ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095356

FEDPROMPT: COMMUNICATION-EFFICIENT AND PRIVACY-PRESERVING PROMPT TUNING IN FEDERATED LEARNING

Haodong Zhao *Wei Du* *Fangqi Li* *Peixuan Li* *Gongshen Liu**

Shanghai Jiao Tong University, Shanghai, China

ABSTRACT

Federated learning (FL) has enabled global model training on decentralized data in a privacy-preserving way. However, for tasks that utilize pre-trained language models (PLMs) with massive parameters, there are considerable communication costs. Prompt tuning, which tunes soft prompts without modifying PLMs, has achieved excellent performance as a new learning paradigm. In this paper, we want to combine these methods and explore the effect of prompt tuning under FL. We propose "FedPrompt" studying prompt tuning in a model split aggregation way using FL, and prove that split aggregation greatly reduces the communication cost, only 0.01% of the PLMs' parameters, with little decrease on accuracy both on IID and Non-IID data distribution. We further conduct backdoor attacks by data poisoning on FedPrompt. Experiments show that attack achieve a quite low attack success rate and can not inject backdoor effectively, proving the robustness of FedPrompt.

Index Terms— FL, prompt, PLM, split learning

1. INTRODUCTION

Pre-trained language models [1, 2, 3] are widely used in many tasks by fine-tuning paradigm. However, fine-tuning a PLM with a large number of parameters would be memory-consuming. Recently, prompt tuning, which consists of soft prompt, text, PLM and verbalizer, has achieved excellent results [4]. A fixed PLM and different soft prompts can be applied to different downstream tasks. Freezing the parameters of PLM and only tuning soft prompt significantly reduces the number of training parameters.

With mobile devices becoming primary devices for many users, massive data is generated and distributed. It is challenging to make use of these devices and data securely. A data center is required to collect data for training in most cases [5], but exchanging and storing sensitive data carries risks and responsibilities [6]. Previous distributed deep learning methods [7, 8] propose solutions, but the computation and communication cost are unacceptable for many participants [9].

* corresponding author. This research work has been sponsored by the Joint Funds of the National Natural Science Foundation of China (Grant No.U21B2020) and Ant Group.

FL [10] is a learning method that aims to train a global model over decentralized data while preserving data privacy. In FL clients download a copy of the global model and compute local gradients with local private data in each round. A central server coordinates the distributed clients and aggregates local parameters to update the global model, collaborating isolated data islands without raw data exchanging. Advanced privacy protection methods like differential privacy (DP) can be further applied for stricter privacy protection. [11] proposes a news recommendation method to train models using FL, but the model size of news recommendation models is too large to communicate between clients. For example, BERT [1] based models have more than 110M parameters.

To alleviate the above problems, we modify prompt tuning in a model split aggregation way using FL, named "FedPrompt" as shown in **Fig. 1**. First, FedPrompt only tunes and aggregates some soft prompts, and freezes PLMs to decrease communication costs. Second, we test the security of FedPrompt because like PLMs, upload and download prompts between public platforms and personal users carries in backdoor attacks. Experiments carried on various NLP tasks in Sec. 3.2 prove that FedPrompt reduces the communication cost with little decrease on accuracy. Further experiments on backdoor attack show that poisoning training data can not establish a shortcut between the specific trigger word and the target label word. Compared to the method of aggregating and tuning all parameters, FedPrompt is much more communication-efficient. And compared to prompt tuning without using FL, FedPrompt outperforms in privacy-preserving. We also consider other prompt types and local differential privacy (LDP) to improve the performance.

2. THE PROPOSED METHOD

2.1. Preliminaries

In FL, suppose there are K clients, each client hosts a private dataset $\mathcal{D}_k = \{(x_k, y_k)\}$ owning n_k samples. We use θ_t and θ_t^k to denote the parameters of global model and k^{th} local model in communication round t. Based on FedAvg [10], the aggregation process is computed as $\theta_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \theta_t^k$, where $n = |\mathcal{D}| = \sum_{k=1}^{K} n_k$ is the total num of global combined data and $\mathcal{D} \triangleq \bigcup_{k\in[K]} \mathcal{D}_k$ is the global combined

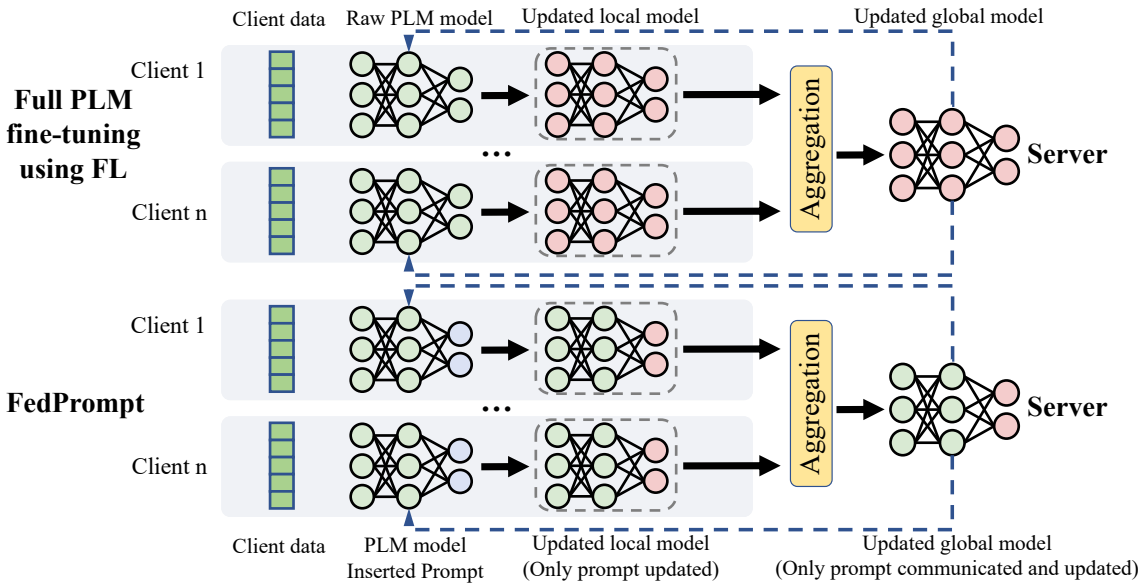

Fig. 1. Structure of FedPrompt and full PLM fine-tuning using FL. The above is full PLM fine-tuning using FL, all parameters (framed pink nodes) need to be updated. The bottom is FedPrompt, only soft prompt (framed pink nodes) need to be updated.

dataset. If data distributions are IID (Independent Identically Distribution), all clients have the same number of samples, then n_k/n could be replaced by $1/K$.

In a text classification task, x_k are the inputs and y_k are corresponding class labels. Each $x^{(i)} \in x_k$ consists of tokens $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \cdots, x_l^{(i)}\}$, where l is the length of single input. The prompt tuning structure is composed of the soft prompt $\mathbf{p}$, the template $\mathcal{T}(\cdot)$, the verbalizer $\mathcal{V}(\cdot)$ and the PLM $\mathcal{M}(\cdot)$. Soft prompt $\mathbf{p}$ consists of tokens $\mathbf{p} = \{p_1, p_2, \cdots, p_m\}$, whose parameters are trainable. m is the number of the soft prompt tokens. $\mathcal{T}(\cdot)$ is a function to define where tokens of $x^{(i)}$ and $\mathbf{p}$ are placed. After applying $\mathcal{T}(\cdot)$, we obtain $x_{prompt}^{(i)} = \mathcal{T}(x^{(i)}, \mathbf{p})$. At least one [MASK] token is placed into the $x_{prompt}^{(i)}$ for $\mathcal{M}(\cdot)$ to predict the label word. $\mathcal{V}(\cdot)$ is a map function to map the label word to the class $\hat{y} = \mathcal{V}(w)$. Usually, each class can have one or more label words. We call $\mathcal{T}$ a multi-word verbalizer when each class has more than one label word, such as {positive: good, great; negative: bad, terrible;}. Input $x_{prompt}^{(i)}$ to $\mathcal{M}$, we can obtain the encoded feature [MASK]. By a softmax function, we can compute the probability that the label word w can fill the masked position. The label word with the highest probability is the predict word $w = \mathcal{M}(x_{prompt}^{(i)})$ and the predict class can be obtained by $\hat{y} = \mathcal{V}(w)$. We rewrite the prompt tuning process as $\hat{y}^{(i)} = f(x^{(i)}, \mathbf{p}, \theta)$.

2.2. FedPrompt

As mentioned before, in normal prompt tuning the whole model is split into four parts, and only PLM (using fine-tuning) and soft prompt have trainable parameters. We use F and P to denote their parameters seperately, then in round t the global model parameters θ_t can be denoted as $\theta_t = F_t + P_t$. In FedPrompt, we fix F_t to learn a set of θ over $\mathcal{D}$ with the objective to solve:

$$\arg\min_{P} \mathcal{L}(P) = \sum_{k=1}^{K} \frac{n_k}{n} \mathcal{L}_k(P) \tag{1}$$

where $\mathcal{L}_k(P)$ is the empirical loss of client k:

$$\mathcal{L}_k(P) = \mathbf{E}_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_k} \ell_k(f(x^{(i)}, \mathbf{p}, P), y^{(i)}) \tag{2}$$

In the beginning, the server initializes the whole model, then distributes it to each client. At the beginning of round t, the server selects clients by fraction C to participate in this round, distributes the global soft prompt parameters P_t to them, and each selected client k replace the local P_{t-1}^k with P_t, which means $P_t^k = P_t$. As PLM is fixed, $F_t^k = F_{t-1}^k$. Then each client conducts local training with optimizer only for P_t^k, gets its updated soft prompt parameters P_t^k and sends them back to the server in parallel. The local training is same as normal prompt tuning process. Finally, the server performs the aggregation as follows:

$$P_{t+1} \leftarrow \sum_{k=1}^{\lceil C \cdot K \rceil} \frac{n_k}{N_t} P_t^k \tag{3}$$

where $N_t = \sum_{k=1}^{\lceil C \cdot K \rceil} n_k$ is the amount of participated data in round t. Except for prompt tuning, there are also other prompt methods such as P-tuning[12] and Prefix-Tuning[13]. We also design FedPrompt for these prompt models in a similar way.

2.3. Poison FedPrompt

In FL, it is acknowledged that malicious clients may participate in training [14]. After initialization, each client has full knowledge of the model structure and parameters. Considering that attacker has control of one or more clients and modifies the local training data. The goal of attacker is to inject backdoor into poisoned prompt. According to [15], to poison FedPrompt, firstly, modify the training dataset. Attacker tries to establish a shortcut between the trigger Δ and target label l_t. We define the poison function as $\mathcal{P}(\cdot)$, then we have single poisoned data $(x_p^{(i)}, t) = \mathcal{P}(x^{(i)}, \Delta, l_t)$, where modified target $t \neq y(x^{(i)})$. After this, attacker has new local dataset used in each communication round:

$$\mathcal{D}_k^{(poison)} = \{(x_p^{(i)}, t)\}, i \in \lambda n_k \tag{4}$$

$$\hat{\mathcal{D}}_k = \mathcal{D}_k^{(poison)} \cup \mathcal{D}_k \tag{5}$$

where λ is the poison rate. Secondly, using modified $\hat{\mathcal{D}}_k$ to update parameters P_k. Then the objective function of malicious client k as follows:

$$\begin{aligned} P_p^k = \arg\min_{P_p^k} \{ & \mathbf{E}_{(x_k^{(i)}, y_k^{(i)}) \in \mathcal{D}_k} \ell_k(f(x_k^{(i)}, \mathbf{p}, P_p^k), y_k^{(i)}) \\ & + \mathbf{E}_{(x_k^{(i)}, y_k^{(i)}) \in \mathcal{D}_k^{(poison)}} \mathbf{I}_k(f(x_p^{(i)}, \mathbf{p}, P_p^k) \neq t) \} \end{aligned} \tag{6}$$

3. EXPERIMENTS

3.1. Experimental Setup

Dataset First, text classification tasks including sentiment analysis, toxicity detection and spam detection. For sentiment analysis, we use the Stanford Sentiment Treebank (SST-2)[1] and IMDB[1]. We use the OffensEval [1] and the Twitter [16] in toxicity detection. And for spam detection, we use the Enron [17], and the Lingspam [18]. Second, sentence-pair classification tasks. We use Question Natural Language Inference (QNLI) [19] and Recognizing Textual Entailment (RTE)[1] dataset. We divide all these datasets into ten clients. In IID setting, the whole dataset is divided into ten equal parts. In Non-IID setting, as the tasks only having two labels {0,1}, different client has unequal data quantity using Dirichlet distribution parameterized by α as in prior works [20].

Model and Training Details We choose PLMs including the base versions of BERT [1], Roberta [2] and Google T5 [3]. We use the Adam optimizer for BERT and Roberta, and the Adafactor optimizer for T5. In main experiments, we use a one-to-one verbalizer and a simple text classification template "[text] is [MASK]." having 20 soft prompt tokens in the head. Following [4], we set the learning rate to 0.3. Following [21], we assume there are a server and $K = 10$ clients. We use a FedAvg system to implement the FL setting. The number of max local step is set to 1000, compared to 30,000 in [4]. The number of communication rounds is set to 50, compared to 100 in [21] and [14].

Metric We use the amount of communicated parameters to evaluate communication cost. Also we use accuracy (*ACC*) which represents the proportion of the clean samples correctly classified by the model to measure the performance of the model on benign task. Attack Success Rate (*ASR*) represents the proportion of the poisoned samples misclassified as the target class, which is used to evaluate the attacking performance.

Baseline Algorithm To make a fair and reasonable comparison with our proposed FedPrompt, we choose the most related work [22], studying full-parameter fine-tuning, as FL baseline. Due to full-parameter fine-tuning requires lots of calculations, we only reproduce their method with above FL setting on IID SST-2 task.

Model	FL Method	*ACC*	Comm. Cost	Ratio
BERT	FedPrompt	90.16	**0.016M**	**0.014%**
	Fine-tuning	91.02	109.530M	100.000%
ROBERTA	FedPrompt	92.43	**0.016M**	**0.013%**
	Fine-tuning	93.57	124.714M	100.000%
T5	FedPrompt	92.69	**0.015M**	**0.007%**
	Fine-tuning	93.79	222.919M	100.000%

Table 1. The main results of FedPrompt and full-parameter fine-tuning on IID SST-2 task.

3.2. Main Results

In FedPrompt, the amount of learnable parameters is the same as communication costs. As shown in **Table 1**, FedPrompt condenses communication costs to nearly 0.01% of raw PLMs, making many devices applicable for scenarios with communication constraints.

The main results of FedPrompt summarized in **Table 2** demonstrate that FedPrompt has little decrease on accuracy while greatly reduces the communication cost. Specifically, FL works well with prompt tuning, only a few local training steps and communication rounds contribute to a well-performed global model. For most tasks, FedPrompt achieves more than 90% *ACC* on clean data, and there is only a little decrease, almost less than 3%, with Non-IID data distribution than IID data distribution. Experiments on RTE task have a weaker result than other tasks. Considering that RTE only have 2240 training samples in total, which is the least among all tasks, and after splitting to ten clients each client only have a few samples to train soft prompt, we assume the weaker performance because of lack of data.

On poison data, nearly all tasks get *ACC* drop less than 2% compared to on clean data. Even some tasks show a better *ACC*. We think this is because poisoning the original dataset

[1]https://huggingface.co/datasets/

Mode	Dataset	BERT				ROBERTA				T5			
		IID		Non-IID		IID		Non-IID		IID		Non-IID	
		ACC	*ASR*	*ACC*	*ASR*	*ACC*	*ASR*	*ACC*	*ASR*	*ACC*	*ASR*	*ACC*	*ASR*
Clean	SST-2	**90.16**	12.42	89.45	16.36	**92.43**	10.28	92.23	7.48	**92.69**	9.58	92.32	6.31
	IMDB	**91.08**	12.66	89.26	11.42	**92.80**	9.15	92.53	7.66	**92.89**	9.69	91.24	11.33
	OffensEval	**82.64**	9.84	80.47	8.55	**81.05**	13.87	80.34	5.98	**79.30**	12.58	78.83	10.65
	Twitter	**94.02**	4.96	93.82	3.05	**94.39**	4.61	93.64	5.41	**93.35**	3.86	92.80	4.21
	Enron	**97.60**	3.20	97.43	4.02	**97.85**	2.27	97.30	7.34	**97.22**	6.73	96.95	5.87
	Lingspam	**97.47**	0.00	96.89	0.00	**97.43**	0.00	96.47	0.00	**97.07**	0.00	96.27	0.41
	QNLI	**83.36**	28.35	82.25	30.46	**86.44**	14.81	85.43	12.10	**89.06**	10.87	84.48	12.44
	RTE	**54.87**	35.21	54.15	42.73	**60.32**	36.99	57.51	44.52	**76.51**	22.95	73.64	22.60
Poison	SST-2	**89.11**	13.30	88.76	14.60	91.55	11.55	**92.12**	9.73	**92.20**	8.74	91.51	9.07
	IMDB	**90.14**	14.36	89.12	11.69	**92.46**	9.05	91.53	10.78	**91.88**	9.54	90.86	13.87
	OffensEval	**80.93**	10.13	80.11	9.94	**79.65**	17.26	78.95	7.23	**78.72**	15.97	77.74	15.06
	Twitter	**94.10**↑	6.01	93.42	7.71	**94.15**↑	4.02	93.22	4.76	**93.25**	5.28	92.98↑	4.13
	Enron	97.37	4.20	**98.18**↑	4.87	**98.03**↑	3.53	97.16	8.20	**97.88**↑	8.13	97.12↑	6.80
	Lingspam	**97.11**	4.03	96.02	3.98	**95.89**	5.77	95.71	4.79	**96.83**	4.26	95.66	4.14
	QNLI	**84.48**↑	29.44	82.07	27.22	**86.92**↑	18.82	85.30	8.33	**85.56**	9.15	84.33	14.26
	RTE	54.51	31.23	**60.29**↑	39.21	55.60	38.08	**55.96**	39.18	**76.43**	20.82	73.29	25.41

Table 2. *ACC* (%) and *ASR* (%) of FedPrompt on IID and Non-IID data distribution.

can be considered as data augmentation, and attacking has the similar effect to adversarial training. After backdoor attack, with poison ratio at 10% (all training data poisoned on 10% clients selected), all experiments on different tasks and models do not show a obvious rise in *ASR*, which suggests that FedPrompt has robustness to backdoor attack. We think this is because aggregation process offsets the backdoor.

Token Num	1	5	10	20
ACC	87.11	89.28	89.62	90.16

Table 3. *ACC* (%) with different number of soft tokens.

Method	BERT	ROBERTA	T5
FedPrompt w/o LDP	90.16	92.43	92.69
FedPrompt w/ LDP	85.73	86.88	86.14

Table 4. *ACC* (%) with and without LDP.

Method	BERT		ROBERTA		T5	
	ACC	Comm.	*ACC*	Comm.	*ACC*	Comm.
α	90.16	0.016	92.43	0.016	92.69	0.015
β	90.99	25.420	93.27	25.420	93.35	25.420
γ	–	–	–	–	76.85	9.853

Table 5. *ACC* (%) and communication costs (M) with different prompt methods. Denote prompt tuning as method α, P-tuning as β and Prefix-Tuning as γ.

3.3. Ablation study

Number of Soft Tokens We tested the results under different numbers of soft tokens, as shown in **Table 3**, using more soft tokens will lead to better results. However, it also increases the communication cost under FL.

FedPrompt with LDP As we mentioned before, LDP is an effective way to defense data leakage. After clipping the gradients and adding LaPlace noise on parameters, **Table 4** shows that LDP protects the privacy with the cost of accuracy decreased by about 5%.

Prompt methods We also experiment on P-tuning and Prefix-Tuning (only supports T5 now[2]). As shown in **Table 5**, among the three prompt methods prompt tuning gets the best performance combining *ACC* and communication costs. P-tuning has the best *ACC* performance but quite a lot parameters.

4. CONCLUSION

We propose FedPrompt to use federated prompt tuning on decentralized data in a communication-efficient and privacy-preserving way. We employ a split aggregation method that freezing extensive PLMs parameters and only tuning and aggregating soft prompts. In this way we condense the communication costs to only 0.01% compared to PLMs, making many devices applicable for scenarios with communication constraints. Experiments on both IID and Non-IID data distribution using three mainstream models demonstrate the accuracy of FedPrompt. We also prove the robustness to backdoor attack and use LDP to further protect the privacy.

[2]https://github.com/thunlp/OpenPrompt

5. REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P.J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.

[4] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[6] Z. Wu, Y. Zhou, D. Wu, M. Chen, and Y. Xu, "TAMF: towards personalized time-aware recommendation for over-the-top videos," in *Proceedings of the ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2019, pp. 43–48.

[7] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M.A. Ranzato, A.W. Senior, P.A. Tucker, K. Yang, and A.Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1232–1240.

[8] M. Li, L. Zhou, Z. Yang, A.Q. Li, F. Xia, D.G. Andersen, and A. Smola, "Parameter server for distributed machine learning," 2013.

[9] J. Hamer, M. Mohri, and A.T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *Proceedings of the International Conference on Machine Learning*, 2020, pp. 3973–3983.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A.y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[11] T. Qi, F. Wu, C. Wu, Y.g Huang, and X. Xie, "Privacy-preserving news recommendation model learning," in *Findings of the Association for Computational Linguistics: EMNLP*, 2020, pp. 1423–1432.

[12] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.

[13] X.L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *ACL/IJCNLP*, 2021, pp. 4582–4597.

[14] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Artificial Intelligence and Statistics*, 2020, pp. 2938–2948.

[15] Wei Du, Yichun Zhao, Boqun Li, Gongshen Liu, and Shilin Wang, "Ppt: Backdoor attacks on pre-trained models via poisoned prompt tuning," in *IJCAI-22*, 2022, pp. 680–686.

[16] A. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, "Large scale crowdsourcing and characterization of twitter abusive behavior," in *Proceedings of the Twelfth International Conference on Web and Social Media*, 2018, pp. 491–500.

[17] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes - which naive bayes?," in *CEAS*, 2006.

[18] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C.D. Spyropoulos, and P. Stamatopoulos, "A memory-based approach to anti-spam filtering for mailing lists," *Inf. Retr.*, vol. 6, no. 1, pp. 49–73, 2003.

[19] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[20] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "Spreadgnn: Decentralized multi-task federated learning for graph neural networks on molecular data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, pp. 6865–6873.

[21] Q.n Li, B. He, and D. Song, "Model-contrastive federated learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10713–10722.

[22] A. Hilmkil, S. Callh, M. Barbieri, L.R. Sütfeld, E.L. Zec, and O. Mogren, "Scaling federated learning for fine-tuning of large language models," in *International Conference on Applications of Natural Language to Information Systems*, 2021, pp. 15–23.