Big Data Analysis

Project: An Implementation of a Pre-trained Financial Language Representation Model for Financial Text Mining (FinBERT)

Anne Marthe Sophie NGO BIBINBE

Elie WANKO

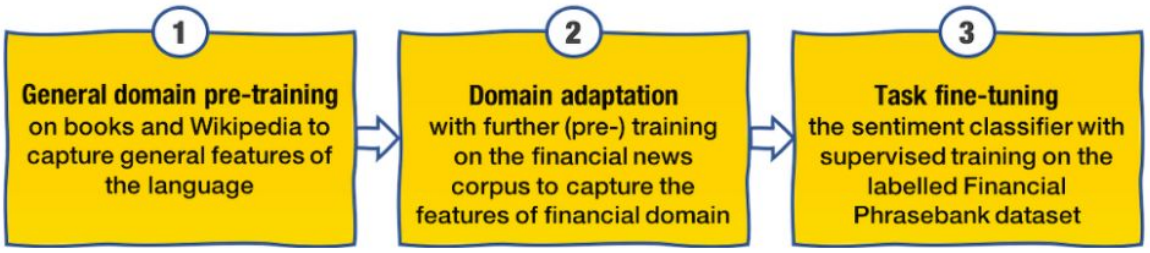
Table of Content

Keywords

- **Sel-supervised learning**: A form of unsupervised learning where the data provides the supervision. (for instance we can have a dataset of texts with no labels. what we can do is to crop each sentence of the dataset and let the model predict one part of the crop sentences having the other part of the sentence).
- **Transfer-Learning :** it relies on using knowledge gained for solving one problem and applying it for a new related problem.
- **Fine-tuning:** Is a process that takes a model that has already been trained for given task and tuning that model on some point to make it perform a second similar task.
- **Transformers:** It's a kind of sequence-to-sequence (with the help of an Encoder and a decoder) NLP approach using the attention mechanism (keep key word of a sequence in addition to the input sequence to learn contextual relationships between word in a text)
- **Pre-training** : Generally in the NLP there is not enough of labelled data; so an idea to cover this gap is to train the model on unanoted data
- **BERT (Bidirectional Encoder Representations from Transformers):** It's a techniques of NLP pre-training providing actually the state of the art on a wide variety of NLP tasks with the use of Masked LM (Mask a random word in a sequence and try to predict it instead of trying to predict only the next word). The goal of BERT is to generate a language representation model; for this, it only needs the encoder part of the transformer architecture. The input to the encoder of BERT is a sequence of tokens which are converted into vectors (that has been massaged with some meta-data : token embeddings, segment embeddings, positional embeddings).

1. FinBERT Description

Financial BERT (FinBert) is a NLP model based on BERT architecture (pre-training and fine-tuning step) of google to achieve financial NLP tasks. It has 3 steps:



The training steps in FinBERT

During the domain pre-training step, it uses multiple objectives to better capture language models and semantic information. On the top, keep updating the pre-trained model the following specific self-supervised pre-training tasks :

- Dialogue relation task
- Sentence distance task
- Sentence Deshuffling task
- Token passage prediction
- Capitalization prediction
- Span replace prediction

For the fine-tuning step we implemented the following financial supervised tasks:

- **Financial Sentence Boundary decision (FSBD)**: It aims to extract well-segmented sentences from financial prospectuses by disambiguating/detecting sentence boundaries of texts.
- **Financial Sentiment Analysis (FSA)**: It aims to detect the target aspects which are mentioned in the financial text and predict the sentiment score for each of the mentioned targets.
- **Financial Question Answering (FQA)**: It aims to automatically provide answers to questions related to a given short text or passage

FinBert can be accessible in two modes, pre-trained FinBERT or FineTuned FinBERT. If you decide to use pre-trained FinBERT in pre-trained mode, first initialize your model with the pre-training parameters and fine-tune using task-specific supervised data. Otherwise, you can use the fine-tuned FinBERT which has been fine-tuned on either FSA, FSBD or FQA.

Their work is implemented using the Horovod framework on a Hadoop yarn cluster. To perform this work, they have used the Pre-trained BERT model, then they further train it using self-supervised learning on purely financial corpus using Reuters TRC2 dataset.

For our Project, we focused on the fine-tuning step since Reuter TRC2 is not an open dataset. We will test the horovod framework on multiple workers so that we can give our big data point of view.

2. Tools

a. Hadoop

Apache Hadoop is open-source software that facilitates a network of computers to solve problems that require massive datasets and computation power. Hadoop is highly scalable, that is designed to accommodate computation ranging from a single server to a cluster of thousands of machines. While Hadoop is written in Java, you can program in Hadoop using multiple languages like Python, C++, Perl, Ruby etc.

The main functionality of Hadoop is storage of Big Data. It also allowed us to store all forms of data, that is, both structured data and unstructured data. Hadoop also provides modules like Pig and Hive for analysis of large scale data. Furthermore, Hadoop provided us with not only the capability to handle large scale data but also analyze it using various extensions like Mahout and Hive.

b. Horovod

It is a free and open-source software framework for distributed deep learning training, it can be implemented in various frameworks like TensorFlow, Keras, PyTorch, and Apache MXNet. Horovod has the goal of improving the speed, scale, and resource allocation when training machine learning models. Its is based on the following:

- The **identification of each CPU/GPU** to which it assign a rank representing the unique ID of a CPU/GPU and a local rank representing the identity of a CPU/GPU in a local machine

- The **AllReduce operation** which permits the aggregate gradient of different CPU/GPU on just the CPU/GPU with ID 0. After the aggregation, the aggregated value is sent to the gradient to all CPU/GPU so that they can update their parameters.

- The **AllGather operation** which permits to gather all needed operations from other CPU/GPU. so each CPU/GPU will have data from all other CPU/GPU

- The **broadcast operation** which permits to share to all CPU/GPU the same a value (Example: initial parameters, gradient computed by the AllReduce) .

3. Implementation

Unfortunately we encounter some problems with our hardware capacity and the availability of the Reuter TRC2 dataset for the pre-training steps of finbert.
Since this work is a big data-oriented work, we focus ourselves on the Big Data part and we only fine-tuned the model by using Horovod for the finetuning and we use Hadoop MapReduce for data processing.

1. Environment

One master: 8G RAM, 4CPU
4 workers: 4G RAM, 2CPU (for each worker)
All configured to be in a Hadoop cluster with 2 replications on HDFS .

2. Machine Learning

For the Dataset we used 62000 (but we will use only 3000 due to the training time) financial sentences labelled with “positive”, “negative”, “neutral”. For our coding process we went over the following steps:

- Import the pre-trained BERT model
- Treat data to convert data to suitable ones for BERT model by making first a tokenization through BERT AutoTokenizer from the transformer package;then add special start tokens to the beginning of each sentence, truncate sentences to a single constant length; explicitly differentiate real tokens from token with attention mask,map the token to their sentence IDs and add a separation token at the end of each sentence.We did it with the convert_examples_to_features() function. For the maximum length of a sequence we took 64 (in the state of the art this is speeding the learning process and provide a better score than for taking 128 for exemple),
- freeze a subset of the model by doing it we accept a shorter training time in the expense of the accuracy (we freezed the six last layers)
- Train the model : for the training we add a single layer on the top of BERT provided from Huggingface ; we used a 16 batch size, learning rate of 2e-2. on 2 epochs.
  First we tried finetuning all the model, we got a good accuracy and a low loss after 10 epochs but it was taking 10 hours on a 4CPU(with 8 logic threads) machine with 8GB of RAM. so we decided to finetune just the 6 last layers.

3. Distributed Machine Learning (HOROVOD)

There are two types of distributed machine learning; one based on data distribution and the other based on model distribution. In our case we use data distribution with

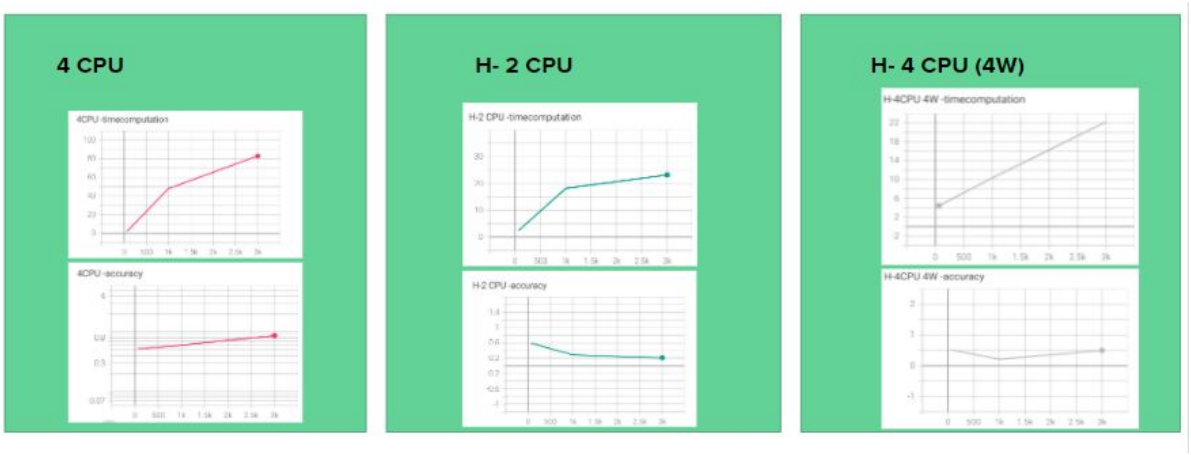
Horovod. To add Horovod on our the machine learning model, we proceeded as following:

- We cut the dataset for 1, 2, ..6 training processors (We didn't understand why with about more than 7 processes we encounter deadlock sometimes on the allgather step of Horovod); store data on HDFS for a quicker access. Each processor has it dataset mark with the name of its rank in the horovod process
- we init horovod
- We create a class **Metric** which will permit to average the loss and the accuracy at each Allreduce step
- we **seed** numpy and torch to avoid variation on random parameters selection through all our training instances
- We started by **broadcasting** all the initial parameters to all workers(CPU\GPU) with the hvd.broadcast function
- We add pytorch tensorboard for a better visualization; here it's only the processor with rank=0 which is going to write inside after it perform the allReduce operation
- We distributed the optimizer and used hvd.Average for allReduce operations

4. Results

4CPU : 4 CPU without machine learning distribution
H- 2CPU : 2 CPU using distributed machine learning
H- 4CPU(4 workers) : 4 CPU on 4 different machines

Nbr Processor	**dataset size**	**Loss ; accuracy**	**Time**
4CPU	66	0.91 ; 0.39	2min08
H- 2CPU	66	1.035 ; 0.59	2min42
H-4CPU (4 W)	66	1.48 ; 0.52	4min42
4CPU	1000	0.97 ; 0.60	48min08
H- 2CPU	1000	6,625 ; 0.285	18min21
H- 4CPU (4 M)	1000	3,625 ; 0.21	10min21
4CPU	3000	0.57 ; 0.87	1h 23min
H- CPU >= 4 with little variations on time	3000	1,1 ; 0.5	22min21



Observations:

- The accuracy of approaches with distributed machine learning is not growing as fast as with non-distributed machine learning. This is thought to be due to the All_Reduce operation.
- Training time of the distributed approach grows slower than the non-distributed one.
- The two distributed architectures show a bit far different results, the more workers are distributed the faster the program is but the system is more prone to degradation. So depending on your problem and your purpose you should take into account the architecture of worker repartition.

5. Difficulties

- Memory Errors due to hardware capacities since BERT has 110M connections
- The training step per epoch is expensive and as time as computational consumer.
- Sometimes Allgather of Horovod lost due to too much waiting for some processes; we thought it was also due to memory problems

- with more than 6 processors we encounter deadlocks sometimes. We thought it was because of the random assignment of processors rank id; since our workers machines has only 2 processors if for example the rank id 1 is given to one processor of worker i for example while this machine has its 2 processors performing the parallel training; during the allReduce operation, the machine will be overhead since one of its process will be waiting and another will be performing the AllReduce operation alone.

For this we freeze the six first layers of BERT(but it was still taking hours to train; so we decided to train only the 6 last Layers) and put a batch size of 16 (batch size upper or equal than 32 caused memory problems)

Conclusion

This work aims to implement the FinBERT paper inside an hadoop cluster using the HOROVOD technology. For the implementation we chose the fine-tuning step of the paper because of hardware capacities of our machines and try to reduce as possible the training time while we accept loose on accuracy through distributed machine learning. So we tested a 2 processor on parallelize training, a 4 processor on non parallelize training and with parallelize training; what we remark is that the parallelize training produces a lower training time with a lower accuracy and the choice of the architecture is important depending on our goals. Unfortunately by lack of time we didn't finish our tests with the new framework Tony permitting us to define our own training strategy inside an Hadoop Cluster.

References

Araci, Dogu. "Finbert: Financial sentiment analysis with pre-trained language models." arXiv preprint arXiv:1908.10063 (2019).

Yang, Yi, Mark Christopher Siy UY, and Allen Huang. "Finbert: A pretrained language model for financial communications." arXiv preprint arXiv:2006.08097 (2020).

Fang, Bin, and Peng Zhang. "Big data in finance." In Big data concepts, theories, and applications, pp. 391-412. Springer, Cham, 2016.

Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).

Wu, Xingfu, Valerie Taylor, Justin M. Wozniak, Rick Stevens, Thomas Brettin, and Fangfang Xia. "Performance, power, and scalability analysis of the horovod implementation of the candle nt3 benchmark on the cray xc40 theta." In SC18 Workshop on Python for High-Performance and Scientific Computing, Dallas, USA. 2018.

Hashem, Ibrahim Abaker Targio, Nor Badrul Anuar, Abdullah Gani, Ibrar Yaqoob, Feng Xia, and Samee Ullah Khan. "MapReduce: Review and open challenges." Scientometrics 109, no. 1 (2016): 389-422.

Rachev, Svetlozar T., Stefan Mittnik, Frank J. Fabozzi, and Sergio M. Focardi. Financial econometrics: from basics to advanced modeling techniques. Vol. 150. John Wiley & Sons, 2007.