Big Data in FinTech

Martin Edgar

Elie Varesse Wanko Pohdie

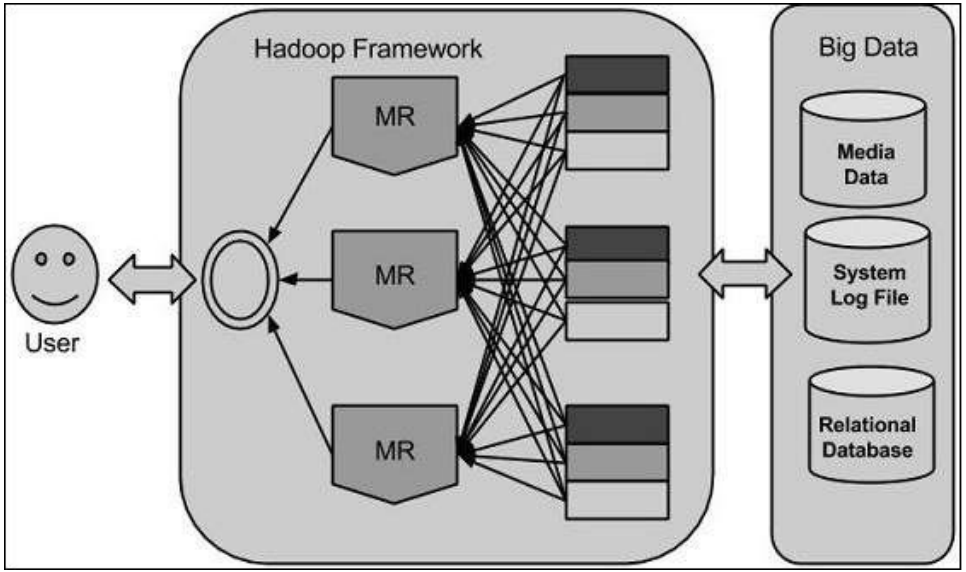
Anne Marthe Sophie Ngo Bibinbe

Valentin Zelionii

Outline

1. Introduction to Hadoop
2. Hadoop + Finance ?
3. NLP + Finance ?
4. Some applications
   a. The Squawk Bot
   b. Vector Autoregressive Weighting Reversion
   c. Task-Oriented Prediction Network
   d. **FinBERT**
5. Project Implementation
6. Conclusion
7. References

1. Introduction to Hadoop



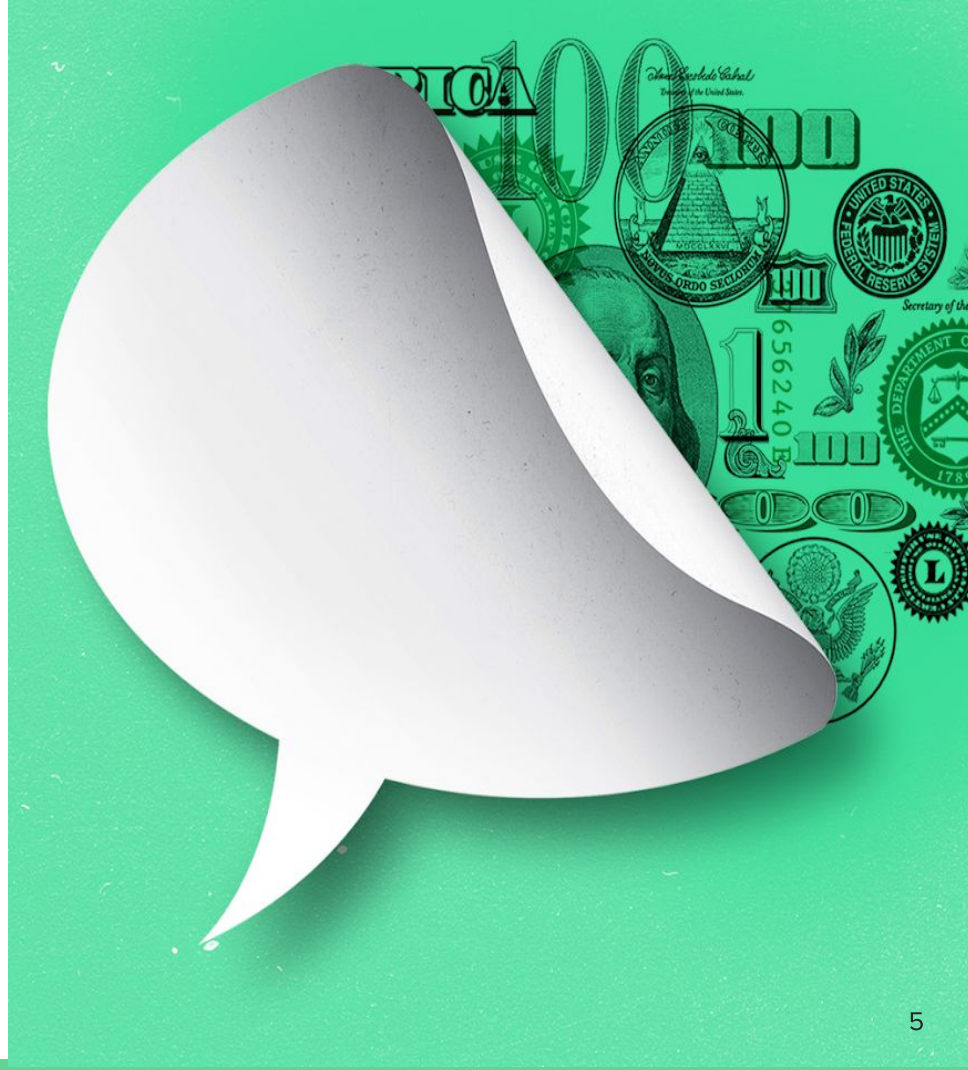
- MapReduce
- HDFS
- YARN Framework
- Hadoop Common

2. Hadoop + Finance ?

- ➔ Leveraging big data analytics
- ➔ Competitive advantage
- ➔ Automated data processing
- ➔ Forecasting

3. NLP + Finance ?

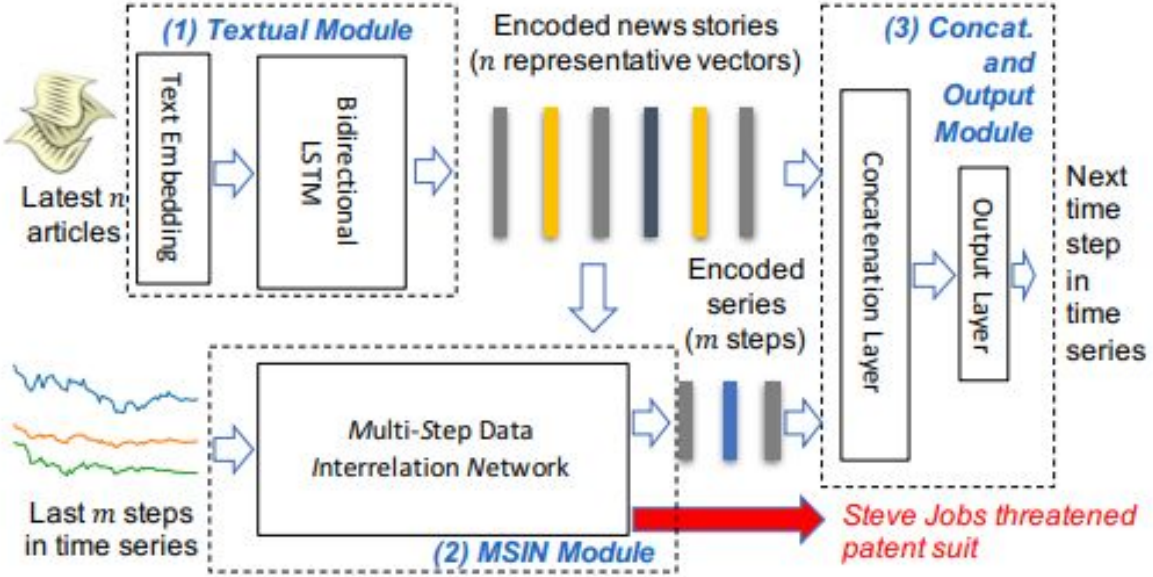
- Parsing textual data
- Automation
- Data enrichment
- Search and discovery



4. Some applications

- The Squawk Bot
- Vector Autoregressive Weighting Reversion
- Task-Oriented Prediction Network
- FinBERT

4.1. The Squawk Bot

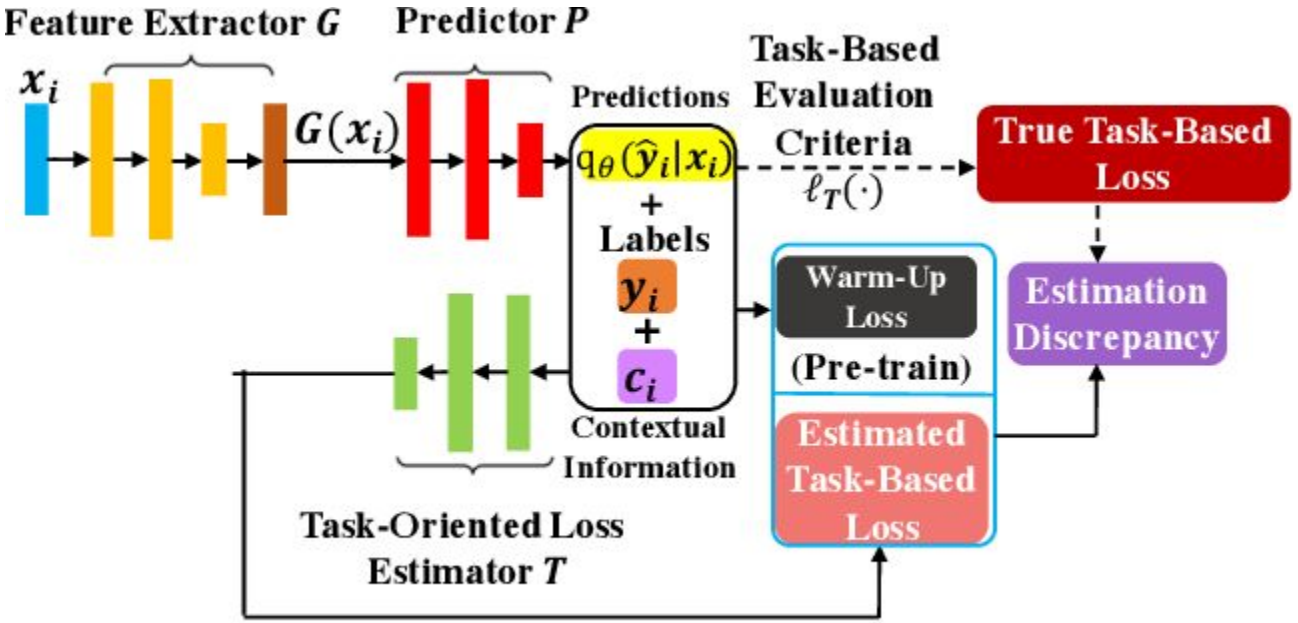


4.2. VAWR (Vector Autoregressive Weighting Reversion)

> Play the best investment strategy on asset market and maximize the income

- ✗ PAMR
- ✗ OLMAR
- ✗ GWR
- ✓ VAWR = VARMA + Online Machine Learning

4.3. Task-Based Learning via Task-Oriented Prediction Network

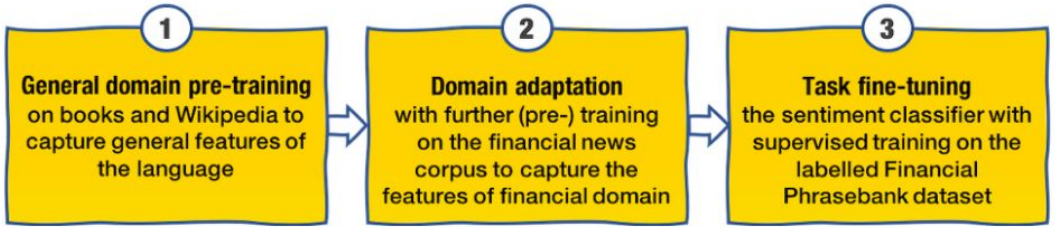


5. Project Implementation

- FinBERT
- HOROVOD
- IMPLEMENTATION
- RESULTS

5.1. FinBERT

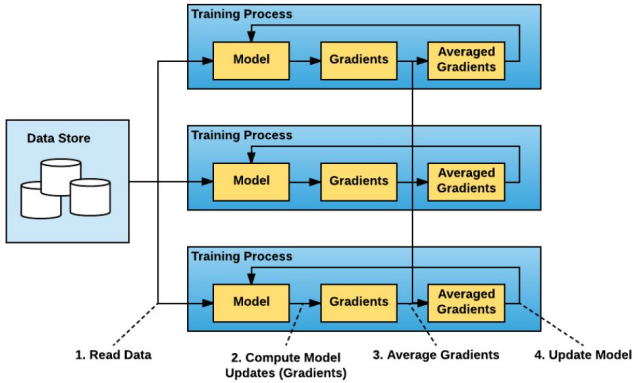
- Sel-supervised learning
- Transfer-Learning
- Fine-tuning
- Transformers
- BERT



The training steps in FinBERT

5.2. HOROVOD

- Rank identification of each worker (CPU/GPU)
- The AllReduce
- The AllGather
- The broadcast



5.3.1. IMPLEMENTATION (FSA) -- pre-training

- **Environnement**:
  Master : 8G RAM, 4CPU .
  4 workers : 4G RAM, 2CPU .
  Hadoop cluster with 2 replications on HDFS .

- **Dataset**: 3000 labeled sentences
  (positive, negative, neutral)

- **Machine learning**:
  - Treat and convert data to be suitable for BERT (tokenization , input formatting)
  - Import the pre-trained BERT (110M of parameters) + classification layer
  - Freeze the 6 first layers of BERT
  - Train the model

```
for (ex_index, example) in enumerate(examples):
    tokens = tokenizer.tokenize(example.text)

    if len(tokens) > max_seq_length - 2:
        tokens = tokens[:(max_seq_length // 4) - 1] + tokens[
                                                      len(tokens) - (3 * max_seq_length // 4) + 1:]
    tokens = ["[CLS]"] + tokens + ["[SEP]"]

    token_type_ids = [0] * len(tokens)
    input_ids = tokenizer.convert_tokens_to_ids(tokens)
    attention_mask = [1] * len(input_ids)
    padding = [0] * (max_seq_length - len(input_ids))
    input_ids += padding
    attention_mask += padding
    token_type_ids += padding
```

```
freeze = 6
for param in model.bert.embeddings.parameters():
    param.requires_grad = False
for i in range(freeze):
    for param in model.bert.encoder.layer[i].parameters():
        param.requires_grad = False
```

```
bertmodel = AutoModelForSequenceClassification.from_pretrained( 'bert-base-uncased',cache_dir=None, num_labels=
config = Config(   data_dir=cl_data_path,
                   bert_model=bertmodel,
                   train_batch_size=16,
                   eval_batch_size=16,
                   num_train_epochs=2,
                   model_dir=cl_path,
                   max_seq_length = 48,
                   learning_rate = 2e-2,
                   output_mode='classification',
                   warm_up_proportion=0.2,
                   local_rank=-1,
                   no_cuda=True,
                   discriminate=True,
                   gradual_unfreeze=True,
                   encoder_no=4)
```

5.3.2 IMPLEMENTATION (FSA) -- pre-training

- **Distributed machine learning** (HOROVOD)
  - Initiate, remove variations due to random initialisation
  - Distribute the dataset then the sampler
  - Broadcast initial parameters and optimizer state
  - Wrap the optimizer with distributed optimiser (to perform average when updating gradient value)

```
hvd.init()
torch.manual_seed(0)
torch.set_num_threads(1)
np.random.seed(0)
```

```
g =tf.io.gfile.GFile(
"hdfs://cluster-bdp-m:8020/user/anne_ngobibinbe/sentiment_data/"+str(hvd.size())+"/train"+str(hvd.rank())+".csv", mode='r')

train = pd.read_csv( g, index_col=False )
```

```
my_sampler = torch.utils.data.distributed.DistributedSampler(
data, num_replicas=hvd.size(), rank=hvd.rank())
```

```
hvd.broadcast_parameters(model.state_dict(), root_rank=0)
hvd.broadcast_optimizer_state(self.optimizer, root_rank=0)
```

```
self.optimizer = hvd.DistributedOptimizer(self.optimizer,
                                          named_parameters=model.named_parameters(),
                                              op= hvd.Average)
```

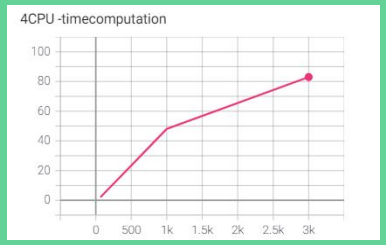
5.4. Results (2 epochs)

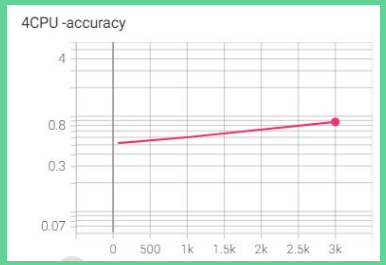
Nbr Processor	dataset size	Loss ; accuracy	Time
4CPU	66	0.91 ; 0.39	2min08
H- 2CPU	66	1.035 ; 0.59	2min42
H-4CPU (4 W)	66	1.48 ; 0.52	4min42
4CPU	1000	0.97 ; 0.60	48min08
H- 2CPU	1000	6,625 ; 0.285	18min21
H- 4CPU (4 M)	1000	3,625 ; 0.21	10min21
4CPU	3000	0.57 ; 0.87	1h 23min
H- CPU >= 4 with little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

- H - means using parallelize machine learning with horovod
- “X” M precise the number of different machines

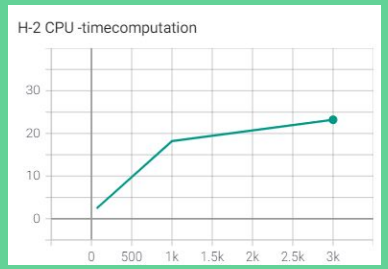
5.4. Results (2 epochs)

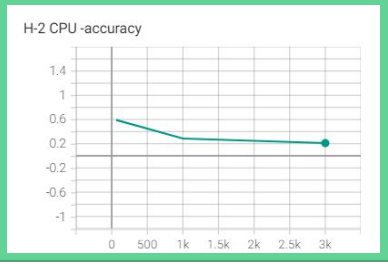
4 CPU



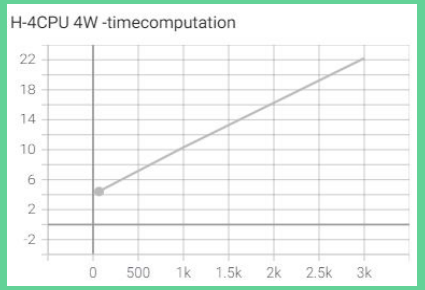


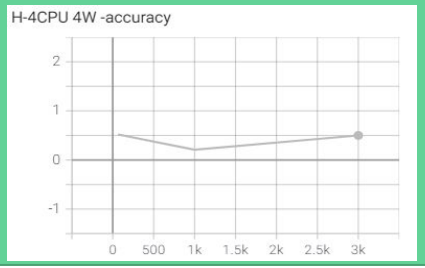
H- 2 CPU





H- 4 CPU (4W)





- H - means using parallelize machine learning with horovod
- “X” M precise the number of different machines

5.5. Result Interpretation

- Improve training time due to data distribution
- Degradation of accuracy due to All_Reduce operation
- The choice of the good architecture is a point that should be taken into account depending on your purpose

5.6. Difficulties

- Hardware capacity
- Reuter TRC2 not opened
- Deadlock with Horovod sometimes
- Failure of AllGather operations sometimes
- Training process was taking too much time
- We couldn’t use all processors of a give machine

6. Conclusion

- Horovod improve training time but degrade the accuracy
- Choosing the good architecture is worthy
- Tony

- Distributed computing can fix the performance issue of online learning techniques that pass with better results than classic models but are slower.

7. References

- Dang et al. “**The Squawk Bot": Joint Learning of Time Series and Text Data Modalities for Automated Financial Information Filtering.,**”, arXiv preprint arXiv:1912.10858 (2019).

- Zhuang et al. “**FinBERT: A Pre-trained Financial Language Representation Model for Financial Text Mining**”, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)

- Di Chen et al. “**Task-Based Learning via Task-Oriented Prediction Network with Applications in Finance**”, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)

- Xia et al. “**Vector Autoregressive Weighting Reversion Strategy for Online Portfolio Selection**”, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)

Thank You