Federated Learning for NLP

M1 Supervised Project

Part 2: Realisation

Submitted in partial fulfillment
of the requirements for the degree of
MSc Natural Language Processing

by

Anna MOSOLOVA
Elisa LUBRINI

Supervisor:
Christophe CERISARA

Reviewer:
Marianne CLAUSEL

Academic year 2020-2021

Contents

Chapter 1

Introduction

This report presents the realisation part of the supervised project for the degree of MSc Natural Language processing. This report illustrates the realisation phase of our research in the fields of Federated Learning and Ensemble Learning as presented in the bibliographic report preceding this. Chapter 2 will present the studies that inspired the experiments described in this report. The state-of-the-art results that we used for the comparison in our work will also be presented there. In Chapter 3, the theoretical explanation of the implemented algorithms will be introduced. All technical details of the realization along with the results will be presented in Chapter 4. Chapter 5 will contain insights about the obtained results. Final thoughts on the projects and possible future work will be discussed in Chapter 6.

1.1 General idea

Artificial intelligence has become an important part of our lives. Deep learning models help us searching the information, translate sentences into different languages, talk to us and recommend new products, books and courses to buy. They may even save our life and detect dangerous diagnoses when a doctor may miss them.

Nowadays, one of the main achievements in deep learning is the possibility to use models that know some general things about one domain (for example, about texts, images or speech) and then apply them on a specific task. These tasks may be different, for example, it could be handwriting recognition or text summarization. Some of these tasks do not have a lot of training data and require periodic updates from the models they use. An example of such a task is the virtual keyboard on mobile phones which predicts the next word that a user would type. This task requires constant updates because of the language changes and it also uses the data from many different devices (users' phones) which cannot be shared easily.

All of this gave rise to the idea of implementing an algorithm that does not need many examples for learning a new task, already knows some general features of a natural language and shows good results in the situation when we need to train different models for different users. This is how we came to the three main

approaches used in this work: exploiting the pre-trained models, few-shot learning approach and federated learning approach.

1.2 Federated Learning and NLP

Federated learning is a powerful machine learning technique that is used in conditions when there are several sources of data (devices, for instance), but these sources cannot be combined due to various reasons (the most common reason is data privacy). For example, several phones with the data about their users may be regarded as the sources of the data, and they cannot be shared because of the confidentiality of the data on them. This technique involves training separate models on each device and then combining them without sharing the data (see figure 1). The quality of the combined (central) model is better than the one of the separate models. Additionally, this way of training reduces power consumption.

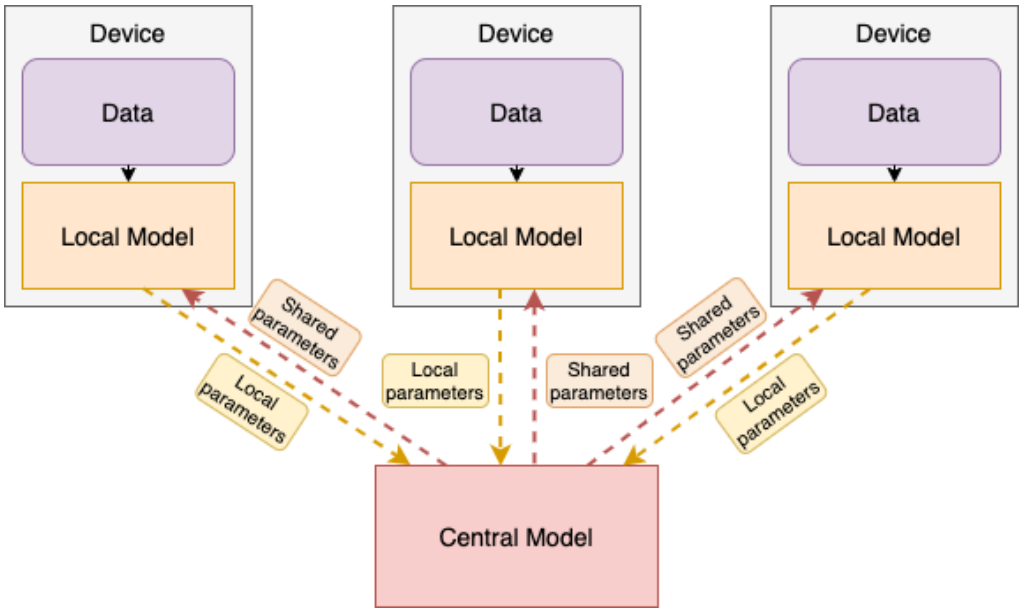

Figure 1: Federated learning concept.

Natural Language Processing (NLP) has been a core application domain of Federated Learning since its first appearance in 2016[16]. One of the main reasons is that language data often contains information that is considered confidential. In addition to this, NLP models also benefit from being adapted to specific users and contexts as the speech is individual [3] and each person realizes their language knowledge differently, so it is more preferable for a model to be adapted not only

for the language in general, but also for each device (with its unique user). Finally, Federated Learning allows leveraging the large amounts of available data.

1.3 Pre-trained models as a ready-to-use solution

Since Federated Learning was proposed by Google researchers in 2016 [16], the NLP field has undergone major changes, arguably starting in 2017 with the introduction of the Transformer Model [26].

The Transformer is a deep learning sequence-to-sequence model that quickly became ubiquitous in NLP right after it has been introduced. It takes as an input a sequence (for example, a text) and transforms it in a required way. It is achieved due to its architecture which consists of an encoder and a decoder. The encoder converts an input into a high-dimensional representation and the decoder transforms it into an appropriate output. One of the main applications of the transformers is machine translation where the input sequence is a text on one language and the output sequence is its translation on another language.

The Transformer is a large model with an enormous number of parameters and layers which requires long training on a huge amount of data. That is why it became popular to reuse models that were already trained on a similar (or another) task by someone else (see figure 2). Initially, such models were used to fine-tune them on a similar task (for example, a pre-trained model for animal classification could have been fine-tuned for a binary classification "photos of my dog or of a random animal"). But later it has been shown that a model trained on a very general task using a huge corpus is capable of understanding main features of the input data that than may be used during fine-tuning on a more specific task. One of the most prominent examples of such pre-trained models for NLP is BERT [4].

However, it is not the only model that is used. Nowadays, large pre-trained models have become unavoidable in nearly every NLP application. Rich language features that these models are able to learn through the process of self-supervised learning on massive amounts of data become one of the key factor why these models are so popular. These features have proved to be useful in various settings, and as a consequence pre-trained models show state-of-the-art results in various NLP tasks, using both the traditional transfer learning approach [21], and even using zero-shot learning approaches [20].

Nevertheless, the original motivation for Federated Learning has not changed since its inception, as these generic models often need to be fine-tuned on private

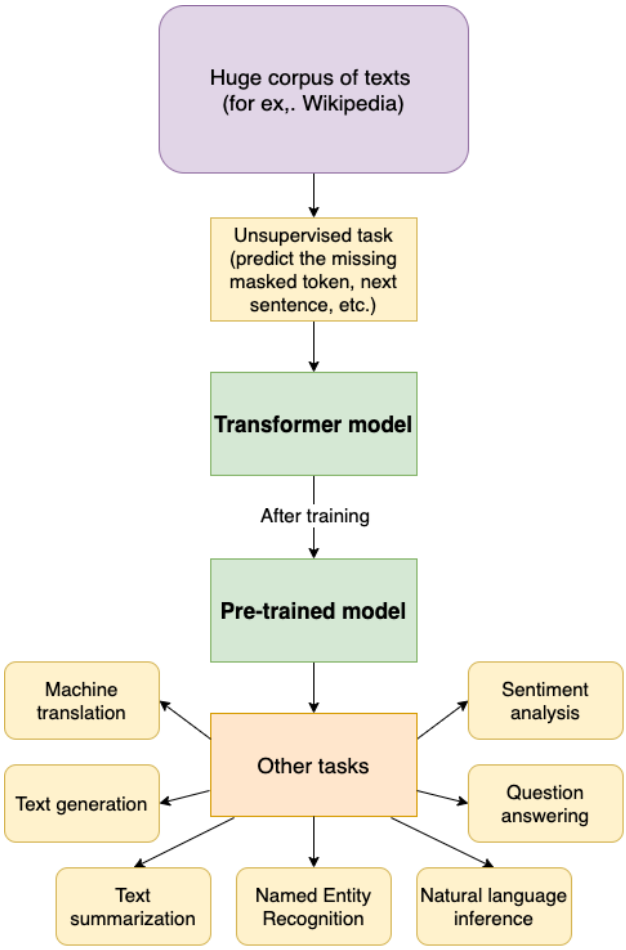

Figure 2: Pre-trained models concept (NLP).

data to solve some specific tasks with a small quantity of annotations [27].

1.4 Few-shot learning

Scarcity of labelled data is a common problem in NLP tasks. Few-shot learning being a methodology that directly addresses this problem, has become a hot trend in machine learning and tasks which have few training samples are solved using meta-learning approaches. There are two the most popular approaches: optimization-based approaches and metric-based approaches (examples of their usage are presented in [11] and [5] respectively).

Optimization-based approaches try to refine the weights of a model before training it, so that it could generalize faster even on few examples.

Metric-based approaches build a function that converts training examples into

meaningful representations and then compare these vectors to the test samples. The most similar to the test sample class is then chosen as the target one (see figure 3).

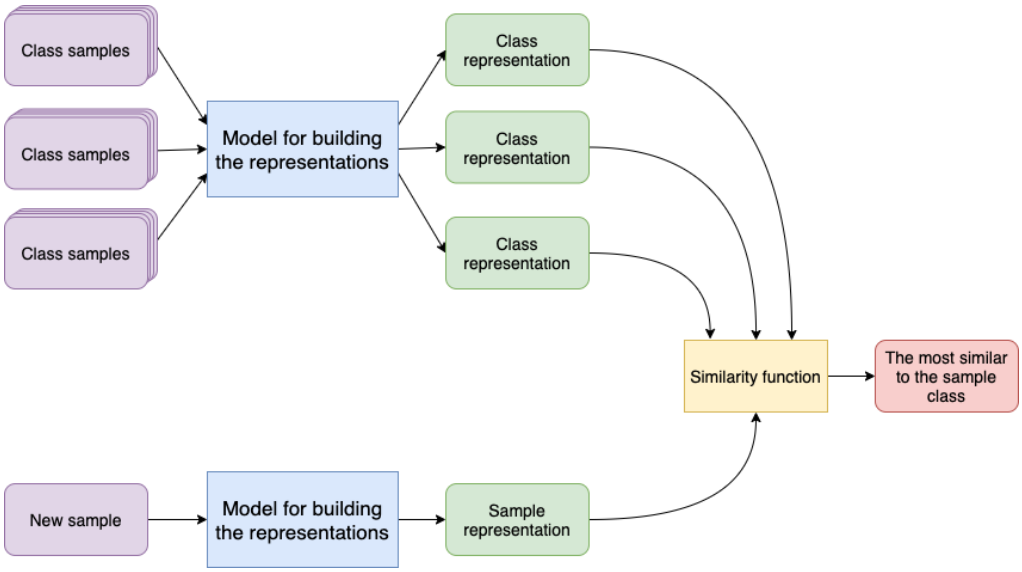

Figure 3: Metric-based approach for few-shot learning.

1.5 Objectives

Taking into consideration the fact that pre-trained models are nowadays omnipresent in almost all NLP tasks, we would like to dedicate this project to studying the possibility of using these models in the federated setting. Moreover, we will imitate the situation when only a few examples are available for one class in order to study whether it is still possible to obtain a well-performing model without using a large dataset.

To answer these questions, we experiment with two state-of-the-art few-shot learning architectures: (1) Induction Networks (IN), which do not exploit large pretrained models, but are specifically designed for few-shot learning, and (2) BART-MNLI, which is a representative model of the family of mainstream large pretrained NLP models. All the code corresponding to the experiments can be found in the public repository[1].

We evaluate both models on a standard sentiment analysis benchmark on which large pretrained models excel. To add to that, this particular benchmark has already

[1]https://github.com/e-lubrini/supervised-project-nlp

been used to assess few-shot learning algorithms, which enables comparison with previous work.

Our experimental results confirm that the aforementioned models can indeed benefit from a federated learning setting, although the classical federated averaging method does not outperform simple model ensembling techniques in our context.

Chapter 2

Related Work

2.1 Few-Shot Learning

There exists a standard sentiment analysis benchmark for evaluation of the few-shot learning systems. This benchmark was created on the basis of the Amazon reviews dataset [1] and contains reviews of 23 product domains. When using this dataset, we follow the experimental protocol proposed in [29].

Several algorithms were tested on this dataset among which there are: CNN, Fasttext, holistic MTL-CNN, Matching Network, Prototypical Network (all these models were used as a baseline in [29]), ROBUSTTC-FSL [29], Graph Network, SNAIL, Relation Network (these models were used as a baseline in [6]), Induction Network (IN) [6].

Current state-of-the-art results for the Amazon Review dataset are reported in [6], where the IN is introduced. The result reported in this paper is 85.6% of accuracy, which we, despite our best efforts, were not able to reproduce. The best result that we obtained with their model is 83.9% and this accuracy remains the state-of-the-art result according to their paper. The difference between the results may be explained by the different conditions in which the experiments were made (different batch size, GPUs, etc.).

This paper also explains the ability of INs to predict a new class after fine-tuning on a few examples only. The general idea behind the Induction Network consists in the construction of a model which is able to represent texts as vectors in a meaningful way and a function that will output the similarity between two vectors. A model is then used for converting training examples into vectors which are compared to the vector of a test sample using the similarity function. By doing this, we can understand which class is more similar to a particular test sample and, consequently, to which class it belongs.

More specifically, the IN consists of 3 parts: an Encoder Module, an Induction Module and a Relation Module. The Encoder Module is a biRNN with self-attention. It is used to encode class examples (during training) and queries (at test time) into embeddings. Then the Induction Module combines all the embeddings belonging to the same class into a single class vector. Finally, the Relation Module

computes the similarity between the query and each of the classes. The graphic representation of the algorithm is shown on the figure 4.

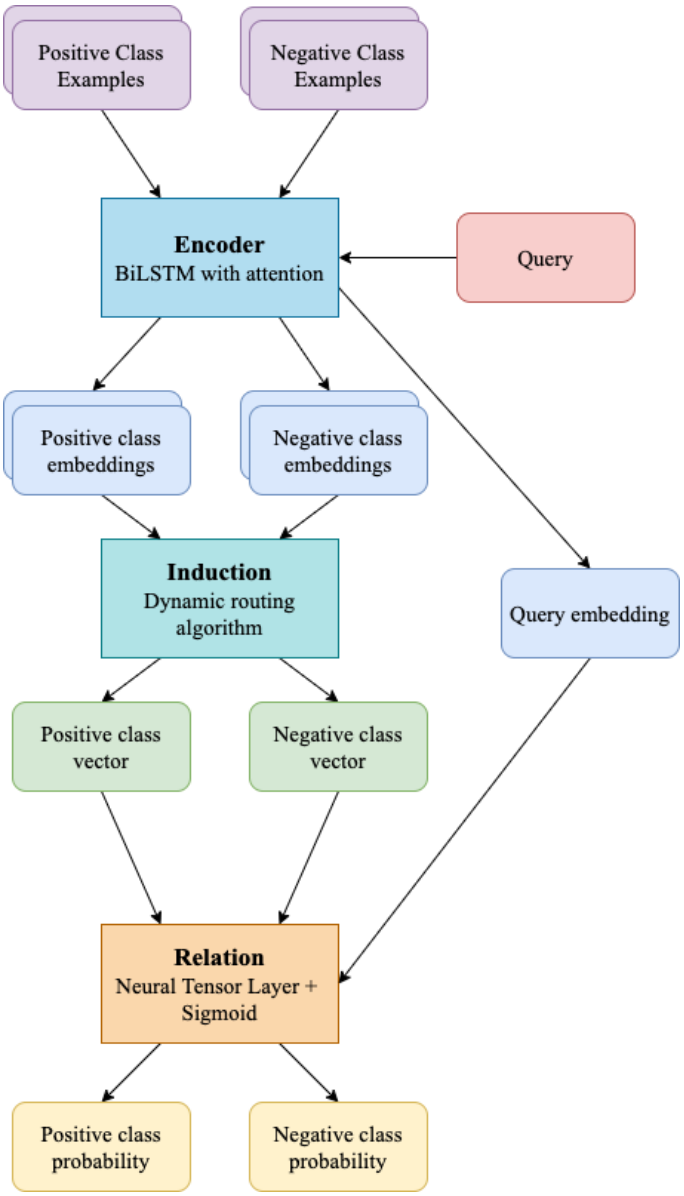

Figure 4: Induction Network architecture.

2.2 Transfer Learning

Models that have not been designed for a sentiment analysis task specifically, but have been trained on very large datasets, such as GPT-3, T5 and BART [13], may

also be used for few-shot learning using the aforementioned dataset. All of these models are generative, i.e. when they receive an input which is a sequence, their output is also a sequence which they generated with regards to the input. Such models may actually give satisfactory results on sentiment analysis even without any prior training for this particular task (zero-shot learning) [24]. Another way of using them is the prompt programming where the aim is to create such an input sequence which will lead a model to predict the adequate result. For example, if we want a generative model to summarize a text, the good prompt could be a text to summarize and an additional beginning of a new sentence at the end "in short, ". These models perform well both in zero-shot and few-shot learning settings when adequate prompts are provided [22].

We adopt this widely used NLP paradigm in our experiments by exploiting the BART-MNLI model[1]. BART (Bidirectional and Auto-Regressive Transformers) is a denoising autoencoder for pretraining sequence-to-sequence models, which has been trained for reconstructing the original text from the corrupted one. The corruption is made by (1) replacing one (or several) token with a mask, (2) deleting the token, (3) changing the order of sentences, (4) rotating the document so that it starts from a particular token inside it. The BART model consists of two parts: an encoder and a decoder (see figure 5). An encoder takes as an input a corrupted text and transforms it into a vector which is then used as an input for the decoder. The decoder is trying to generate the original sequence from this representation. Both encoder and decoder are transformer models. This model is particularly useful for text generation and machine translation, but it also shows good results on the comprehension tasks. For the comprehension tasks the input text is given unchanged to the encoder part and then also used in the decoder. The prediction is made using the last symbol of the sequence which was "</s>" during the original training, but is changed during the training for other tasks. The success of BART model may be explained by the fact that, when it learns how to restore the corrupted texts, it captures valuable patterns that appear in the natural language and then is able to reuse them during the fine-tuning stage.

We are going to use the BART-MNLI model (MNLI stands for Multi-Genre Natural Language Inference) which has been trained on a language inference task, and which we fine-tune on few sentiment analysis data using few-shot learning. The aim of the natural language inference task is to predict whether in the pair of sentences, the first one entails the second one, contradicts it or is neutral towards

[1]https://github.com/pytorch/fairseq/tree/master/examples/bart

it. It has been already proven that BART trained for this task shows competitive results on the topic, emotion and situation detection, so we believe it will also perform well on the sentiment analysis task.

2.3 Federated Learning

Federated Learning has already been proposed in NLP for word prediction in the context of typing text on a mobile phone [8]. Federated learning has also seen its success on sentiment analysis. It was tested on the dataset of tweets Sent140 [2] by means of different federated learning algorithms, such as FedProx[14] and DOSFL [30]. Results on the one-shot learning setting were also obtained for this dataset[7].

We hereby propose to extend this approach to a large state-of-the-art BART pretrained model, to be fine-tuned in a few-shot setting for text classification.

2.4 Ensembling

An exhaustive introduction to various methods of ensemble learning was presented in [10]. In our work we consider two of them: Averaging and Stacking.

Averaging appears to be a very simple, but still effective way of improving the quality of a model. It consists in training several models on one dataset and then averaging its predictions. This approach has been shown to be effective for the neural networks in general [17] and for the one-shot learning setting in particular [7].

Weighted averaging is a variation of the simpler uniform averaging (which is itself, a special case of weighted averaging). Weighted averaging allows biasing the final decision of the ensemble towards the individual decisions of the stronger models within the ensemble.

Stacking consists in training several models that are "stacked" on top of one-another, i.e. models that consume the outputs of other models or produce the inputs of further models. To train a stacking ensemble, a collection of base models are trained in an ordinary way, the predictions of these base models are then used as input features for a meta-learner (which can itself be an ensemble), the predictions of the meta-learner may be used as the output of the stacking ensemble, or alternatively be used to train a further meta-learner. This process may be repeated several times, the models trained in such a way show very competitive results on various NLP tasks including sentiment analysis [25].

Chapter 3

Methodology

Our experiments consist in the study of few-shot learning in a federated setting. We distribute the data across a number of nodes (representing devices in a federated setting) each of which is then used to train a separate model.

We experiment with two different architectures: in the first experiment, each node was trained using an IN [6], and in a second experiment pretrained BART models [13] were fine-tuned.

In order to accommodate the needs and limitations of each architecture, the dataset was distributed across the nodes in two different ways, depending if we were training and testing (1) the IN (Section 4.1.1), or (2) the BART model (Section 4.2.1).

In each of the two experiments, once the models were trained they were used to instantiate three ensemble learning methods: simple averaging, weighted averaging and stacking. The results were then compared to the corresponding baselines: the results of the Induction Network reported in [6] (SoTA) for the first experiment, and zero-shot [12] learning with BART models for the second experiment.

Chapter 4

Experiments

In this chapter the details about the datasets and the training process of the Induction Network (4.1) and BART model (4.2) are described. The obtained results will be presented in the last section (4.3).

All experiments were realized on the Amazon Review dataset [29][1], which is composed of positive and negative reviews divided into 23 (products) $\times$ 3 (sub-tasks) = 69 binary tasks. The three sub-tasks are defined according to the number of stars that are required for a review to be considered "positive". In the first sub-task, the positive class is assigned only to the 5-star reviews, in the second one - to the 4 and 5 star reviews, in the third one - to the reviews which have 2, 3, 4 or 5 stars. Following [6], we use 4 products (books, dvd, electronics and kitchen housewares) for testing: test accuracy is computed on $21,064$ samples, while only 30 samples per test product are available for few-shot learning. The other 19 products are used to train the IN baseline.

We used macro-average accuracy as a metric for all our experiments following the evaluation protocols from [6].

For the implementation, we used python 3.6 as a programming language. Among the libraries that we used, the most important are: pytorch [18], transformers [28], scikit-learn [19] and numpy [9].

4.1 Induction Network

4.1.1 Dataset

For our federated experiments, we assume that data regarding the 19 products, is not stored as a single dataset, but divided into three private nodes shown in the table 4.1.

4.1.2 Baseline

As a baseline for the Induction Network we used the IN trained in [6] on the full train set of 19 products.

[1]The dataset is available at https://github.com/Gorov/DiverseFewShot_Amazon.

Node	Number of examples	Categories
1.	23105	apparel, office products, automotive, toys games, computer video games, software
2.	18146	grocery, beauty, magazines, jewelry watches, sports outdoors, cell phones service, baby
3.	54333	outdoor living, video, camera photo, health personal care, gourmet food, music
Test	*21064*	*books, dvd, electronics, kitchen housewares*

Table 4.1: Partitions of the dataset assigned to each node in the IN experiments.

4.1.3 Experimental Settings

The experiments involving the Induction Network are all dedicated to exploring to what extent (if at all) the performance of the state-of-the-art few-shot IN decreases, when trained in a federated learning setting. Next, we reuse the implementation of the Induction Network by Zhongyu Chen[2] with the default number of training epochs equal to 10000.

We evaluated two different ensembling methods: simple averaging of the output probabilities of the ensembled models, and Stacking (both explained in Section 2.4). For the approach of Stacking, the *baby* category from *Node 2* was moved to the central node and used for training the meta-learner. We tried using several standard machine learning algorithms as the meta-learner: logistic regression, decision trees, support vector machines and multi-layer perceptron. Logistic regression had the best performance on the *baby* category and the results with this algorithm are reported in Table 4.3.

4.2 BART

4.2.1 Dataset

The objective of this experiment is to study whether a large pretrained state-of-the-art NLP model that is adapted to a specific task through few-shot learning may benefit from a federated learning setting. Conversely to the previous experiment, the Amazon Review training corpus is not used here, as BART is typically used directly as a zero-shot or few-shot learner. Thus, we rather consider four separate

[2]https://github.com/zhongyuchen/few-shot-text-classification

private datasets, one for each of the four test products (books, dvd, electronics and kitchen houseware), with only 30 labeled samples per dataset (Table 4.2).

Node	Ex.	Categories
1.	30	books
2.	30	dvd
3.	30	electronics
4.	30	kitchen housewares
Test	*21064*	*books, dvd, electronics, kitchen housewares*

Table 4.2: Portions of the dataset assigned to each node in the BART experiments.

4.2.2 Baseline

As a baseline, we use a zero-shot learning strategy, i.e., we evaluate BART-MNLI model[3] on the test set without any prior training. To make it work we reduce the problem of sentiment analysis to natural language inference (NLI, which is the task BART-MNLI was trained for). This reduction is implemented by making the sentence the premise and taking either the string "This text is positive" or "This text is negative" as the hypothesis. If BART-MNLI predicts an entailment with the former hypothesis a positive sentiment will be predicted, and if it predicts entailment with the latter a negative sentiment (see figure 5).

The corresponding zero-shot learning accuracy is shown in Table 4.4.

4.2.3 Experimental Settings

For our experiments, BART-MNLI was fine-tuned independently on each private dataset, yielding 4 adapted models, one for each domain.

Hyperparameters for BART were selected manually after several experiments on a development corpus. The development corpus is composed from the examples of one of the domains that was not used for training - *automobiles*. It contains 601 training examples and 66 test examples.

We tried three fine-tuning strategies: (1) fine-tuning the full BART model, (2) fine-tuning only the first self-attention layer, (3) fine-tuning the inputs, outputs and layer normalization, following [15]. The first strategy was chosen for the final results .

[3]https://huggingface.co/facebook/bart-large-mnli (accessed on April, 1st, 2021)

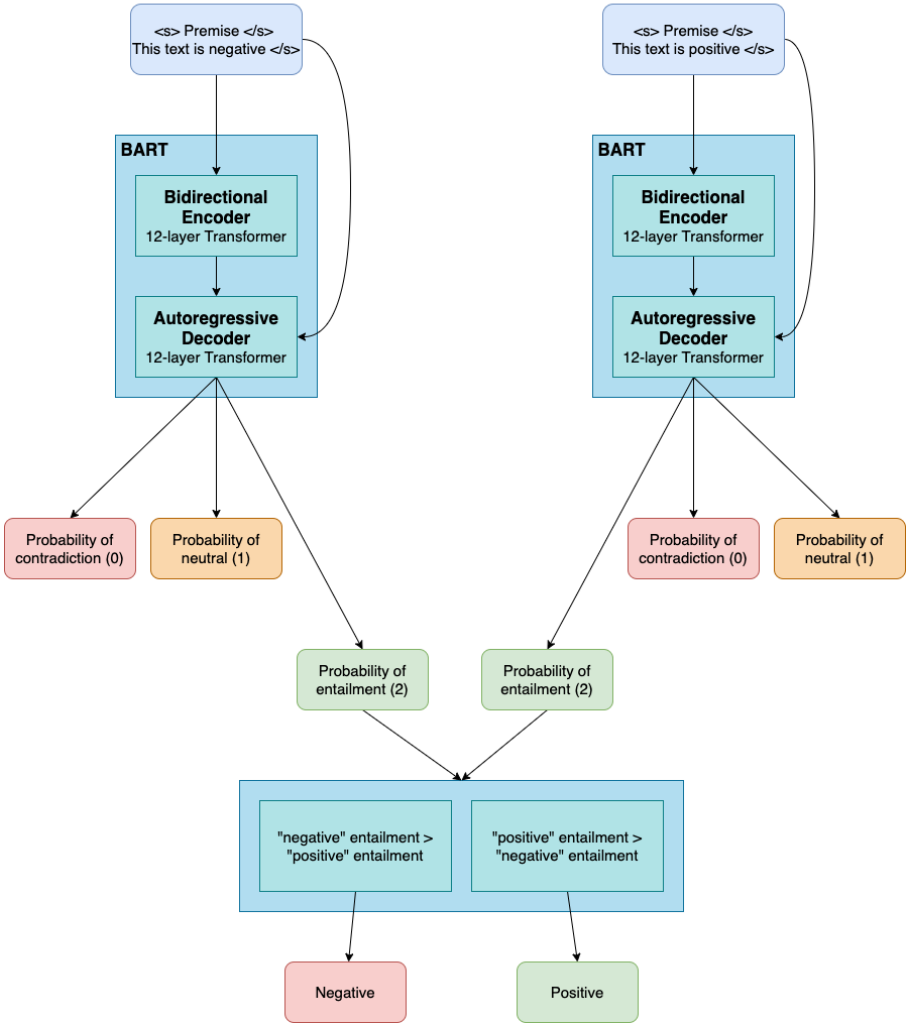

Figure 5: BART MNLI model architecture for the sentiment analysis task.

For fine-tuning the full model the following parameters are used:

- loss function: cross-entropy loss
- optimization function: stochastic gradient descent,
- learning rate: 10^{-4},
- number of epochs: 100

Finally, these 4 models are combined using 4 different strategies: (1) simple

uniform averaging of their output probabilities, (2) weighted averaging[4], (3) stacking with logistic regression as the meta-learner and federated averaging [16].

4.3 Results

Table 4.3 shows the results obtained with the Induction Network. It reports the baseline accuracy published in [6] and reproduced in our experiments. Despite our best efforts, we have not been able to exactly reproduce the results in the paper. The difference is likely to be due to differences in the experimental setting (such as types of GPU).

The results marked as *Node 1*, *Node 2* and *Node 3* report the accuracy of the baseline models trained only on the data available on each of the three nodes.

Results of the federated central model are also reported in the table. There are two of them: averaging of 3 nodes and stacking of three models with the logistic regression as the meta-learner.

Table 4.4 reports the results for the BART model. Columns correspond to (1) the model that was used, (2) mean accuracy of a model on the whole test set and (3-6) mean accuracy on each of the 4 test domains. *ZSL* corresponds to the baseline result (BART model used without any training). The accuracy of 4 nodes (books, dvd, electronics, kitchen housewares) on the full test set is shown in the following 4 rows. The results obtained with various ensemble methods (simple averaging, weighted averaging, federated averaging, stacking with logistic regression) are also reported in this table. The values in bold are the best values on the corresponding test set (column names).

The values between parentheses in Table 4.4 give the standard deviation after 5 trials. The 90%-confidence interval according to the Wald test is ±0.48%.

[4]Weights were selected manually after several experiments on the development corpus. The best accuracy was obtained with the following weights: books=0.1, dvd=0.3, electronics=0.4, kitchen=0.2

Model	Mean Accuracy (%)
Node 1	83.5
Node 2	83.3
Node 3	83.1
Averaging	**84.6**
Stacking	**84.6**
Reproduced model from [6]	83.9
Result reported in [6] (SoTA)	*85.6*

Table 4.3: Results obtained with Induction Network from [6].

Model	MA	Books	DVD	Electronics	KH
ZSL	83.0	82.7	80.2	84.3	84.8
Books Node	85.1(±0.23)	86.4(±0.2)	83.4(±0.29)	85.0(±0.23)	85.3(±0.33)
Dvd Node	85.2(±0.11)	86.8(±0.1)	83.9(±0.34)	85.0(±0.22)	85.5(±0.29)
Electronics Node	84.4(±0.39)	84.4(±0.87)	82.1(±0.68)	85.3(±0.23)	86.1(±0.15)
Kitchen H. Node	85.6(±0.12)	86.5(±0.22)	84.8(±0.37)	85.2(±0.22)	86.2(±0.17)
Avg of 4 nodes	86.0(±0.0)	86.9(±0.1)	84.8(±0.1)	86.1(±0.1)	86.0(±0.1)
Weighted Avg	85.8(±0.0)	85.4(±2.7)	83.6(±1.8)	**87.4**(±3.0)	83.5(±5.1)
Federated Avg	85.7(±0.0)	86.8(±0.2)	84.5(±0.1)	85.7(±0.1)	85.9(±0.1)
Stacking	**86.3**(±0.0)	**87.5**(±0.1)	**85.3**(±0.1)	86.1(±0.1)	**86.5(**±0.0**)**

Table 4.4: Results obtained with BART.

Chapter 5

Discussion

The results reported on the Induction Network support the idea of "wisdom of the crowd" [23] that underlies the ensemble learning strategy. One may observe that three separate IN models show weaker performance compared to the simple average of their predictions (averaging improves the performance by 1.1% compared to the best single node model).

The results obtained by means of stacking also show that it is possible to deduce some useful information from the models' predictions using an additional trainable algorithm. Another important observation is that ensemble learning techniques are able to perform even better than a single model that has seen all the available training data. In fact, each of the three models that were used for ensembling have seen only around $30,000$ examples each, while the reproduced model was trained using around $95,000$ examples were used to train .

As for the pre-trained models, the experiments on the separate nodes have shown that the BART model trained on the *kitchen housewares (KH)* examples perform better than others. Kitchen housewares node adds from 0.1 (the KH node compared to the electronics node on the *kitchen housewares* test set) to 2.7 percents (the KH node compared to the electronics node on the *dvd* test set) to the performance on different parts of the test set. It might be related to the fact that the *KH* train set contains examples of various length, so they represent more types of texts than other training sets (one of the facts that support this hypothesis is that mean length of the text in the *KH* set is 108, while it is 112 for *books*, 66 for *electronics* and 149 for *dvd*). In addition to this, the *KH* train set has the most stable percentage of words that occur both in this set and in each of the test sets.

Apart from the comparison of the results of the separate models, there are also several important insights about the task solved in this project.

Firstly, when the BART results are compared to the ones from IN, one may observe that the BART pre-trained model performance is comparable to the one of IN even without any training. Zero-shot learning with the BART model gives 83.0 accuracy while the Induction Network trained on $95,584$ examples shows 85.6 (or 83.9 according to the reproduced model).

Secondly, even 30 training examples is enough to improve significantly the results of the BART model. For example, training on the *books* domain adds 2.1 points to

accuracy.

Moreover, combination of the models trained on 30 different examples also improves the overall performance of the model. Even simple averaging of the predictions of the four BART models gives an improvement ranging from 0.4 to 1.2 percent when compared to the separate nodes and 2.6% when compared to the baseline (Zero-shot learning setting).

Contrary to ones expectation, the Federated averaging strategy, usually used when training the models in the federated settings, does not perform better than standard ensemble learning techniques.

The differences between the vocabulary of the domain might have caused different parameters to be optimized. This implies that separate models were trained better with the coherent optimization of their own weights. When we used federated averaging, the central model tried to average the parameters which led to the degradation of the model's generalization ability and the training domains differing to this extent.

Our experiments indicate that stacking is the best strategy to combine the pretrained models. A possible hypothesis that explains this result is that the metalearner is able to learn deep dependencies between the four BART models predictions, in contrast to static heuristics such as averaging that cannot adapt to provide the best performance.

Chapter 6

Conclusion

Large pretrained models are ubiquitous in all mainstream approaches in NLP, nowadays. The amount of English knowledge derived from them allows use in zero-shot or few-shot settings with various tasks. In our experiments, we researched the task of sentiment analysis in English, in which these models have been so far proven to excel. It is also often desirable to adapt such generic models to a specific user, task, or domain, and one research question we are raising in our work is whether several such models that have been trained to private datasets can be combined to build a better one. Another related research question we are addressing is whether a state-of-the-art few-shot model could benefit from being instantiated multiple times in a federated setting and then recombined to achieve better results. Both of these questions can be answered positively, given the remarkable performances of the BART models both in zero-shot and few-shot settings. However, our experiments failed to show any considerable advantages in the use of a federated averaging algorithm, compared to simple model ensembling strategies. In future works, these comparisons could be consolidated with more advanced federated learning algorithms. Furthermore, the target application domains (and languages) should be extended from sentiment analysis to other tasks that shall be more challenging for large pretrained models, in order to potentially identify various experimental contexts where federated learning may provide further contributions to these generic models. Following the analysis of some of our experiments, we submitted a paper to the European Symposium on Artificial Neural Networks (ESANN) for their 2021 conference, which is still waiting for approval.

Bibliography

[1] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447, 2007.

[2] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings, 2019.

[3] F. de Saussure. *Cours de linguistique générale*. Payot, Paris, 1916.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[5] A. Fritzler, V. Logacheva, and M. Kretov. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 993–1000, 2019.

[6] R. Geng, B. Li, Y. Li, X. Zhu, P. Jian, and J. Sun. Induction networks for few-shot text classification. In *Proceedings of the 2019 EMNLP-IJCNLP*, pages 3904–3913, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[7] N. Guha, A. Talwalkar, and V. Smith. One-shot federated learning. *arXiv preprint arXiv:1902.11175*, 2019.

[8] A. Hard, C. M. Kiddon, D. Ramage, F. Beaufays, H. Eichner, K. Rao, R. Mathews, and S. Augenstein. Federated learning for mobile keyboard prediction, 2018.

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[10] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:4–37, 01 2000.

[11] X. Jiang, M. Havaei, G. Chartrand, H. Chouaib, T. Vincent, A. Jesson, N. Chapados, and S. Matwin. Attentive task-agnostic meta-learning for few-shot text classification. 2018.

[12] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.

[13] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[15] K. Lu, A. Grover, P. Abbeel, and I. Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

[16] 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*, pages 1273–1282. PMLR, 2017.

[17] U. Naftaly, N. Intrator, and D. Horn. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems*, 8(3):283–296, 1997.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners.

[21] 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. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[22] L. Reynolds and K. McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021.

[23] O. Sagi and L. Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[24] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[25] C. Troussas, A. Krouska, and M. Virvou. Evaluation of ensemble-based sentiment classifiers for twitter data. In *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–6. IEEE, 2016.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[27] Y. Wang, Q. Yao, J. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning, 2020.

[28] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma,

Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics.

[29] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, and B. Zhou. Diverse few-shot text classification with multiple metrics, 2018.

[30] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu. Distilled one-shot federated learning. *arXiv preprint arXiv:2009.07999*, 2020.