A Federated Learning approach for text classification using NLP

No Author Given

No Institute Given

Abstract. Text classification is important in many aspects of NLP, such as word semantic categorization, emotion analysis, question answering, and conversation management. Law, health, and marketing are just a few of the professions that have made use of it throughout the last century. We focused on emotion analysis in this research, which includes categories like happiness, sadness, and more. We investigated the precision of three alternative models for assessing the emotional tone of written material. Deep learning models like GRU (Gated Recurrent Unit) and CNN (Convolutional Neural Network) are used in conjunction with the Bi-LSTM (Bidirectional LSTM) model. These three major deep learning architectures have been extensively studied for classification applications. Finally, the model with the greatest accuracy on the dataset was trained using Federated Learning. Using the FL approach, more data has to be collected, eliminating data gaps and ensuring data security. The focus of this research is to increase the model's accuracy by collaborating with FL approaches while maintaining data confidentiality.

Keywords: : NLP · Federated · Deep learning · CNN · GRU · Bi-LSTM · ML

1 Introduction

Text may be categorized using a variety of approaches, such as the number of words in a paragraph or the genre of the text. It is possible to employ text classification for a variety of NLP tasks, such as sentiment analysis, question answering, and subject identification. A text classification issue is often broken down into four stages: dimension reduction, text preparation, text classification, and text assessment. Text categorization may be useful in natural language processing (NLP) tasks including sentiment analysis, question answering, and topic labeling. Text classification problems often include four-step processes: dimension reduction, text preparation, classification, and evaluation. However, the main issue is uploading or sharing text data in order to improve model performance, which has previously been achieved utilizing deep learning models. It is not always practicable because of the various privacy concerns of the customers. It is done with the help of a sophisticated deep learning model. Despite the fact that deep learning models have already attained the highest levels of performance in text categorization, uploading and sharing text data is still a significant barrier. This method is implemented via the usage of a pre-trained deep learning model. One way to keep text classification private is to use federated learning. With the introduction of Google's federated learning in 2016, service providers may benefit from models that have access to private data without needing to access such data or aggregate all the information into a single dataset. If many devices were trained together, the model would be distributed to each device for additional refinement and improvement[1]. There are numerous clients working together to solve a machine learning problem under the direction of one central server or service provider in the Federated Learning model. Training machines via federated learning is a decentralized method. No third party receives any of the raw data from any of the clients. The framework's database was used to train local models. When the central server has acquired gradients from all platforms and has completed collecting them, a global model update is generally done. Once this procedure is complete, a new set of parameters is transmitted to each platform for the next round of modeling training.

2 Related Work

Liu and Miller[2] presented the federated pre-training and fine-tuning of the BERT model using clinical data. They were the first ones who used the federated settings for a large transformer model. In a federated manner, they trained the clinical corpus with both pre-training and fine-tuning methods while using multiple silos of data. Even though the original BERT model was trained on books and Wikipedia, they did it on clinical data using MIMIC-III discharge summary in a federated manner, for which they conducted 6 different experiments.

Leroy et al. [3] proposed in their paper a realistic technique for dealing with out-of-domain issues utilizing continuous speech-based models like wake word detectors. Precision is more critical since the model may be utilized at any time. An embedded wake word detector is used to investigate the usage of federated learning. They offer a stochastic gradient descent optimization method that uses both local and global per-coordinate gradient scaling to increase convergence. How to train a wake word detector with entirely decentralized user data will be the subject of the first section of this article. Next, we'll talk about federated optimization and how to replace its global averaging with an adaptive averaging algorithm inspired by Adam.

In the paper by Basaldella et al. [6], federated learning uses client devices to train a neural network model, which is then delivered back to the machine. The learning model's owner is the server. The server compiles these models into a single global model and delivers it back. We repeat this until the global model reaches our desired accuracy. Any transaction and model data in FL is stored on the server. The records and models stored on the server will be lost or destroyed if it is damaged or malfunctioned. If this happens, the model will need to be retrained.

FedNer[7] is a framework where the server usually coordinates multiple clients for the purpose of local model updating and global model sharing. Basically, here the clients are different medical platforms, and they are used to train their local models with the data that is stored privately. FedNer is also implemented in different places like Dataset and Experimental Settings, Experimental Results, Influence of Training Data Size, Model Decomposition Strategy, Influence of Overlapped Entity Number, and Generalization of the FedNer Framework.

Healthcare access[8] is proving to be a significant difficulty. Through social media, users from all around the world may share their views and ideas. The categorization job is the most important aspect of this research. Text data may come from a variety of places, including emails, chats, site data, customer evaluations, and feedback. The accuracy of the algorithms used in DL is used to evaluate the performance for text categorization. The process of text classification includes categorizing news, assessing the text based on user opinion, and classifying the content. For NLP tasks, the standard RNN model failed miserably. Deep learning in combination with natural language processing (NLP) offers great potential for text classification challenges.

3 Proposed Methodology

3.1 Data-Set

We have a fairly well-labeled dataset that comprises around 21450 unique data points of multi-class emotions. Our dataset contains more than simply sentiment classification; there is more to a sentence's sentiment than just positive, negative, and neutral sentiment. We were trying to figure out what was going on in the mind of the person who wrote the words. Sadness , rage, surprise, joy, love and fear are just a few of the emotions represented in this multi-class data collection. We chose to create a csv file from all of the text files since arranging the data from a text file is very time-consuming and difficult. The csv file has 2 columns and they are text and emotions. The Emotions column has a number of different categories, ranging from happy to sadness, to love and fear.

Text	Emotion
i didn't feel humiliated	sadness
i can go from feeling so hopeless to so damned	sadness
i'm grabbing a minute to post i feel greedy wrong	anger
I am ever feeling nostalgic about the fireplace	love
i am feeling grouch	anger

3.2 Data Pre-processing

We have previously processed the data before delving into the model's work. This data processing will allow the models to operate more effectively and with fewer mistakes. To begin, we eliminated the emoji symbols from each sentence in the dataset so that the models could only deal with text. After that, we removed any URLs from the texts. We do not require the URL in our dataset since it does not indicate any kind of emotion. Unnecessary punctuation marks may appear in the text. Unnecessary punctuation may lead us to wrong evaluation , so we eliminated all unnecessary punctuation. We divided the data into three sets: the training dataset (which contains about 70 percent of the whole dataset), validation dataset (which contains about 20 percent of the whole dataset) and the test dataset (which contains 10 percent of the total dataset). We will use the training dataset to train the models, while the testing dataset will be used to assess the model's accuracy after training. Finally, we ran the dataset through the tokenization algorithm. Tokenization is the process of breaking down a phrase, sentence, or text document into smaller pieces, such as individual words.

3.3 Bi-LSTM

The extraordinary capacity of LSTM to extract complex text information is crucial in text classification. Variable-length sequences benefit from the attention mechanism as well as a more even distribution of weight. We employed Word2vec weights that had been pre-trained with a large corpus, guaranteeing that the model was more accurate. The model's accuracy was evaluated using precision, recall, and the F1-score. For natural language processing (NLP) applications, the LSTM model plus a CNN proved to be unexpectedly successful. An increasing number of research have proven that sequence-based sentiment categorization may be achieved using LSTMs. The LSTM's accuracy is further hindered by its inability to distinguish between different relationships between document components. To solve this problem, we have used data preprocessing. Before being input into the model, the text was preprocessed. This includes, among other things, removing whitespace and unneeded words, converting other words to approximations, and minimizing duplicate words.

3.4 Fed Averaging

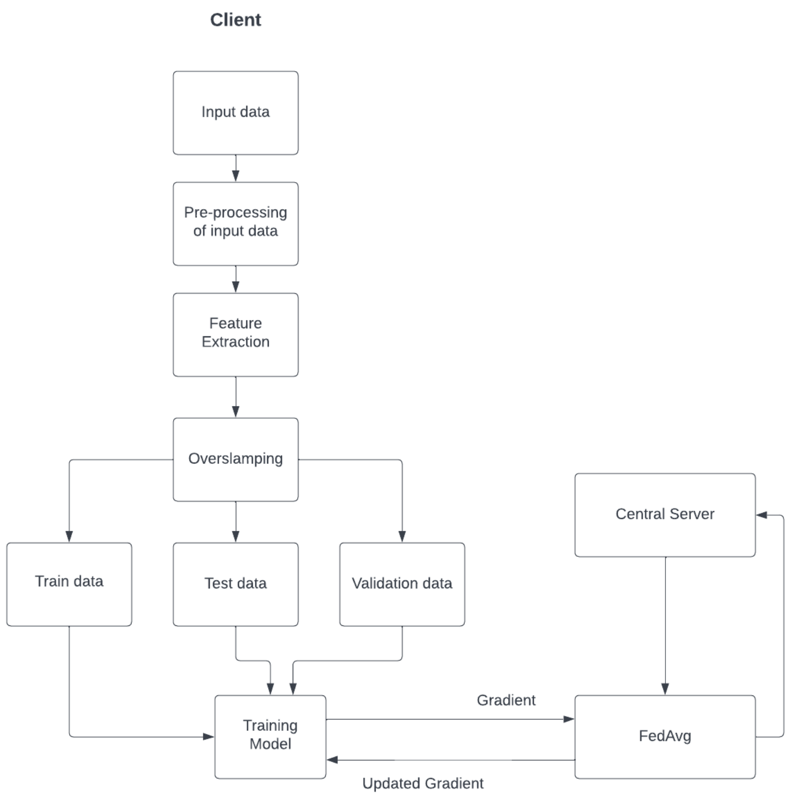


Fig. 1. Workflow Diagram

Our research workflow is shown in the flowchart. Our planned study on federated learning would include numerous clients, however, we are only displaying data for one. Initial processing will be performed on the supplied data. The pre-processed data is then used for the extraction of features. The extracted featured data is separated into three sets, and these sets are fetched into the training model. FedAvg, an algorithm used in the federated learning technique, is fed the gradient of the learned model. The gradient is sent to the central server. Each server transmits an epoch for a particular time to the central server. The central server updates gradients and sends them back to client-servers using the Fed Average Algorithm. Local servers provide data to the central one, resulting in a federated learning paradigm

3.5 GRU

GRUs are a gating mechanism that is used in RNN. There is a problem with short-term memory in RNN. When a sequence becomes large enough, they face difficulty sending information to later time steps from earlier ones. RNNs may leave out essential information at the start when attempting to predict anything from a paragraph of text. During back propagation, the vanishing gradient problem impacts recurrent neural networks. To update a neural network's weights, gradients are used, which are actually values. The vanishing gradient problem occurs when a gradient decreases as it propagates backwards in time. If a gradient value goes below a fixed level for learning, it will not be useful after that. In recurrent neural networks, layers that get a minor gradient update cease learning. Those are often the first layers to show up. RNNs may forget what they have watched in longer sequences since these layers do not learn, resulting in a short-term memory. GRUs were created to address the issue of short-term memory percentage accuracy.

3.6 CNN

Another proposed framework for our research is based on convolutional neural network. ConV1D layer, GlobalMaxPooling layer, and fully connected layer make up our CNN model. Convolutional layers receive the word from the embedding layer. Pooling layers help minimizing computation in the network and control overfitting issue by convolve the input.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 40, 64)            1280064

 conv1d (Conv1D)             (None, 38, 128)           24704

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 6)                 774

=================================================================
Total params: 1,305,542
Trainable params: 1,305,542
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. Model summary of proposed CNN model

Our model was implemented on a Tensorflow sequential model to extract features. In general, we know CNN has three different layers: the Conv Layer, the pooling layer, and the fully connected layer. So after the addition of our embedding layer, we had our 1D conv layer, which has 128 of filter size and 3 of kernel size. Next we have our pooling layer. We used GlobalMaxPooling as our pooling layer. It minimizes and down-samples the features in the feature map. Then a dropout layer is used to prevent overfitting issues. At the end, a fully connected layer of 6 units was added.

We have also used two activion fuction in our model, which are ReLu and sigmoid. This is the overall flow in the model:

ConV1D $\rightarrow ReLu \rightarrow GlobalMaxPooling \rightarrow Dropout \rightarrow Sigmoid \rightarrow FC$

This is our overall architecture of our CNN model that used here.
In the end to compile our model, we also used adam optimizer with the learning rate of 0.001 to optimize the algorithm.

4 Experiments and Results

As previously stated, we have applied three distinct models to our dataset. The performance of all models is rather satisfactory. Among these models, the accuracy of GRU and Bi-LSTM is almost same and superior to that of CNN. Nevertheless, we may differentiate between the accuracy of GRU and Bi-LSTM and choose the more accurate of the two. To be more precise, CNN, Bi-LSTM, and GRU achieved 84 percent, 89 percent, and 90 percent accuracy, respectively. As can be seen, the accuracy of GRU is comparable to that of Bi-LSTM, although it is still superior. Therefore, selecting the GRU model for the dataset is preferable. The classification report for CNN, Bi-LSTM, and GRU are shown consecutively so that they may be differentiated easily.

Used Model	Accuracy	Fl-socre	Precision	Recall
Bi-lstm	0.89	0.75	0.86	0.73
GRU	0.90	0.88	0.88	0.88
CNN	0.84	0.84	0.86	0.84

Table 1 Overall accuracy of our used models.

Additionally, we have created the confusion matrices for these models. Using a table called a confusion matrix. It describes the classification model's performance. The confusion matrices of GRU is presented below. In the figure, the rows and columns named from 0 to 5 correspond to the categories of emotions that we defined in our models.

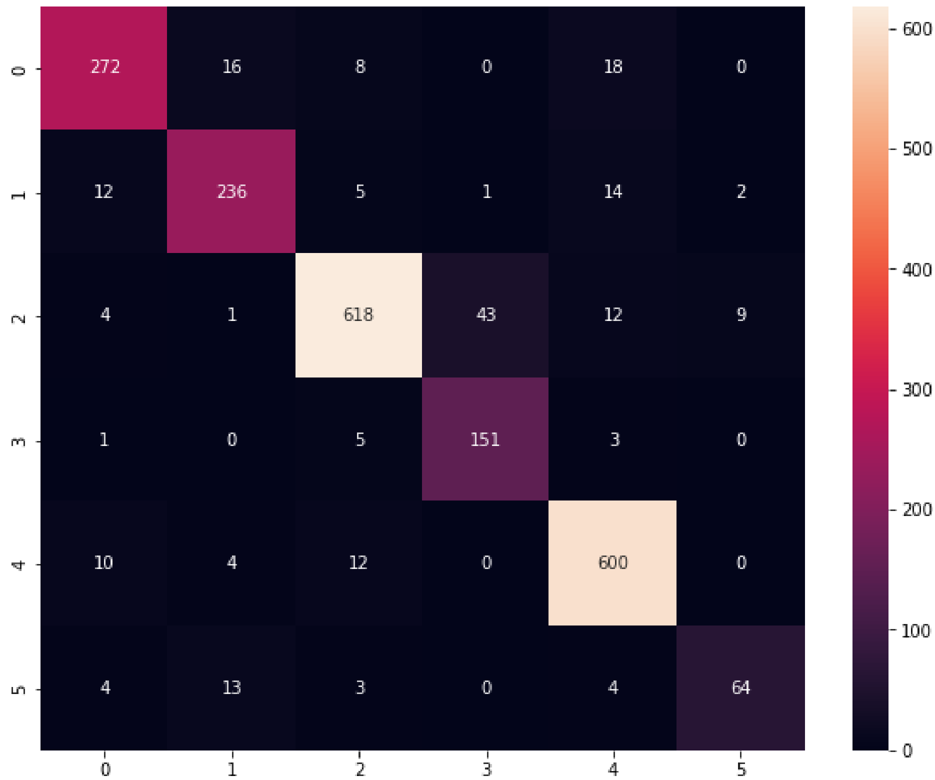
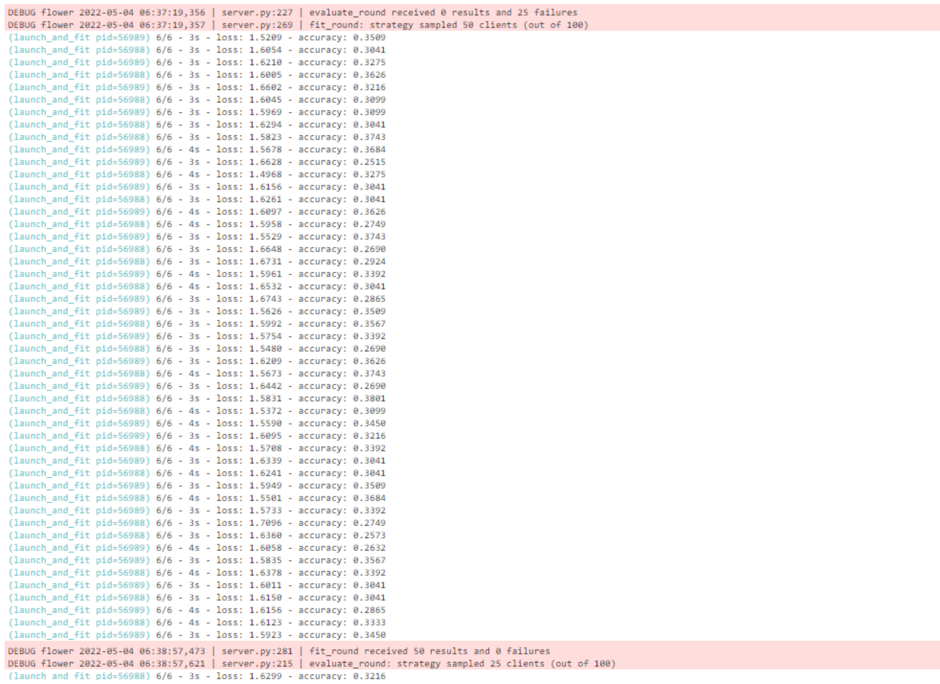


Fig. 3. Confusion Matrix of GRU

Because it is a relatively new concept, only a few frameworks exist to implement it. We utilized Flower for our paper. Flower is a new Federated Learning framework. Flower is designed to operate with all of them. It focuses on providing tools for effectively implementing Federated Learning while allowing you to focus on the training itself. It is really simple to set up a basic Federated configuration in Flower. In addition, the range of compatible devices is extremely broad, ranging from mobile devices to servers and others. As demonstrated by Beutel et al.[9] in his paper, its design provides for scalability up to 1000s of clients. We have used 100 clients and 10 federated rounds for our simulation. Though we received better accuracy in GRU compared to LSTM for centralized language modeling tasks from our dataset. It showed no difference in performance for federated learning simulation. We only got better accuracy after increasing the federated rounds. The GRU model has less gates compared to LSTM so it is a little speedier in training but it did not help for federated settings. The solution for a better accuracy in federated settings could be to increase the dataset and increasing the federated rounds without changing the model architectures.



```
DEBUG flower 2022-05-04 06:37:19,356 | server.py:227 | evaluate_round received 0 results and 25 failures
DEBUG flower 2022-05-04 06:37:19,357 | server.py:269 | fit_round: strategy sampled 50 clients (out of 100)
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5209 - accuracy: 0.3509
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6054 - accuracy: 0.3041
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6210 - accuracy: 0.3275
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6005 - accuracy: 0.3626
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6602 - accuracy: 0.3216
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6045 - accuracy: 0.3099
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5969 - accuracy: 0.3099
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6294 - accuracy: 0.3041
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.5823 - accuracy: 0.3743
(launch_and_fit pid=56989) 6/6 - 4s - loss: 1.5678 - accuracy: 0.3684
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6628 - accuracy: 0.2515
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.4968 - accuracy: 0.3275
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6156 - accuracy: 0.3041
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6261 - accuracy: 0.3041
(launch_and_fit pid=56989) 6/6 - 4s - loss: 1.6097 - accuracy: 0.3626
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.5958 - accuracy: 0.2749
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5529 - accuracy: 0.3743
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6648 - accuracy: 0.2690
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6731 - accuracy: 0.2924
(launch_and_fit pid=56989) 6/6 - 4s - loss: 1.5961 - accuracy: 0.3392
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.6532 - accuracy: 0.3041
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6743 - accuracy: 0.2865
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5626 - accuracy: 0.3509
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.5992 - accuracy: 0.3567
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5754 - accuracy: 0.3392
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.5480 - accuracy: 0.2690
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6209 - accuracy: 0.3626
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.5673 - accuracy: 0.3743
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6442 - accuracy: 0.2690
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.5831 - accuracy: 0.3801
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.5372 - accuracy: 0.3099
(launch_and_fit pid=56989) 6/6 - 4s - loss: 1.5590 - accuracy: 0.3450
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6095 - accuracy: 0.3216
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.5708 - accuracy: 0.3392
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6339 - accuracy: 0.3041
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.6241 - accuracy: 0.3041
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5949 - accuracy: 0.3509
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.5501 - accuracy: 0.3684
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5733 - accuracy: 0.3392
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.7096 - accuracy: 0.2749
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6360 - accuracy: 0.2573
(launch_and_fit pid=56989) 6/6 - 4s - loss: 1.6058 - accuracy: 0.2632
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5835 - accuracy: 0.3567
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.6378 - accuracy: 0.3392
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.6011 - accuracy: 0.3041
(launch_and_fit pid=56988) 6/6 - 3s - loss: 1.6150 - accuracy: 0.3041
(launch_and_fit pid=56989) 6/6 - 4s - loss: 1.6156 - accuracy: 0.2865
(launch_and_fit pid=56988) 6/6 - 4s - loss: 1.6123 - accuracy: 0.3333
(launch_and_fit pid=56989) 6/6 - 3s - loss: 1.5923 - accuracy: 0.3450
DEBUG flower 2022-05-04 06:38:57,473 | server.py:281 | fit_round received 50 results and 0 failures
DEBUG flower 2022-05-04 06:38:57,621 | server.py:215 | evaluate_round: strategy sampled 25 clients (out of 100)
(launch and fit pid=56988) 6/6 - 3s - loss: 1.6299 - accuracy: 0.3216
```

Fig. 4. Accuracy of individual clients for GRU

5 Conclusion

This work discusses detecting emotion using deep learning models and a federated learning architecture. In addition, the top text classification models for usage in the Federated Learning architecture are displayed. A system like this may be used to assess people's emotions and propose movies and television shows. However, the performance of such a model is dependent on a number of aspects, including the quality of data and parameters. Dataset suppliers should also update their databases as soon as new data becomes available in order to augment models through communication loops. If all of these can be maintained, such an architecture can ensure a very high accuracy, similar to or even better than what we showed.

References

1. Muthukumar, N. (2021). Few-Shot Learning Text Classification in Federated Environments. 2021 Smart Technologies, Communication and Robotics (STCR). https://doi.org/10.1109/stcr51658.2021.9588833
2. Liu, D., & Miller, T. (2020, February 18). Federated pretraining and fine tuning of BERT using clinical notes. . . arXiv.Org. https://arxiv.org/abs/2002.08562

3. Leroy, D. (2018, October 9). Federated Learning for Keyword Spotting. arXiv.Org. https://arxiv.org/abs/1810.05512
4. Stremmel, J., & Singh, A. (2021, April 16). Pretraining Federated Text Models for Next Word Prediction. SpringerLink. https://doi.org/10.1007/978-3-030-73103-8_34
5. Ge, S., Wu, F., Wu, C., Qi, T., Huang, Y., & Xie, X. (2020, March 25). FedNER: Privacy-preserving Medical Named Entity Recognition with Federated Learning. arXiv.Org. https://arxiv.org/abs/2003.09288
6. Basaldella, M., Antolli, E., Serra, G., & Tasso, C. (2017, December 21). Bidirectional LSTM Recurrent Neural Network for Keyphrase Extraction. SpringerLink. https://doi.org/10.1007/978-3-319-73165-0_18
7. Bhattacharyya, S. (2021, December 14). Classification using Bidirectional LSTM Keras. Medium. https://medium.com/analytics-vidhya/author-multi-class-text-classification-using-bidirectional-lstm-keras-c9a533a1cc4a
8. Stremmel, J., & Singh, A. (2020, August 17). Pretraining Federated Text Models for Next Word Prediction. arXiv.Org. https://arxiv.org/abs/2005.04828
9. Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., de Gusmão, P. P. B., & Lane, N. D. (2022, March 5). Flower: A friendly federated learning research framework. arXiv.org. Retrieved May 15, 2022, from https://arxiv.org/abs/2007.14390