loading	Torchtext 0.4.0
Data-set re-processing	Spacy 2.2.3
Data visualization	TensorBoard 2.0.1
Development environment	PyCharm 2023.3
Hardware	CPU: Intel(R) Core (TM) i7-8700K CPU @ 3.70GHz; RAM: 48GB; GPU: Geforce RTX 3050 Super 8G, Geforce GTX 1080 8G; Storage: 51 2GB SSD, 2TB HDD

4.2.2 Introduction of data-set

The scenario validated in this dissertation is text categorization based on NLP techniques, specifically, the client-side local model after FedCache federated learning needs to predict the corresponding sentiment labels for a given English text. Therefore, two commonly used open-source English data-sets, the IMDB Movie Review Dataset and The Stanford Sentiment Treebank (SST-2), are used as experimental data in this chapter. The above two data-sets will be described specifically in the following respectively.

(1) IMDB data-set

The Internet Movie Reviews dataset is the official dataset of the Internet Movie Database (IMDB), containing 50,000 movie reviews that can be used in English text classification searches. According to the official description of IMDB, the data-set contains 25,000 training reviews and 25,000 test reviews, each of which is categorized as either "positive (pos)" or "negative (neg)", which indicates its sentiment tendency. Comments with a tendency value between 0.0 and 0.2 were categorized as negative, and comments with a tendency value between 0.9 and 1.0 were categorized as positive.

The IMDB dataset is useful for machine learning research because it provides training and testing corpus and sentiment labels that can be used for developing and evaluating a variety of NLP applications such as sentiment analysis, text categorization, and text generation, etc., and the details of the data distribution of this dataset are shown in Table 4-2. The IMDB contains several attributes, and by setting different attributes as the learning objectives, it can carry out different learning tasks, Table 4-3 shows the basic information of the IMDB data-set. This dissertation selects the sentiment tendency attributes of IMDB, divides the data-set into heterogeneous clients, and trains federated learning text categorization on the local models of the clients, in order to validate the superiority of the FedCache architecture compared with the traditional federated learning architecture and to obtain high-precision text sentiment categorization models.

Table 4-2 Data distribution of the IMDB data-set

Data-set	Label	Number
Training set	pos	25000
	neg	25000
Testing set	pos	25000
	neg	25000

Table 4-3 IMDB data-set details

Table Name	Data Size	Data
aka_name	0.90M	id, person_ id, name, imdb_ index, name_ pcode_ cf, name_ pcode_ nf, md5_sum
cast_info	36.00M	id, person_ id, movie_ id, person_ role_ id, note, nr_ order, role_ id
movie_companies	2.60M	id, movie_ id, company_ id, company_ type_ id, note
movie_info	7.30M	id, movie_ id, info_ type_ id, info, note
movie_info_idx	1.40M	id, movie_ id, info_ type_ id, info, note
movie_keyword	4.50M	id, movie_ id, keyword_id
person_info	3.00M	id, person_ _id, info_type_ id, info, note

(2) SST-2 data-set

SST (Stanford Sentiment Treebank) data-set is a data-set for sentiment classification, released by Stanford University in 2013. The data-set uses movie reviews as the original data samples for the task of sentiment classification of text sequences. The SST data-set uses two kinds of labels labeling sentiment for the sample data, positive evaluation and negative evaluation, and unlike the IMDB data-set, the SST data-set has less data, so this dissertation selects less experimental data from the SST data-set. In this dissertation, 6000 movie review data with labeled sentiment categories are selected as training data from the SST data-set, of which 3000 are positive review samples and 3000 are negative review samples; in addition, 2000 movie review data are selected as test data, of which 1000 are positive category review

samples and 1000 are negative category review samples.

Table 4-4 demonstrates the statistical information of the above data-sets, and both the model proposed in this dissertation and the comparison model are evaluated experimentally on the IMDB data-set and the SST data-set.

Table 4-4 Statistics for IMDB and SST data-sets

Data-set	Training set		Testing set	
	pos	neg	pos	neg
IMDB	12500	12500	12500	12500
SST-2	3000	3000	1500	1500

4.2.3 Construction of non-IID dataset

In this chapter, the authoritative datasets of NLP, IMDB dataset, and SST-2 data-set are selected for experiments. The main research content of this dissertation is to apply FedCache architecture in the practical scenarios of NLP. In the practical application scenarios, there are usually multiple clients involved in federated learning, and there are obvious differences in the model structure and local data between different clients, so it is necessary to divide the data-sets are divided to construct the data non-IID scenarios in federated learning.

To address the problem of client-side non-IID data, this dissertation takes the construction of non-IID data-set from the original balanced data-set by utilizing Dirichlet distribution sampling as follows:

Assuming that p is the prior distribution of all categories of N, the training data of each client follows the distribution of the N categories q, and the sum of the probability distributions in each client is 1, using the Dirichlet distribution with the parameter alpha to be able to sample from the p distribution to get the distribution of each client's data, that is, $q \sim Dir(\alpha, p)$. The parameter α can control the degree of imbalance between client data, and the closer the parameter α is to 0, the greater the degree of client data non-IID. This chapter adopts the above method to divide the original data-set using Dirichlet distribution to restore the non-IID distribution of client data labeling imbalance in practical application scenarios.

This dissertation takes the IMDB dataset as an example to specify the process of data-set preparation before the experiment. First of all, the parameter setting of data-set division is

carried out, the IMDB data-set has only two kinds of emotional tendencies, namely positive and negative, so its number of labeling categories is set to 2. At the same time, the number of clients participating in federated learning is set to 20, and the parameter of Dirichlet distribution is set to 0.5. Then the IMDB data-set is divided into the training set and test set according to its characteristics, the "data" part of the IMDB data-set is the raw data of the training set and test set, so it is divided into X_train and X_test, which represent the raw data used for training and testing. Similarly, the target part of the IMDB data-set is divided into Y_train and Y_test, representing the sample labels used for training and testing. After sampling, the number of local data and the distribution of labels obtained by each client are shown in Figure 4-4:

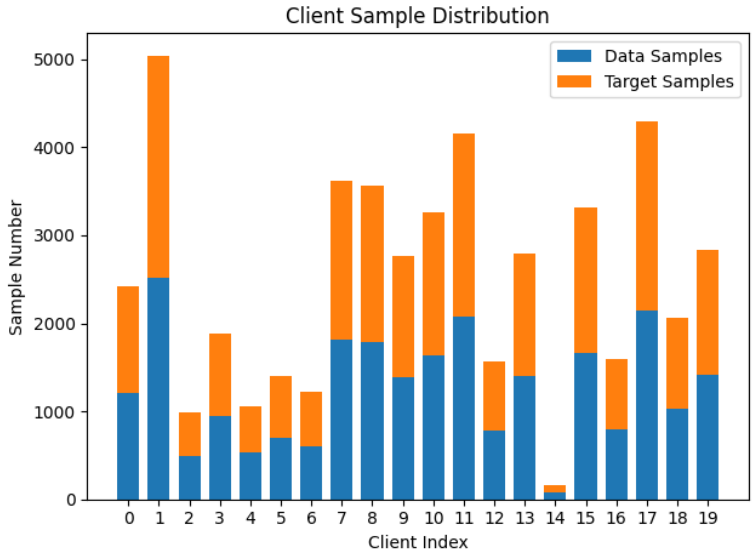


Figure 4-4 Results of Dirichlet distribution constructing client non-IID sample data

After processing the above-unbalanced data distribution and accurate labeling, this experiment successfully built an experimental environment for non-IID data with the same distribution characteristics on each client. This process involved technical means such as resampling and weighting of data to ensure the quality and consistency of data. On this basis, it further constructs a natural language processing (NLP) network model suitable for this experiment, including selecting a suitable network architecture and optimizing algorithms and hyperparameters. This process lays a foundation for the subsequent model training and evaluation and helps to improve the generalization ability and accuracy of the model.

4.2.4 Model network settings

The IMDB data-set and SST-2 data-set used in this chapter are both English text data-sets, so the experimental model uses Recurrent Neural Networks (RNN) that can extract text features better. Because individual client models in practical application scenarios often differ due to the limitations of device arithmetic, this dissertation selects two variants of RNN, namely, Long Short-Term Memory Recurrent Neural Network (LSTM) and Gated Recurrent Unit Networks (GRUs), as client-side local models for training. For the encoder in the initialization phase of the knowledge cache, this dissertation uses an LSTM pre-trained model and removes the fully-connected layer of this model. Meanwhile, this dissertation selects the clients with even serial numbers to use the LSTM model, and the clients with odd serial numbers to use the GRU model, at the same time, the number of local samples of each client has a large difference, so it can roughly simulate the heterogeneity of the clients in the actual application scenarios.

In addition, the IMDB and SST-2 data-sets are similar and have the same text length and label meanings, the difference lies in the amount of data and the source is different, so whether it is the LSTM model or the GRU model, the same model setup is used for the two data-sets, and the detailed model structure and parameters are shown in Table 4-5.

Table 4-5 LSTM model structure and parameter settings

layer Name	Input Feature	Hidden Units	Sequence Length	Return Sequence
LSTM-1	128	256	changeable	TRUE
LSTM-2	256	128	changeable	TRUE
Linear	-	-	10	FALSE

The experiments of this chapter train the proposed network model. Specifically, it sets the dataset batch size to 8 to improve training efficiency. At the same time, it divided the local data set of each client into a training set and a test set with a ratio of 1:1 to ensure that the model has a good generalization ability during training and evaluation. In terms of the optimization algorithm, it used SGD with a learning rate of 0.001, which helps the model converge gradually during training. In terms of loss function, this experiment chooses the crossing loss function. Through these settings, it is easier to get better model performance.

4.3 Experimental Assessment Indicators and Comparison Methods

This section will focus on the specific indicators for evaluating the experiments and demonstrate the rigor of the experimental results by specifying the comparison method and comparison process of the comparison experiments.

4.3.1 Indicators for experimental assessment

The evaluation metrics of this experiment can be specifically categorized into model performance evaluation metrics and federated learning architecture evaluation metrics.

First of all, the model performance index, this dissertation aims to use two RNN models to perform the text classification task, so whether the corresponding sentiment labels can be accurately output for a given English text is the most important index to consider the performance of the model, and this dissertation selects the accuracy rate (acc) as the evaluation index of the model. Specifically, the accuracy rate is the proportion of correctly labeled samples to the total number of samples in the data-set that participate in the model prediction. The formula for calculating the accuracy rate is as follows:

$$acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{18}$$

The TP is the positive sample size that the model predicts correctly, TN is the negative sample size that the model predicts correctly, FP is the positive sample size that the model predicts incorrectly, and FN is the negative sample size that the model predicts correctly.

Then, the evaluation metrics of the architecture performance, the main purpose of this dissertation is to compare the FedCache architecture with the traditional FedAvg and pFedMe architectures to highlight the superiority of the FedCache architecture in terms of communication efficiency and model heterogeneity support. This dissertation also measures the performance of different federated learning architectures in terms of the overhead of communication performed by federated learning systems and the maximum client average model accuracy of each system. Besides, the average client model accuracy is obtained in a reasonable training time, and the experiment ends when the algorithm converges or the communication overhead reaches the limit. In terms of model heterogeneity support, clients with better performance can get more information from other clients for training, and obtain models with better generalization ability, thus improving the performance of client models.

4.3.2 Implementation and demonstration of comparison methods

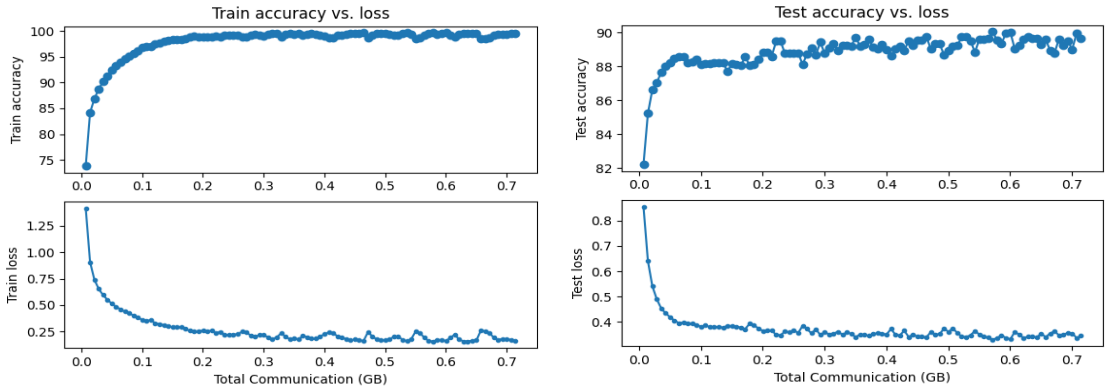
The experiments in this chapter select the traditional federated learning frameworks FedAvg and pFedMe as the comparison methods to highlight the performance advantages of the FedCache framework. The FedCache framework is able to bypass the parameter interaction behaviors during the model training process through knowledge distillation, and each client can obtain a personalized model that is most suitable for its local data distribution, which effectively mitigates the performance degradation caused by the clients' non-IID data distribution caused by the degree of performance degradation of federated learning training models. At the same time, only logits and indexes are transferred between the server and clients of FedCache, so the communication overhead does not increase with the growth of the number of clients, which makes the communication overhead lighter compared to the other two federated learning architectures. Next, using acc as the model evaluation metric, based on the experimental base setup in Section 4.2, it experimentally observes the performance of FedCache architecture versus traditional federated learning architecture in the non-IID data scenario setup.

For the traditional federated learning architecture, the client selection ratio in the FedAvg and pFedMe architectures is set to be 1 on the server side, namely, all 20 clients participate in federated training throughout. The algorithm stopping condition is that the current number of communication rounds is greater than 300 rounds, and the average value of the model parameter update of each client in the current round in the current group is small enough. In this experiment when the average value of the change of the model w parameters on each client is less than 0.4, it is considered that the FedAvg as well as pFedMe algorithms converge and their federated learning process ends.

For the FedCache framework, set the ratio of clients selected by the server therein to 1, namely, all 20 clients participate in the federation training throughout. The algorithm stopping condition is that the current number of communication rounds is greater than 200 rounds, and the average value of the model parameter update of each client in the current group in the current round is small enough. In this experiment when the average value of the change of the parameters of the model w on each client is less than 0.3, it is considered that the FedCache algorithm converges and its federated learning process ends.

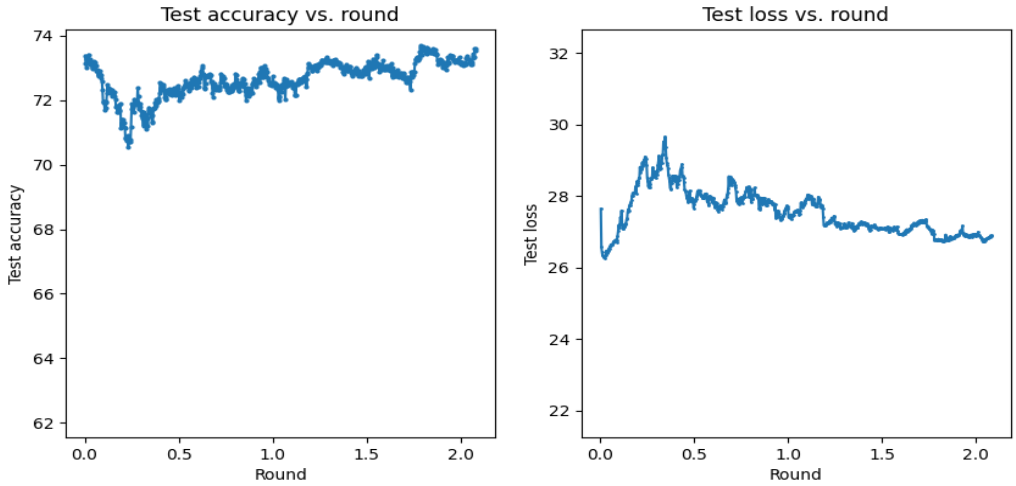
The average of the test accuracy and the average of the training accuracy of the model generated by federated learning on the local data-set of each client is used as the accuracy (acc) of the model, and the total communication overhead is defined to be the magnitude of

the volume of the data transmitted in all the communication rounds, where the communication of each communication is defined to be the sum of the volume of bytes occupied by the data such as logits and indexes transmitted between the server and the client in GB. round is the sum of the bytes occupied by the data such as logits and indexes transmitted between the server and the client, in GB. Uniformly set the number of communication rounds for the IMDB data-set and SST-2 data-set to 200 rounds on the FedCache framework and 300 rounds on the FedAvg and pFedMe frameworks, and observe the changes in the model test accuracy and loss rate, as well as the important federated learning process parameters that are directly related to them with the number of communication rounds, in the case of in the FedCache architecture for example, the changes of the above parameters are schematically shown in Figure 4-5 and Figure 4-6, respectively.



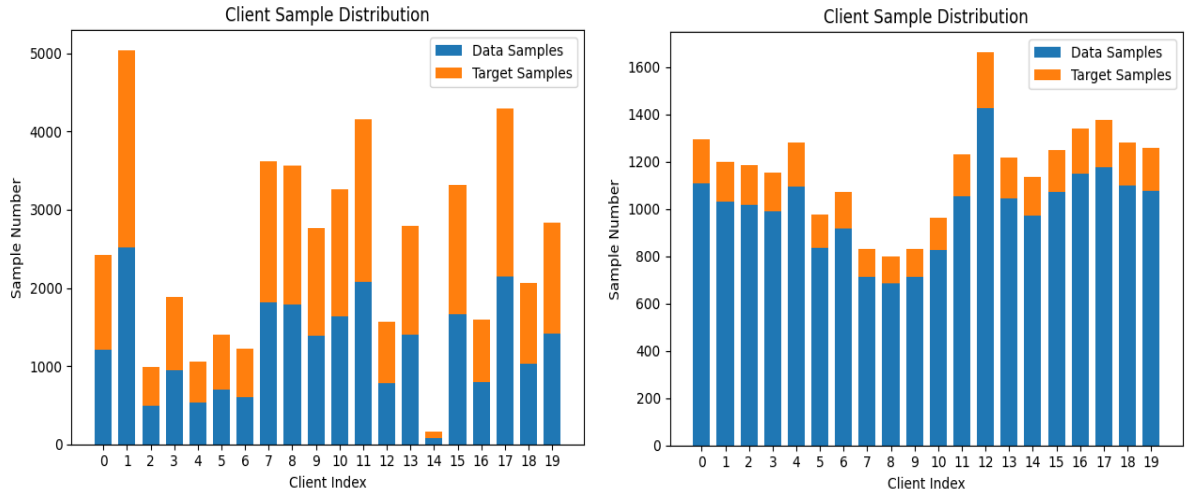
(a) Train accuracy - communication overheads (b) Test accuracy - communication overheads

Figure 4-5 Federated learning of FedCache on the SST-2 data-set



(a) Train accuracy - communication overheads (b) Test accuracy - communication overheads

Figure 4-6 Federated learning of FedCache on the IMDB data-set



(a) Distribution of IMDB data-set (b) Distribution of SST-2 data-set

Figure 4-7 Result of IMDB and SST-2 data-set partitioning

With Figure 4-5, Figure 4-6, and Figure 4-7, the experimental process of the FedCache framework is specifically described as follows:

In the first step, the FedCache architecture randomly divides the data-set so that the data presents non-IID data-sets on the client and the distribution of the data-set after division by this architecture is shown in Figure 4-7. Since the sample size of the SST-2 data-set is smaller compared to the IMDB data-set, the model converges faster in the case of data heterogeneity, and the obtained model test accuracy is higher. In the second step, the client encodes the local data based on the encoder, obtains the hash value, and uploads the hash value with its corresponding indexes and tags to the server for knowledge cache initialization. This step involves the transmission of data such as hash values and occupies the main part of the communication overhead of the FedCache architecture. In the third step, the client conducts model training locally, outputs the training accuracy of the model, and uploads the output logits with their corresponding indexes and other data to the server, which acquires the relevant logits and transmits them to each client. Finally, each client performs knowledge distillation based on the acquired knowledge, optimizes the local model, and outputs the testing accuracy of the model.

As can be seen from Figure 4-6, when using the IMDB data-set for federated learning, the communication overhead of FedCache is roughly 4KB per round, together with the communication overhead of about 0.41GB generated by transmitting the hash value during the initialization process, the total communication cost is no more than 2GB overall after

completing 200 rounds of training and testing, and the model's testing accuracy reaches 71% accuracy rate. Compared to the IMDB data-set, SST-2 also has relatively less communication overhead due to less sample data, the amount of data generated by the initialization of the knowledge cache is only 0.26GB, the communication overhead per round is about 2.4KB, and it achieves higher model testing accuracy.

4.4 Comparative Experiments and Analysis of Results

In this dissertation, subsection 4.3.2 experimentally shows comparative methods and metrics for measuring the performance of federated learning architectures and gives some examples of the FedCache architecture. Next, this section completes the interpretation of the experimental results of the other three traditional federated learning architectures based on the same experimental setup as in the previous subsection and compares the FedCache architecture with the FedAvg, pFedMe, and MTFL architectures in terms of the total number of communicated rounds, the total amount of communication, and the model accuracy, demonstrating the FedCache architecture's NLP Federated Learning application scenario's superiority.

4.4.1 Experimental parameter settings

During the experiment, four frameworks, FedCache, FedAvg, pFedMe, and MTFL, need to be tested. Each federated learning architecture needs to go through the four stages of IMDB data-set and SST-2 data-set partitioning, model training and testing, server-side and client-side communication, and client-side model optimization. In this dissertation, the experimental settings are divided into three parts: the basic parameter settings of the architecture, the parameter settings of the data-set distillation algorithm, and the specific parameter settings of different architectures, respectively.

First, for the basic parameter settings of the architectures, all architectures set the number of clients to 20, the learning rate η is set to 0.001, and all architectures conduct experiments on the data-sets IMDB and SST-2 respectively. Non-IID data were constructed following the data preparation process described in subsection 4.2.3, namely, the IMDB and SST-2 data-sets were randomly partitioned to each client using a Dirichlet distribution with the parameter $\alpha = 5$, and the recurrent neural network in subsection 4.2.4 was chosen as the model used for all architectures. The size of the small batch of data sampled from the data-set is set to 8, and

the representative labeling category is set to 2.

Secondly, for the experimental setting of the knowledge distillation algorithm in the FedCache architecture, the initialization model used for each client distillation is one of the two variants of the RNN model, and the hyperparameters for the distillation training are set as follows: the number of distillation data is set to get one distillation data for each class of labels, and the temperature of the knowledge distillation is set to 1.0.

Finally, among the specific parameter settings for the different architectures, the number of communication rounds is set to 200 for FedCache, while the number of communication rounds is set to 300 for all other architectures. In this dissertation, the number of samples R for R-nearest neighbor sample matching in FedCache is set to 16, namely, each sample will be matched to the knowledge of its most relevant 16 samples during the stage of communication between the server and the client. The following shows the experimental results comparing the FedCache architecture with the FedAvg, pFedMe, and MTFL architectures.

4.4.2 Performance comparison of federated learning architectures

This subsection describes the experimental results comparing FedCache architecture with other traditional federated learning architectures and quantitatively analyzes the superiority of FedCache in NLP application scenarios from the data level by combining experimental charts and graphs. FedCache architecture is far superior to traditional architectures in terms of communication efficiency and client heterogeneity, and it also has a certain advantage in terms of model training efficiency. Therefore, this subsection is divided into two main modules to illustrate the experimental results.

Firstly, there is the advantage of FedCache in terms of client heterogeneity supportability. Tables 4-6 and Table 4-7 show a comparison of the maximum average client model accuracy of FedCache and communication overhead over different data-sets and with different benchmark algorithms, and the maximum average client model accuracy per unit communication overhead is shown in Figure 4-8, Figure 4-9 and Figure 4-10. As can be seen from Table 4-7, the FedCache architecture achieves the maximum average client model accuracy of 71% and 89% on the IMDB data-set and the SST-2 data-set, respectively, the FedAvg architecture achieves the maximum average client model accuracy of 53% and 69% on these two data-sets, the pFedMe architecture performs at 54% and 66.7%, and the MTFL performs 54% and 68%, respectively.

Table 4-6 Server-side model and client-side model

Model	Notation
LSTM	M_1^{C}
GRU	M_2^{C}
ResNet	M_1^{S}

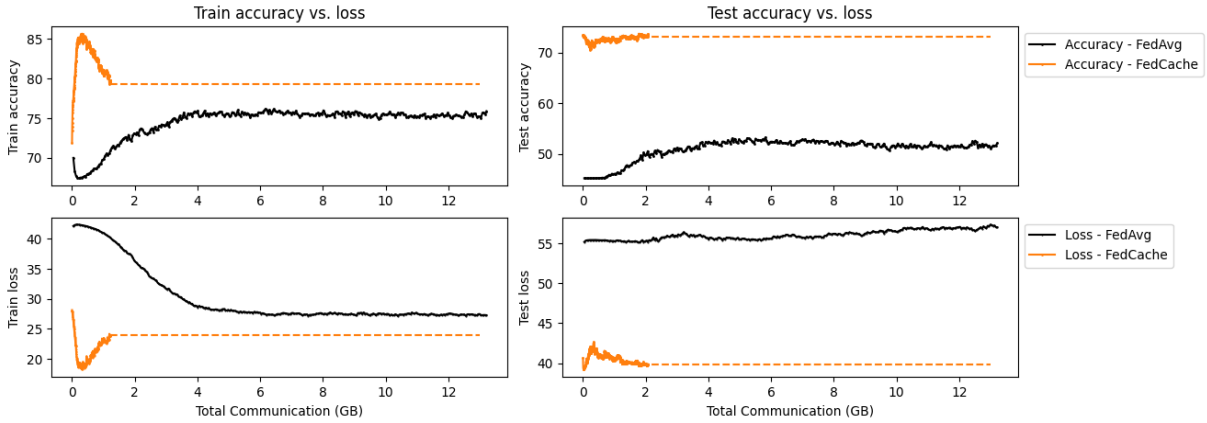
Table 4-7 Average accuracy and communication overhead of models

Dataset	Method	Model		Metric		
		Client	Server	Accuracy	Comm. (G)	Comm. (Round)
IMDB	pFedMe	M_1^{C} M_2^{C}	M_1^{S}	53.82%	6.82	300
	FedAvg			52.97%	6.34	300
	MTFL			54.43%	6.37	300
	FedCache		-	71.24%	0.82	200
SST-2	pFedMe	M_1^{C} M_2^{C}	M_1^{S}	66.73%	1.08	300
	FedAvg			69.07%	1.18	300
	MTFL			67.96%	1.32	300
	FedCache		-	89.21%	0.06	200

Overall, the FedCache architecture substantially outperforms the FedAvg, pFedMe, and MTFL architectures in terms of model accuracy regardless of whether the IMDB data-set or the SST-2 data-set is used when both the model structure and the hyperparameters affecting the model accuracy are the same, and thus the FedCache architecture provides better support for the federated learning task in the client-side heterogeneous case.

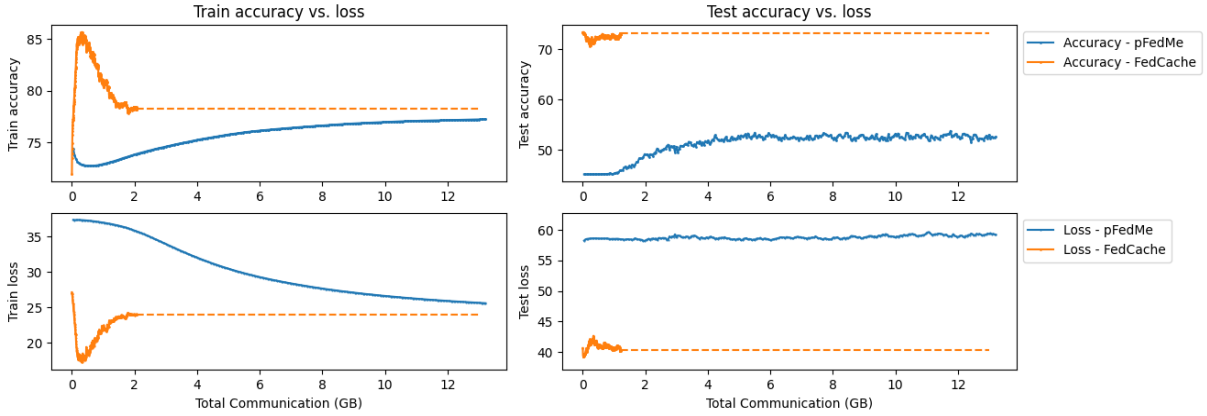
In the subsequent part of this section, data-level comparisons will be made through the illustrations below to specifically illustrate the benefits of FedCache in terms of communication efficiency. According to Table 4-7, the communication overhead incurred by the FedCache model to reach convergence in terms of accuracy is about 0.8 GB when using the IMDB data-set and the number of clients is 20, while the communication overhead incurred by the FedAvg, pFedMe, and MTFL models under the same conditions is 6.3 GB, 6.8 GB, and 6.4 GB, respectively, which is roughly 8 times the communication overhead of FedCache. Meanwhile, the communication overhead incurred on the SST-2 data-set when FedCache reaches convergence is about 0.06 GB, while the communication overheads of

FedAvg, pFedMe, and MTFL are about 1.2 GB, 1.1 GB, and 1.3 GB, which are about 20 times higher than the communication overhead of FedCache, and these properties can be further verified in Figure 4-8, Figure 4-9 and Figure 4-10 below. As shown in Figures 4-11, Figure 4-12, and Figure 4-13 below, the convergence curve of FedCache is much steeper than that of FedAvg, pFedMe, and MTFL, which is due to the fact that FedCache employs a lightweight protocol that transmits only logits and hashes and does not transmit model features of larger sizes such as model parameters. Meanwhile, the number of clients and the size of the data-set are positively correlated with the communication overhead, and in real-world scenarios, more clients tend to participate in federated learning, and the total amount of data involved will be more, so the traditional federated learning architecture tends to incur more communication overhead, whereas the communication overhead of FedCache stabilizes at a relatively low level.



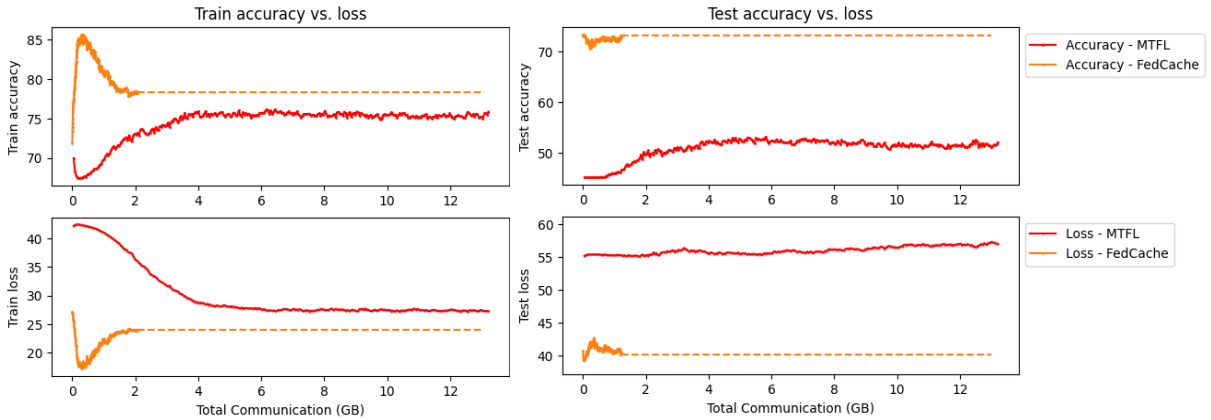
(a) Training curves (b) Testing curves

Figure 4-8 Comparison of FedAvg with FedCache on IMDB data-set



(a) Training curves (b) Testing curves

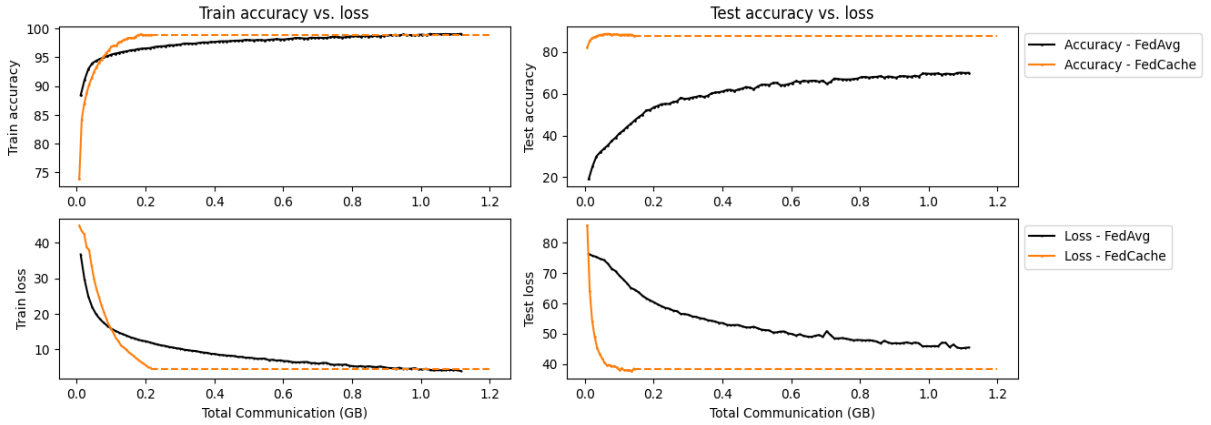
Figure 4-9 Comparison of pFedMe with FedCache on IMDB data-set



(a) Training curves (b) Testing curves

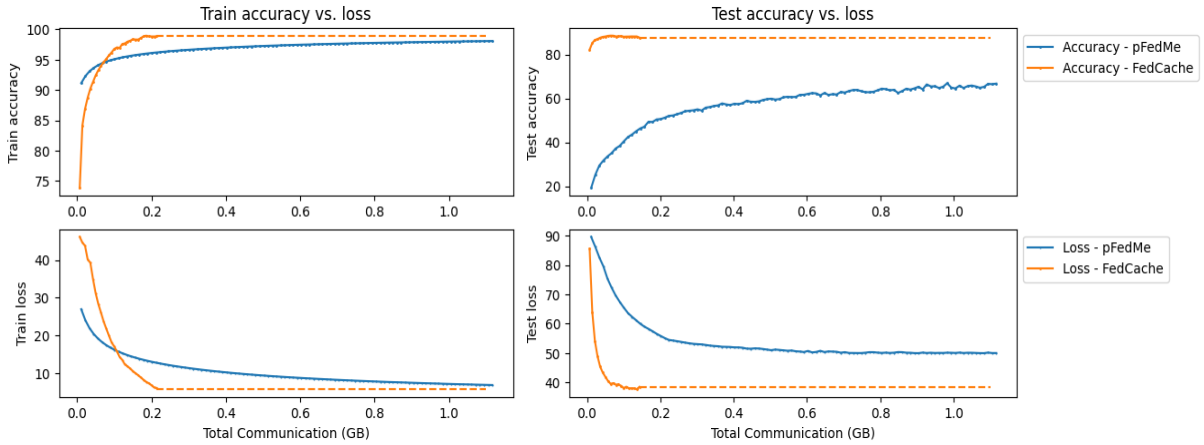
Figure 4-10 Comparison of MTFL with FedCache on IMDB data-set

Figure 4-8, Figure 4-9, and Figure 4-10 shown above are the comparative experiments on FedCache, pFedMe, FedAvg, and MTFL architectures performed in this experiment on the IMDB data-set. In particular, the experimental curves of FedCache architecture appear in the legends of MTFL, FedAvg, and pFedMe to make it easy to compare the characteristics of FedCache architecture and its advantages in performance. Figure 4-11, Figure 4-12, and Figure 4-13 shown below are the results of the experiments based on the SST-2 data-set. First, experiments on multiple example data-sets can be performed to verify the generalization of the FedCache architecture in NLP scenarios to some extent. Further, the difference in FedCache architecture on different data-sets can be characterized by comparing the experimental results on the IMDB data-set and the SST-2 data-set.



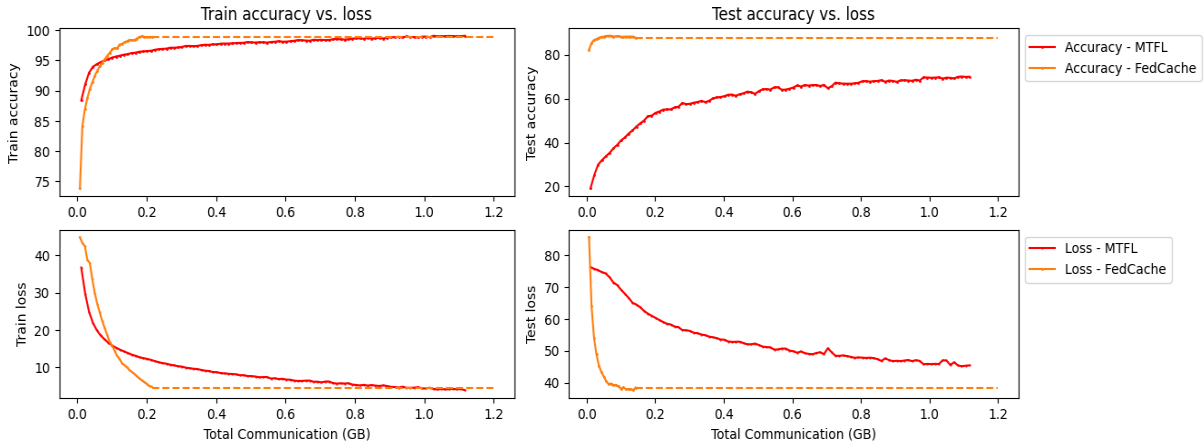
(a) Training curves (b) Testing curves

Figure 4-11 Comparison of FedAvg with FedCache on SST-2 data-set



(a) Training curves (b) Testing curves

Figure 4-12 Comparison of pFedMe with FedCache on SST-2 data-set



(a) Training curves (b) Testing curves

Figure 4-13 Comparison of MTFL with FedCache on SST-2 data-set

4.5 Summary of Chapter

This chapter focuses on the performance analysis of federated learning architectures through experiments. Firstly, the construction of non-IID data-sets and the basic experimental settings are introduced, and secondly, based on the experimental setup, FedCache and other federated learning architectures are implemented in NLP scenarios respectively. Then, the FedCache, FedAvg, pFedMe, and MTFL architectures are compared on two different example data-sets. The experimental results show that the architectures can save 1 to 2 orders of magnitude in communication overhead and achieve higher model accuracy.

5 Implementation of Movie Review Classification Application

This chapter will introduce the movie review sentiment classification application constructed for the dissertation task requirements, which integrates the work results of the text sentiment classification task and provides a query and detection interface to satisfy the user's functional modules for browsing data, viewing features, and analyzing complex English movie reviews. The application provides users with a text classification model based on the FedCache architecture, and users can input English reviews for sentiment analysis and obtain recognition results. Meanwhile, the application will provide users with a data collection interface. Finally, this chapter tests this application and shows the effect of the application.

5.1 Introduction

This chapter will introduce the movie review sentiment classification application built for the task requirements of this dissertation, which is designed to integrate the work and results achieved during the text sentiment classification task in the form of an application, and at the same time to meet the needs of users to browse the data, view the features, and analyze the complex English movie reviews by using the text sentiment classification by providing them with the corresponding query and detection interfaces. analyzing complex English movie reviews using textual sentiment classification. The application provides the user with the text classification model obtained from the training of the FedCache architecture implemented in this dissertation, and the user can directly login to the application to input the content of the English comments that need to be analyzed in terms of sentiment, and then the application will automatically give the identification results of the model; at the same time, the application also provides the relevant interfaces for data collection, so that every time the application is opened, the user can receive the latest movie introductions and the relevant comment messages. The main contents of this chapter include: introducing the background of the application requirements and the solutions provided by the application according to the requirements in Section 5.2; introducing the architectural design and functional modules of the application in Section 5.3; introducing the related technologies used in the application and the implementation methods of the modules in the application in Section 5.4; and lastly introducing the functions of the application modules and the test results of the module functions in Section 5.5. Finally, in section 5.5, the functions of each application module and the test results of the module functions are presented.

5.2 Requirements of Movie Review Classification Application

This section will briefly introduce the requirement background and functional requirements of this movie review categorization application to pave the way for the later description of the functional implementation process of the application.

5.2.1 Requirement context of the application

With the rapid development and popularization of Internet application platforms and terminal devices such as smartphones and tablets, the amount of information posted on forums, blogs, WeChat, and other platforms has increased. People learn more about the trends of daily life, study, work, and other aspects based on the platform information, and make their own behavioral choices and judgments based on these contents.

Unlike other English-language movie review platforms on the Internet, Chinese movie reviews, especially short movie reviews, on IMDB's movie forums have received a high level of attention from netizens and movie lovers due to the large amount of information in the reviews, the fast update speed, the large number of reviewers, and the incisive and unique viewpoints. Relevant data show that the monthly independent coverage of IMDB has reached 200 million. Users are the senders and receivers of opinions. Before watching a movie, users can read and browse online friends' movie reviews to choose to watch a movie that meets their interests and tastes; after watching a movie, users can write a movie review, score the movie to express their feelings and opinions about the movie, give suggestions to other netizens and friends, and can also rate the movie reviews they have seen before as "useful" or "useless", thus affecting the ranking order of the movie review. Users can also rate a movie review as "useful" or "useless", which will affect the ranking order of the review. By commenting on movie reviews, users can also meet more like-minded movie fans, enriching users’ interests and improving users’ artistic appreciation.

As the content expressed in movie dramas is extremely rich, it causes viewers to watch the movie with different focuses and thinking perspectives. By analyzing the emotional tendency of movie reviews, moviegoers can better understand what other moviegoers think about the movie, and make up for the levels that their thinking has not touched. Movie producers can improve their artistic level according to the views and needs of moviegoers, produce better works for appreciation, and help the cultural atmosphere and artistic flavor of society.

The emotional classification of movie and TV reviews is different from other reviews. The performance is rich in its review characteristics, not only the review of the movie itself, such as pictures, plot, sound, special effects, etc., but also about other elements outside the movie, such as the director, actors, and so on. Compared with other reviews that focus only on the product itself, the characterization of the diversity of film and television reviews is more challenging. Currently, existing studies on text sentiment classification are based on centrally trained deep learning models. Although this training approach can obtain text classification models with high accuracy, the generalization ability of the trained models is weak due to the limited data-set, and thus performs poorly in high-intensity text classification tasks. In addition, the distributed traditional federated learning has risks such as data privacy leakage, so this dissertation implements a training approach based on cache-driven PFL, which yields a text classification model with strong generalization ability and high communication efficiency, and is suitable for performing high-intensity text classification tasks.

Network elements are facing the change from the situation filled with the traditional culture to the information explosion of modern network popular culture, and the emotional classification of English movie reviews, which is still in the development stage, especially for the evaluation of the emotional tendency of movie reviews, is gradually reflecting its important research significance and value. Therefore, the analysis of emotional tendency and emotional intensity of English movie reviews is very meaningful from the perspectives of both theoretical research and practical application.

5.2.2 Functional requirements for the application

In the previous sections, the current data-set related to movie review sentiment classification, the data content in the data-set, and the features used in the classification model have been introduced. As a system specialized in text categorization, users are most concerned about whether the system can provide classification services, and the ease of use of this service will also affect the user experience. If the system requires the user to input a large amount of feature information when utilizing the classification service of the application, the service is unfriendly to the user; if the detection service of the system only requires the user to input the basic content of the review, the system can give the identification results, then such a service is very user-friendly. In addition to the movie review sentiment classification service, the user may also need to view the movie synopsis, which identifies the movie that the user is interested in by using information such as the movie poster, movie cast, and plot synopsis, and

then the user accesses the movie's review information and analyzes it sentimentally. In addition to data viewing, the app is able to analyze the rating of each movie based on its reviews and sort the movies, and the app can also sort the movies based on their release date, which helps improve the user experience of the app.

Therefore, the application implemented in this dissertation needs the following features:

(1) Data Viewing Function: View the latest IMDB movie introduction, including the movie poster, actors, director plot synopsis, and other information for users to view and select.

(2) Movie review emotion categorization function: The user enters the content of the English review of the movie to be analyzed on the classification page, then selects the two text classification models provided by the application, the LSTM model or the GRU model, and finally clicks on the Analyze button, and the application automatically gives the results of the sentiment analysis and the time needed to analyze the text.

(3) Data acquisition function: The application uses the API of IMDB data-set through the protocol of asynchronous HTTP to get the JSON file containing the information of the best movie from the official website of IMDB, and then parses the file in the application to display the corresponding information of the movie in the application page. This data can be viewed by the user and can also be analyzed and given to a text categorization model for classification.

(4) Movie recommendation sorting function: The application has a built-in sorting algorithm interface, through which the sorting algorithm analyzes the movie information obtained from the JSON file, and can sort the movies according to their release time or average rating, which can facilitate users to quickly find the movies they are interested in.

5.3 Overall Design of the Application

In this section, the design modules of the Movie Review Categorization application will be specified, including the architectural design approach of the application and the design of the functionality of each module. The specific description of these designs will set the context for the later presentation of the functionality of the application.

5.3.1 Architecture design of the application

Based on the requirements analysis in Section 5.2, this dissertation adopts the MVVM architecture in system architecture design to build a movie review sentiment classification application. MVVM architecture is a simpler design pattern among common system architectures, based on which this chapter designs the movie review sentiment classification application architecture shown in Figure 5-1.

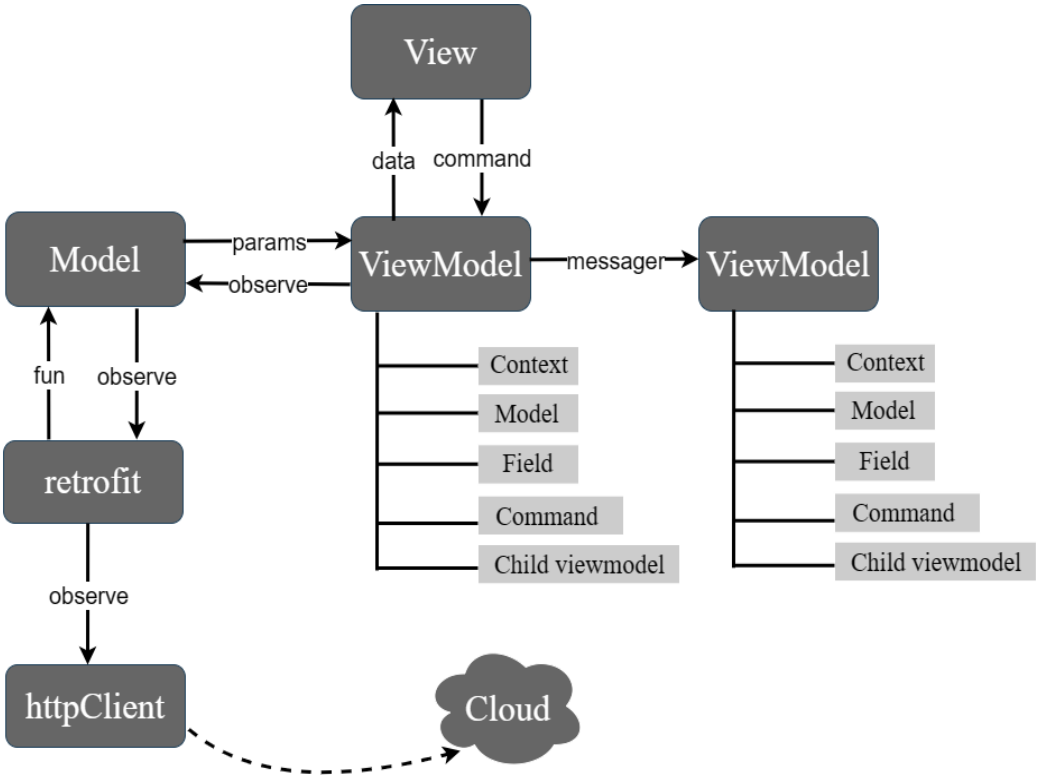


Figure 5-1 Schematic diagram of the MVVM architecture

A description of the main building blocks in the architecture is given below:

(1) Model layer: The model layer is mainly responsible for storing data, reading IMDB website data, manipulating database data, and I/O, usually set view model to get the data in this part.

(2) View layer: The view layer is the user interface, and contains the pure representation logic that implements the visual behavior. The “view” is completely unaware of business logic. That is to say, “view” never contains data and does not directly manipulate data, it communicates with the view model through data binding and does not communicate with the model directly.

(3) View model layer: The view model acts as a bridge in the whole MVVM architecture, responsible for presenting the data from the model to the view and passing the data input from the user back to the model. Viewing completes the bidirectional binding of public attributes and command data through view model. If a state change occurs, the view model notifies the "view" through events to respond to the latest model.

5.3.2 Functional modules design of the application

By analyzing the requirements of the movie review sentiment classification application in Section 5.2.2, four functional modules are designed in this section as shown in Figure 5-2: data viewing Module, data acquisition module, model classification module, and algorithm recommendation module.

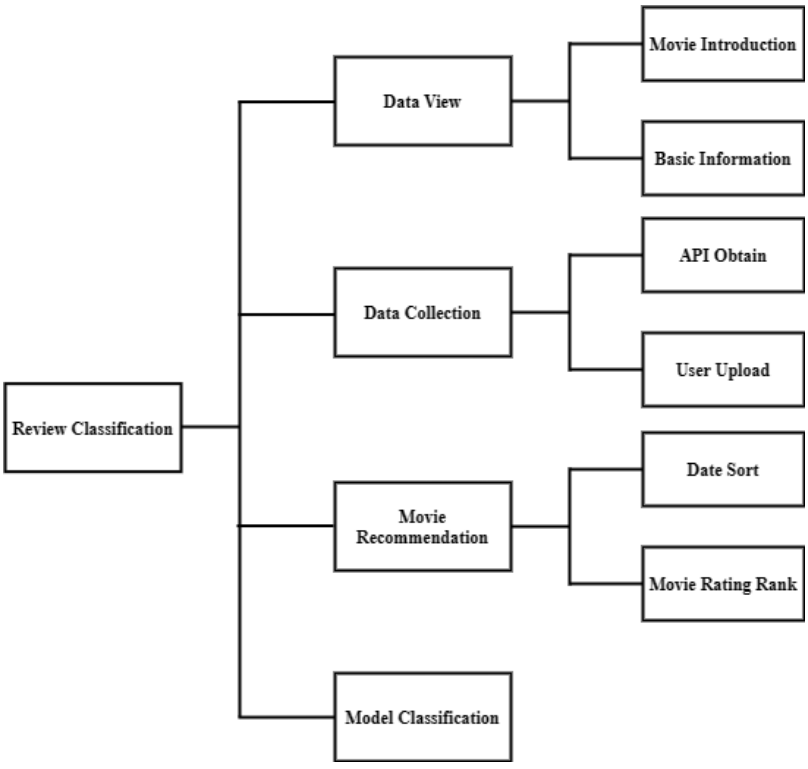


Figure 5-2 Application function module

The functions of each module are described below:

(1) Data viewing module: This module is mainly for users to view all the English movie reviews stored in the current database, and users can view the data already stored in the database through this module. Meanwhile, since the application collects officially

released classic movie information from the IMDB website from time to time, it also provides a function to view the latest English movie information in this module.

(2) Data collection module: This module is mainly for the collection of new data. Users can choose to use the asynchronous HTTP protocol interface that comes with the system to fetch the API of the IMDB website and then import it into the application database. If the user already has relevant data and the data content and characteristics meet the system requirements, users can choose to personally input the movie review information and upload the data file into the database via the upload page.

(3) Movie recommendation module: This module is mainly to sort the acquired movie data so that users can quickly find the movie they are interested in and read the basic information and comments of the movie. The application can sort the movies according to their ratings or date data, and the movies with high ratings or the latest movies will be on the top of the page for users to prioritize their viewing.

(4) Model text classification module: This module holds the pre-trained LSTM model and GRU model of FedCache for the user to classify the selected comments. The user enters the content of the English comments to be analyzed on this application page, and at the same time selects the text classification model to be used for this analysis, and clicks on the “Start” button, the system will automatically return the results.

5.4 Module Implementation Methodology

This section provides a specific explanation of how the modules are implemented, including the development environment of the application and the specific implementation of the modules. Specifically, the configuration of the development environment for testing is described, and all the steps for the realization of each module are explained.

5.4.1 Development environment

The movie review sentiment classification application designed and implemented in this dissertation is built based on the Android Studio integrated environment, which contains a series of IDE development environments that can support the rapid construction of Kotlin programs. For mobile development, the commonly used development language is Kotlin, which is cross-platform compatible and has more powerful features than Java in mobile development and can be applied to the development of different platforms. Android SDK is

compatible with most of the Android systems and is a common choice for development on Android phones today. In addition, this dissertation chooses two ways to test the Kotlin project, a virtual machine configured in Android Studio and a real cell phone, during which the IP address and port of the cell phone need to be changed manually in order to access the API interface provided by the IMDB website. The specific configurations of the virtual machines Pixel 6 Pro and HUAWEI Nova 7 Pro used for testing are shown in Table 5-1. In addition, the versions of the parts in Android Studio are Android Studio 2023.2.1, SDK Android 14.0, JDK 11, Gradle 7.0.2, SDK Build Tools 30.0.2, and the versions of the parts need to correspond to each other.

Table 5-1 Application test environment

Mobile Device	Operating System	CPU	RAM	ROM
Pixel 6 Pro	Android 17	Intel Atom	1GB	2GB
HUAWEI Nove 7 Pro	Harmony OS 4.0.0	Kirin 985	8GB	128GB

5.4.2 Module specific implementation methods

According to the description in 5.3.2, this system has four functional modules, including a data viewing module, a data acquisition module, a movie recommendation module, and a model text classification module. Each module has a unique function and role and cooperates to form a complete system. Next, this section will detail the implementation of the individual modules in order to better understand the overall architecture and functionality of the system. The organic combination of these modules enables the system to efficiently complete various tasks and provide better services for users.

The data viewing module is mainly provided to the user for viewing the contents of the JSON files retrieved from the IMDB official API, the data in these JSON files must be read and stored in the original order and format, and all the movies are placed together according to the category, the user can view the corresponding movie information according to their own needs. At the same time, each movie page contains a link button to the official IMDB website, after clicking on the button the application will automatically jump to the comment area of the corresponding movie, and the user can select their interest in the comments for sentiment analysis, the realization of the process shown in Figure 5-3.

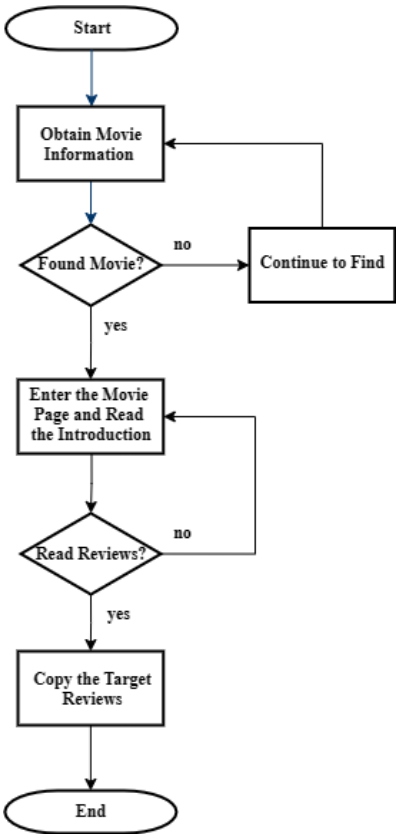


Figure 5-3 Data viewing module

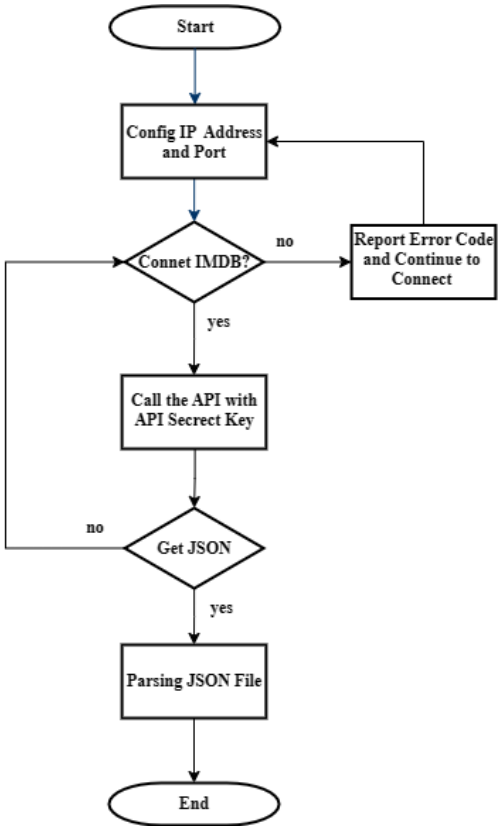


Figure 5-4 Data acquisition module

The data collection module mainly collects all the latest movie information selected by the official IMDB website. The module has a built-in API access secret key and API access URL of the IMDB website, which can obtain and parse the JSON file every time the user logs into the application to get the latest movie information from IMDB. If the corresponding movie content is not obtained due to network problems, the application will return the corresponding network error code and re-access it after some time, the implementation flow is shown in Figure 5-4.

The movie recommendation module is mainly to sort the movies acquired by the data collection module by date or rating, the default is to sort by date, the movies with earlier dates will be in the front; users can click the sorting button to change the sorting mode to sort by rating, the movies with high rating will be in the front. Movie recommendation is based on each movie category, different categories of movies will not be sorted together, which can make the application clearer. The implementation flow is shown in Figure 5-5.

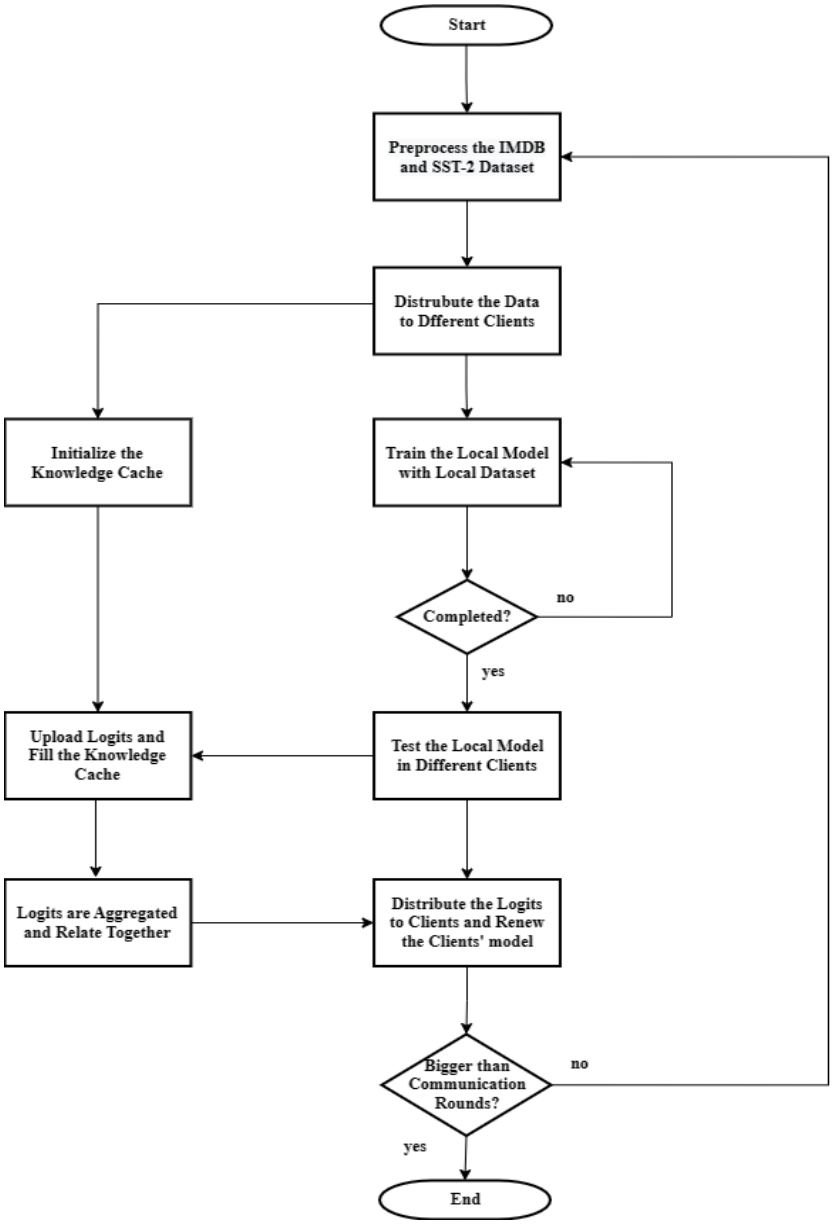


Figure 5-5 Model text classification process

The model text classification module mainly uses cache-driven PFL to obtain two text classification models for the client, namely, LSTM and GRU, which are then selected by the user. After determining the text classification model, the user inputs the movie review to be analyzed into the text box and clicks the Start Analysis button to get the final sentiment classification result, which indicates the sentiment tendency will be displayed in a quantitative manner below the text box. At the same time, the time spent on sentiment classification of the review is also displayed above the classification result. The implementation flow of the above function is shown in Figure 5-5.

5.5 Application Functionality Testing and Effectiveness

Based on what has been described earlier, this dissertation has successfully designed and implemented a movie review sentiment categorization application based on the Android Studio integrated development environment. In the next section, it will conduct comprehensive tests on these features and present the corresponding test results to verify the performance and accuracy of the application. To ensure the validity of the tests, this dissertation details the configuration of the test environment as shown in Table 5-1, including key information such as hardware devices, operating systems, programming languages, related libraries, and dependencies.

First of all, users need to configure their mobile devices for proper network configuration before using the features of the application. This includes setting the IP address and access port of the mobile device to ensure that the device can connect to the Internet without any problems. After completing the configuration, users should ensure that they can successfully access the IMDB website, which is a critical step in verifying the success of the configuration. Only when the user can access the IMDB website without any problems, it can ensure that the configuration is correct. Next, users can normally access the official API provided by IMDB, through which they can get the latest movie information, including movie ratings, cast, plot synopsis, and other rich content. In this way, users can make full use of the application's functions and enjoy a wealth of movie and TV information.

After the user opens the application, the user will enter the home page of the application. The right side of the home page of the application is the function navigation window, which is the text categorization function, mode switching, visiting the official website, and recommendation mode. The text classification application is the model text classification module, which is used to analyze the comments entered by the user and view the results of the feature analysis of the comments; the data view is the data view module, which is used for the user to view the data stored in the JSON file. After the user clicks on the movie users are interested in it will display some data statistics about the movie, displaying information about the current movie's poster, crew information, plot synopsis, and IMDB review links. At the same time, the app will recommend movies in each category according to their release date or rating. There are also display themes to choose from on the right side of the homepage, and users can switch between the night theme and the day theme. The details of the home page are shown in Figure 5-6. After entering the home page, users can test and use each function module of the application.

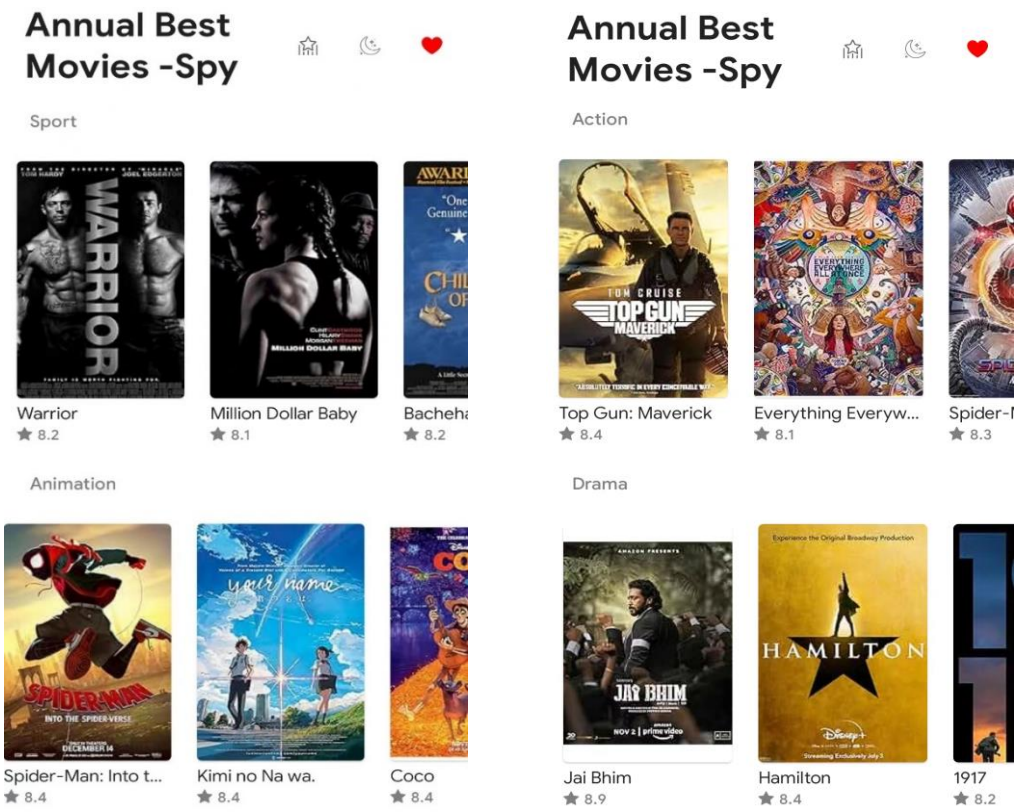


Figure 5-6 Application home page effect

5.5.1 Test of data viewing function

The data view function is for the user to view the data stored in the JSON file, the data is organized according to the movie category, and different categories of movies are stored separately, the movie information on the homepage mainly contains movie posters, movie titles, and movie ratings. The purpose of testing this module is to verify that it can correctly query the corresponding data in the database according to the user's selection. The a priori condition for the completion of the test is that the backend of the software is functioning normally and it can maintain the connection with the official website of IMDB. During the test, try to scroll up and down the main page of the application, after which the application will automatically display more movies in the category; try to scroll left and right to a certain category of movies on the main page of the application, the application will automatically display more movies in the category. Clicking on the poster of a movie of interest, the app will automatically switch to the main page of the movie, which contains all kinds of basic information about the movie, as well as a link to the IMDB comment section of the movie, and the test results are shown in Figure 5-7.

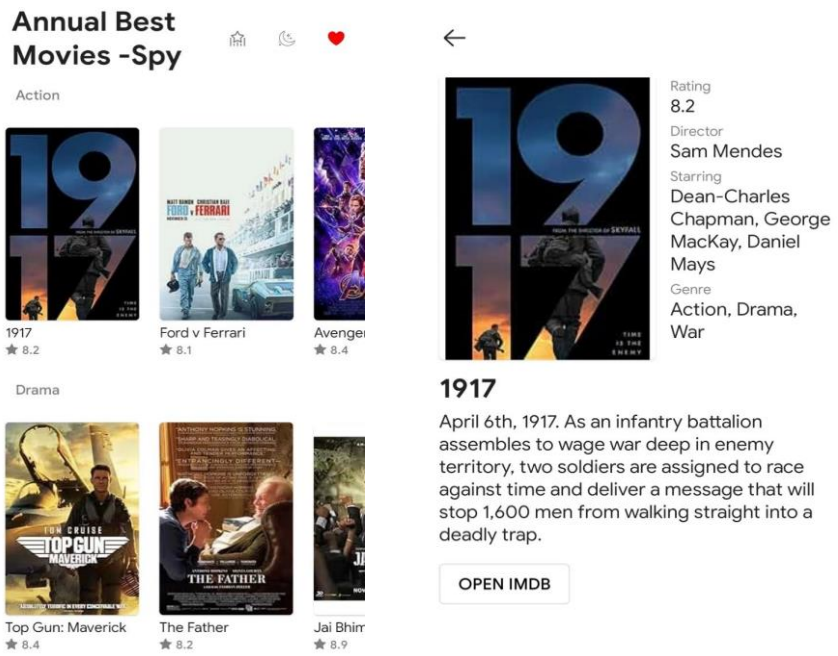


Figure 5-7 Result display of data viewing

As shown in Figure 5-8, after clicking the "OPEN IMDB" button, the application will enter the corresponding page of the movie on the IMDB website. On this page, there are selected movie clips, detailed plot introductions, movie ratings, and all English comments on the movie from other moviegoers, and users can select the comments they are interested in for sentiment analysis.

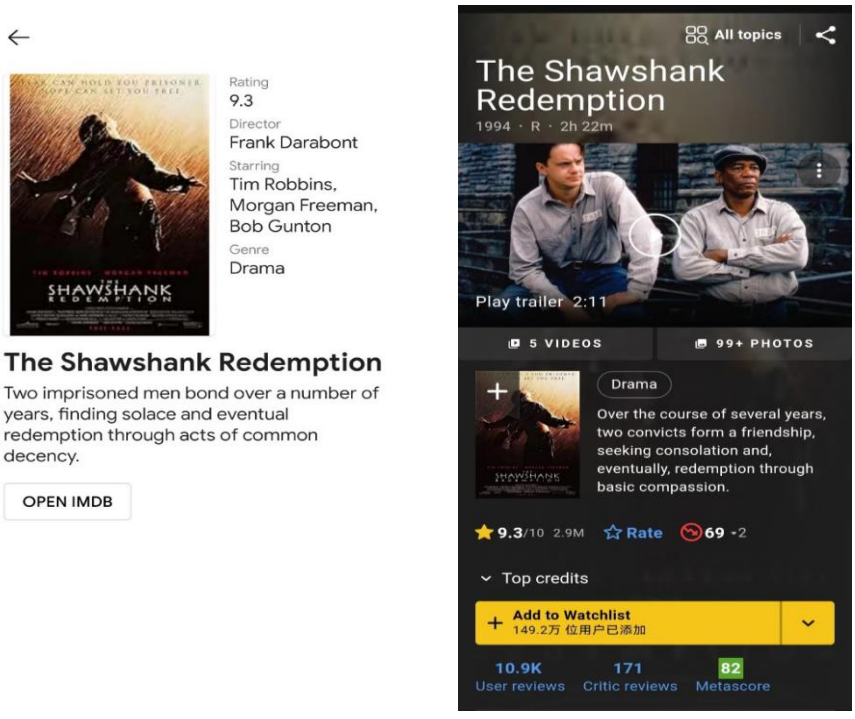


Figure 5-8 Detailed view of the movie

Based on the test results, it can be verified that the data viewing module of this application functions as expected and can display the corresponding correct results according to the user's selection.

5.5.2 Test of data acquisition function

As described in Section 5.4.2, the Movie Review Sentiment Classification application obtains a JSON data file by accessing the IMDB website's API and then parses that JSON file to write the corresponding movie information to a local database. The data collection is performed every time a user logs into the application to get the latest movie information from IMDB. If the corresponding movie content is not obtained due to network problems, the application will return the corresponding network error code and revisit it after some time, and the test results are shown in Figure 5-9.


```
[
  {
    "actors": [
      "Toshirô Mifune",
      "Eijirô Tôno",
      "Tatsuya Nakadai"
    ],
    "desc": "A crafty ronin comes to a town divided by two criminal gangs
and decides to play them against each other to free the town.",
    "directors": [
      "Akira Kurosawa"
    ],
    "genre": [
      "Action",
      "Drama",
      "Thriller"
    ],
    "image_url": "https://m.media-
amazon.com/images/M/MV5BZThiZjAzZjgtNDU3MC00YThhLThjYWUtZGRkYjc2ZWZ1OTVjXk
EyXkFqcGdeQXVyNTA4NzY1MzY@._V1_.jpg",
    "thumb_url": "https://m.media-
amazon.com/images/M/MV5BZThiZjAzZjgtNDU3MC00YThhLThjYWUtZGRkYjc2ZWZ1OTVjXk
EyXkFqcGdeQXVyNTA4NzY1MzY@._V1_UX182_CR0,0,182,268_AL__QL50.jpg",
    "imdb_url": "/title/tt0055630/",
    "name": "Yôjinbô",
    "rating": 8.2,
    "year": 1961
  },
  {
    "actors": [
      "Tyrone Power",
      "Marlene Dietrich",
      "Charles Laughton"
    ],
    "desc": "A veteran British barrister must defend his client in a
murder trial that has surprise after surprise.",
```

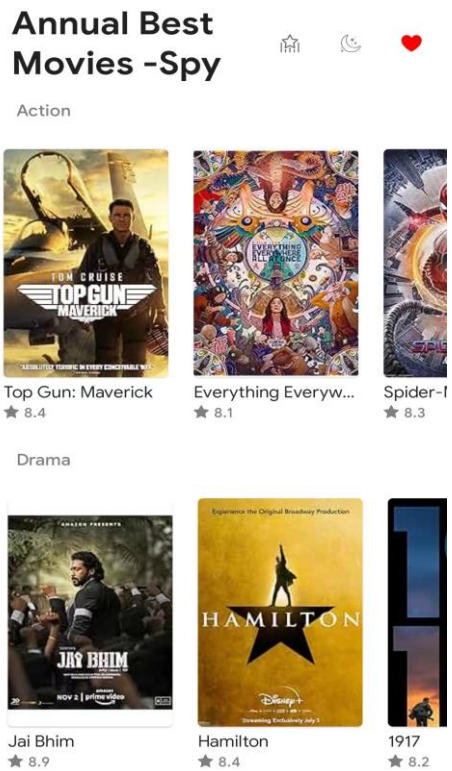



Figure 5-9 Data acquisition result display

According to the above analysis results, when the client successfully connects to IMDB, it can successfully obtain and analyze the JSON file obtained from calling the API, and arrange the data neatly on each page of the application.

5.5.3 Test of model text categorization function

Model text classification is the core function of the whole movie review sentiment classification application, the application provides users with two text classification models, LSTM and GRU, and the quantitative results of the analysis of each model and the time spent are given in the front-end page. Users enter the text they want to categorize in the input box, and then select the corresponding text classification model, and the application will give the classification results. The content of IMDB English movie reviews used for text classification and the model text classification results are shown in Figure 5-10.

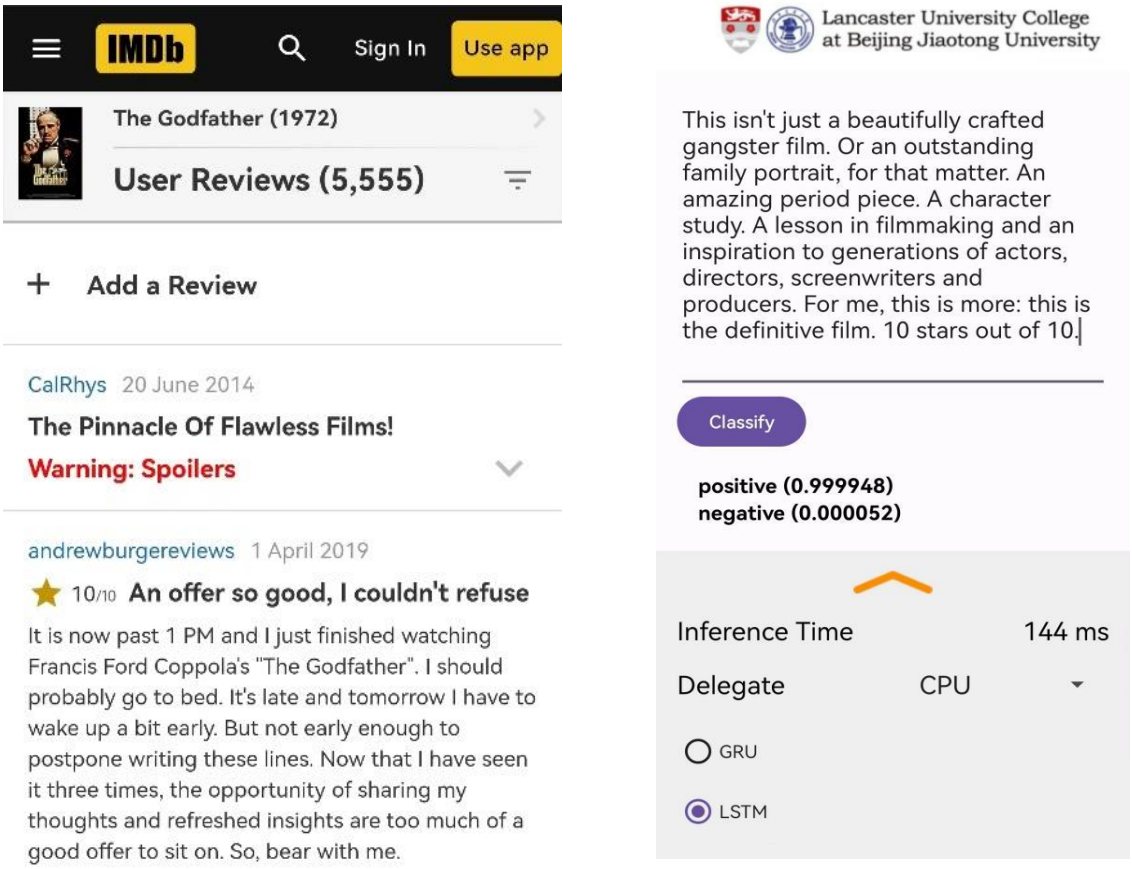


Figure 5-10 Display of model text categorization results

Based on the above results, after the user inputs the content of the English comment to be analyzed into the text box, the sentiment tendency of the selected English comment can be successfully analyzed and identified, and the sentiment tendency value and the time spent on analyzing the text are neatly arranged on the analysis page.

5.5.4 Test of movie recommendation function

Many information retrieval applications add an auto-recommendation function, which can sort the contents that users may retrieve according to the reference data so that users can retrieve their interested contents more conveniently. This application takes the movie release date and movie rating provided by IMDB to sort the movies in each category. By default, the application refers to the movie release date to sort, users can click the "Recommend" icon in the upper right corner of the application to switch the sorting mode of the movie, and the first movie is the newest released movie or the highest rated movie. In the application test, the sorting results of the two recommendation methods are shown in Figure 5-11.

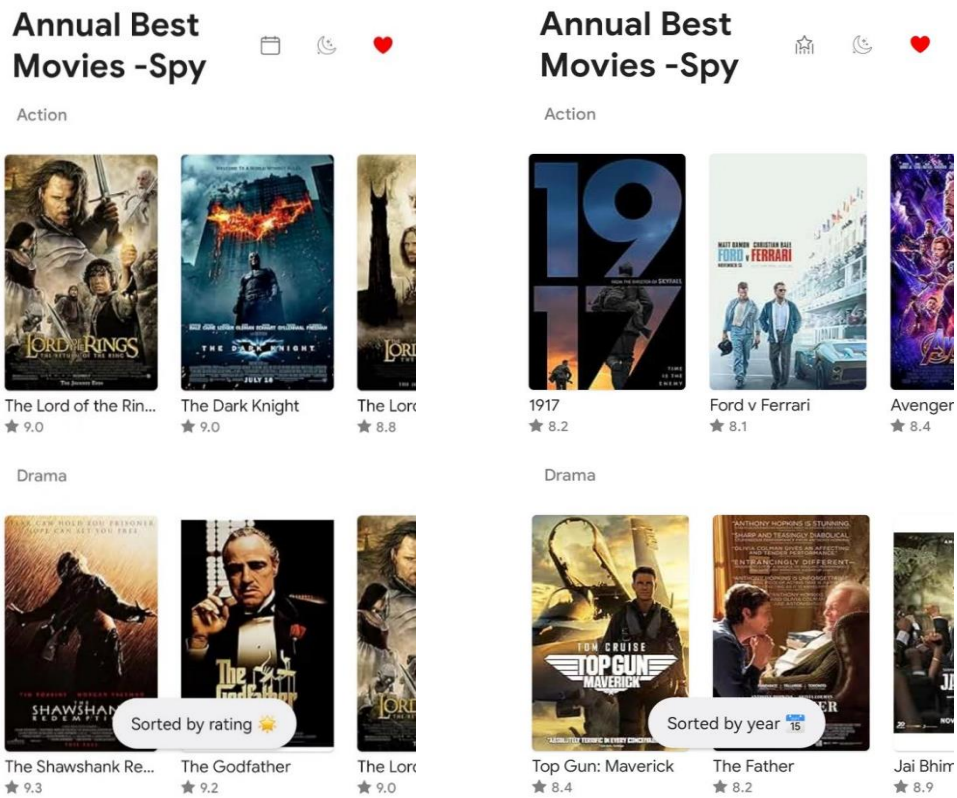


Figure 5-11 Movie recommendation result display

According to the above results, the user selects the sorting method according to the demand, and then the movies will be sorted according to their respective dates or ratings according to the categories respectively. The left side of Figure 5-11 is the sorting result obtained concerning the movie rating data, and its right side is the sorting result obtained with reference to the date data of the movie, and the main page defaults to sort movies according to the date.

5.6 Summary of Chapter

This chapter describes the application of FedCache federated learning architecture to NLP scenarios and presents the process of designing, implementing, and testing a movie review sentiment classification system. In the design section, this chapter firstly designs the general architecture of the application and then divides the application into four modules according to its functions for specific analysis as well as design, namely, the data collection module, data viewing module, movie recommendation module, and model text categorization module. In the implementation part, this chapter implements the sentiment analysis of text and the visual presentation of the corresponding results. In the testing section, this chapter tests the functionality of all four modules of the application as well as the non-functional modules of the application, showing the good performance of the application in text categorization scenarios.

6 Conclusion and Prospects for Future Work

This chapter is the final chapter of the whole dissertation, which will conclude the work done in the whole dissertation. Meanwhile, this chapter will analyze the shortcomings of this research and look forward to the future research direction.

6.1 Conclusion of Work of This Dissertation

This dissertation focuses on the problem of client heterogeneity and high communication overhead that often exists in federated learning in practical application scenarios, improves the traditional federated learning architecture, implements FedCache, a PFL architecture based on cache-driven learning, and designs and implements a movie based on the model trained by the FedCache architecture using Android Studio as a platform.

The results of the work in this dissertation are summarized below:

(1) In the research of solving the client-side non-IID problem in federated learning, the current federated learning pFedMe architecture mitigates the impact of client-side non-IID sample data and model heterogeneity on the performance of federated learning through the introduction of loss-function-based Moreau envelopes optimization for constructing client-side heterogeneous models, however, the high communication cost and client-side heterogeneity support accompanying the pFedMe, FedAvg and MTFL architectures accompanied by high communication cost and client heterogeneity support need to be further improved, so this dissertation implements FedCache, a cache-driven PFL architecture based on the idea of knowledge distillation algorithms, and generalizes the analysis in terms of communication cost and model heterogeneity support.

(2) In the implementation of FedCache based on NLP scenarios, the knowledge cache is used to store the newly extracted knowledge uploaded by the clients and to obtain the desired personalized knowledge from the samples with the closest neighbor correlation to the specified sample data. Based on this, knowledge distillation is performed on the local models of each client for personalization construction and optimization. FedCache, as implemented in this dissertation, is the first attempt at cache-driven PFL architecture based on NLP scenarios, which allows sample-level logits interactions without parameter transfer or public data-sets.

(3) The FedCache architecture is implemented on the IMDB as well as the SST-2

data-set, and the superiority of the FedCache architecture compared to the traditional federated learning architecture is verified through comparative experiments. The above comparative experiments show that FedCache can achieve model accuracy comparable to state-of-the-art federated learning architectures in NLP scenarios while reducing the communication cost by an order of magnitude.

(4) The movie review sentiment classification application based on the NLP model obtained by training the FedCache architecture was developed on the Android Studio platform. The application has four main functional modules: data viewing, data collection, movie recommendation, and movie review sentiment classification. The FedCache architecture is applied to real text classification scenarios, and the performance is used to show the practicality of the FedCache architecture.

6.2 Prospects for Future Work

Federated learning is an emerging field of machine learning with wide application value, and lightweight PFL algorithms have been widely noticed and researched as a major challenge in federated learning in recent years. Although this dissertation makes improvements in model heterogeneity supportability and communication efficiency, there are still many issues need to be improved. They can be concluded in the following three aspects:

(1) FedCache only performs knowledge extraction based on local customer examples and ignores knowledge learning for global generalization. As a result, FedCache is only suitable for client-specific tasks, but not suitable for general tasks that require global generalization. The generalization performance of FedCache can be improved by providing additional information, such as global parameters or global public data.

(2) FedCache currently only has practical applications in the field of NLP and CV, and there is no research precedent for multimodal applications. The main difficulty is that different kinds of data-sets are mixed, which makes it difficult to carry out effective knowledge extraction and knowledge distillation.

(3) Although FedCache reduces the communication overhead of federated learning to a lightweight degree, the communication generated by the hash value transmission during FedCache initialization is still very large and occupies the vast majority of its communication overhead, so replacing the hash value with another, more lightweight sample feature vectors is a very valuable direction of work for research.

References

[1] Arivazhagan, M. G., Aggarwal, V., Singh, A. K., et al. (2019). Federated learning with personalization layers[J]. arXiv preprint arXiv:1912.00818, 2019.

[2] Brisimi, T. S., Chen, R., Mela, T., et al. (2018). Federated learning of predictive models from federated electronic health records[J]. International Journal of Medical Informatics, 2018, 112: 59-67.

[3] Chen, Y., Qin, X., Wang, J., et al. (2020). Fedhealth: A federated transfer learning framework for wearable healthcare[J]. IEEE Intelligent Systems, 2020, 35(4): 83-93.

[4] Dinh, T. C., Tran, N., Nguyen, J. (2020). Personalized federated learning with moreau envelopes[J]. Advances in Neural Information Processing Systems, 2020, 33: 21394-21405.

[5] Duan, M., Liu, D., Chen, X., et al. (2020). Self-balancing federated learning with global imbalanced data in mobile systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 32(1): 59-71.

[6] Fallah, A., Mokhtari, A., Ozdaglar, A. (2020). Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach[J]. Advances in Neural Information Processing Systems, 2020, 33: 3557-3568.

[7] Huang, Q., Li, Z., Xie, W., et al. (2020). Edge computing in the smart home[J]. Computer Research & Development, 2020, 57(9): 1800-1809. (In Chinese).
（黄倩怡，李志洋，谢文涛等. 智能家居中的边缘计算[J]. 计算机研究与发展，2020, 57(9): 1800-1809.）

[8] Huang, Y., Chu, L., Zhou, Z., et al. (2021). Personalized cross-silo federated learning on non-iid data[C]. In Proceedings of the AAAI conference on artificial intelligence. 2021, 35(9): 7865-7873.

[9] Jeong, E., Oh, S., Kim, H., et al. (2018). Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data[J]. arXiv preprint arXiv:1811.11479, 2018.

[10] Jiang, Y., Konečný, J., Rush, K., et al. (2019). Improving federated learning personalization via model agnostic meta learning[J]. arXiv preprint arXiv:1909.12488, 2019.

[11] Kairouz, P., McMahan, H. B., Avent, B., et al. (2021). Advances and open problems in federated learning[J]. Foundations and Trends in Machine Learning, 2021, 14(1-2): 1-210.

[12] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al. (2017). Overcoming catastrophic forgetting in neural networks[J]. Proceedings of the National Academy of Sciences, 2017, 114(13): 3521-3526.

[13] Li, D., Wang, J. (2019). Fedmd: Heterogenous federated learning via model distillation[J]. arXiv preprint arXiv:1910.03581, 2019.

[14] Liang, P. P., Liu, T., Ziyin, L., et al. (2020). Think locally, act globally:erated learning with local and global representations[J]. arXiv preprint arXiv:2001.01523, 2020.

[15] Lin, T., Kong, L., Stich, S. U., et al. (2020). Ensemble distillation for robust model fusion in federated learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 2351-2363.

[16] Malkov, Y. A., Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 42(4): 824-836.

[17] Mansour, Y., Mohri, M., Ro, J., et al. (2020) Three approaches for personalization with applications to federated learning[J]. arXiv preprint arXiv:2002.10619, 2020.

[18] McMahan, H. B., Moore, E., Ramage, D., et al. (2017). Communication-efficient learning of deep networks from decentralized data[C]. In Proceedings of the Artificial intelligence and statistics. PMLR, 2017: 1273-1282.

[19] Nichol, A., Achiam, J., Schulman, J. (2018). On first-order meta-learning algorithms[J]. arXiv preprint arXiv:1803.02999, 2018.

[20] Rivest, R. L., Adleman, L., Dertouzos, M. L. (1978). On data banks and privacy homomorphisms[J]. Foundations of Secure Computation, 1978, 4(11): 169-180.

[21] Sanh, V., Debut, L., Chaumond, J., et al. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter[J]. arXiv preprint arXiv:1910.01108, 2019.

[22] Shoham, N., Avidor, T., Keren, A., et al. (2019). Overcoming forgetting in federated learning on non-iid data[J]. arXiv preprint arXiv:1910.07796, 2019.

[23] Smith, V., Chiang, C. K., Sanjabi, M., et al. (2017). Federated multi-task learning[J]. Advances in Neural Information Processing Systems, 2017, 30.

[24] Wu, Z., Sun, S., Wang, Y., et al. (2024). FedCache: A knowledge cache-driven federated learning architecture for personalized edge intelligence[J]. IEEE Transactions on Mobile Computing, 2024.

[25] Xu, C., Mao, Y. (2020). An improved traffic congestion monitoring system based on federated learning[J]. Information, 2020, 11(7): 365.

[26] Yang, H., He, H., Zhang, W., et al. (2020). Fedsteg: A federated transfer learning framework for secure image steganalysis[J]. IEEE Transactions on Network Science and Engineering, 2020, 8(2): 1084-1094.

[27] Yang, Q., Liu, Y., Chen, T., et al. (2019). Federated machine learning: Concept and applications[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(2): 1-19.

[28] Yao, X., Sun, L. (2020). Continual local training for better initialization of federated models[C]. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP). IEEE, 2020: 1736-1740.

[29] Zhao, Y., Li, M., Lai, L., et al. (2018). Federated learning with non-iid data[J]. arXiv preprint arXiv:1806.00582, 2018.

[30] Zhu, Z., Hong, J., Zhou, J. (2021). Data-free knowledge distillation for heterogeneous federated learning[C]. In Proceedings of the International conference on machine learning. PMLR, 2021: 12878-12889.

Acknowledgements

Time flies, the magnolia blossom blooms again, and it's time to graduate from undergraduate school. At the beginning of April in Weihai, it still drizzles from time to time, and although my mood is just like the weather, the little sparrows that walk with me on campus every morning make me feel that the time is just right and the future is promising. The four years of undergraduate life are short and hurried, in these four years I have been lost and confused, helpless and uncertain, but through continuous exploration, now in research and life have a new understanding and a lot of harvest. As I am about to leave the university campus and enter graduate school, I have mixed feelings in my heart. I am looking forward to welcoming another new chapter in my life and expecting the future to be as I imagined, and I am also worried about the gap between reality and my dream. Undergraduate four years there are many past events worth cherishing, and there are also too many people I want to thank.

First of all, I would like to thank my undergraduate advisor, Prof. Gao Bo, for his devoted teaching during the four years of my undergraduate studies. Mr. Gao helped me a lot in both research and daily life. Mr. Gao's profound academic accumulation, integrity and scientific research attitude to defying difficulties have set a good example for me and benefited me a lot.

Secondly, I would like to thank Wu Zhiyuan. Mr. Wu gave me detailed and helpful advice on my research, especially during my dissertation. I would like to thank Mr. Wu for his diligent and practical research life, which is an example for me to follow. My daily communication with Mr. Wu has solved a lot of my confusion in my research. I would also like to thank all the students in the lab for their understanding and help.

I would also like to thank my family, my parents are my warmest harbor, you teach me to treat people, give wireless tolerance and care, always support my choice, sure of my achievements. Your unreserved support and dedication in my academic career have always been the driving force for me to move forward, thanks for your support and understanding.

Finally, I would like to express my best wishes and thanks to all the teachers who took time out of their busy schedules to review this dissertation.