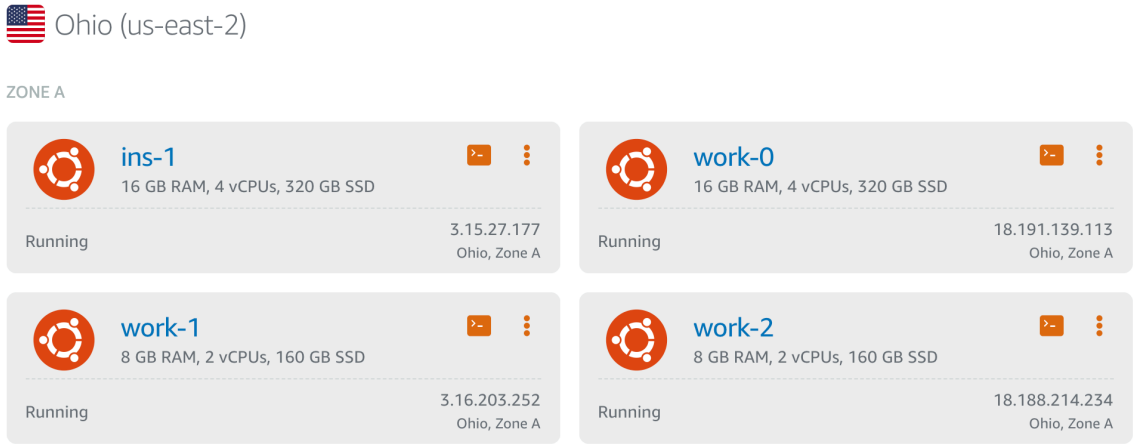
Analysis HK Protest Data from Telegram Application

Yuntao Lu
EID: yl34386

https://github.com/yuntaolu/capstone_0

Build Computing Cluster on AWS

- Apply 4 servers on Amazon Web Service (LightSail Node)
- Configuration server environment
- It took a long time to build an available cluster.
  - SSH Errors, reservation time limitation……



Steps to configure a cluster

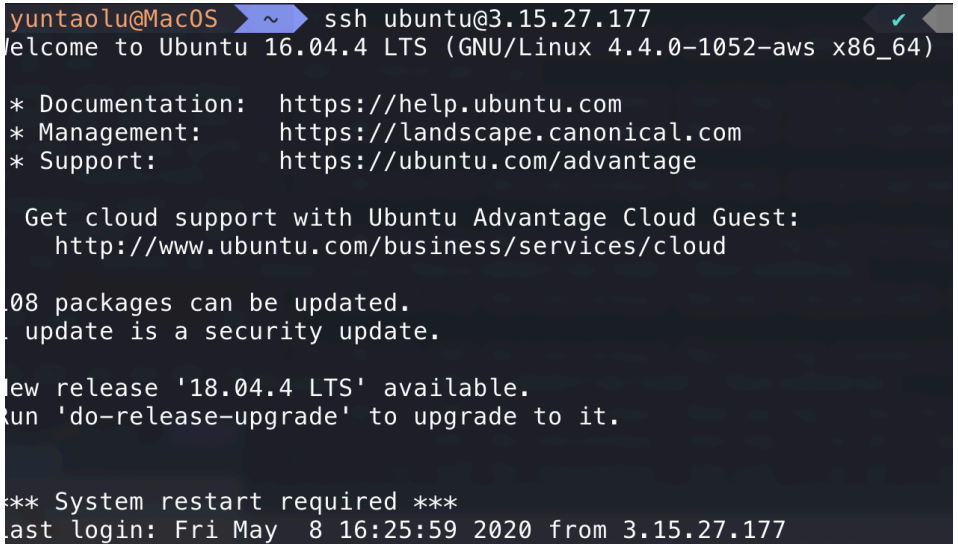
- Step 1 : reserve servers on a Cloud Computing Platform
- First, I use a free cloud platform Chameleon Cloud, which is the interface of TACC.
  - Problem: a user can only reserve 7 days and extend 2 days each time.
- Second, I use AWS to reserve 4 servers
  - Problem: comparing with TACC (48 cpus each node), each server on AWS LightSail only has 1 ~ 8 cpus, and a user can only reserve maximal two same type servers.

Steps to configure a cluster

- Step 1 : reserve servers on a Cloud Computing Platform
  - Create an instance with a public SSH key.
  - Save the private SSH key into ~/.ssh/ on local machine
  - Add the private key to SSH agent

```
ssh-add ~/.ssh/id_rsa
```

  - Connect to the instance



Steps to configure a cluster

- Step 2 : Install essential tools
- Install Anaconda python environment

```
wget https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh
sha256sum Anaconda3-2020.02-Linux-x86_64.sh
bash Anaconda3-2020.02-Linux-x86_64.sh
```

- Install joblib and dask for distribution computing

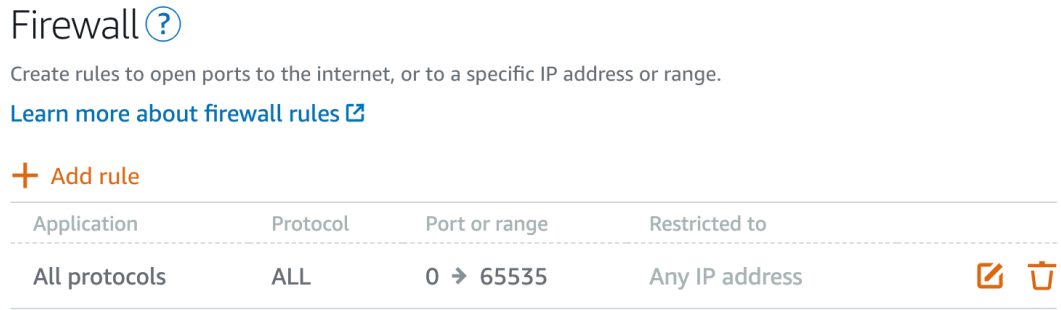
```
conda install dask distributed -c conda-forge
conda install bokeh
pip install paramiko joblib
```

- Reference tutorials
  - http://tomdlt.github.io/blog/dask_distributed_joblib.html
  - https://jima.me/?p=950

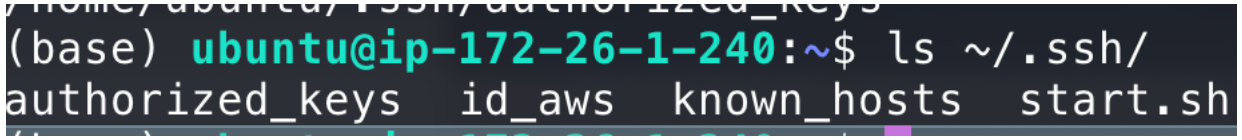
Steps to configure a cluster

- Meet ERRORS
  - Unreachable IP address

- Step 3 : Configure the SSH keys and TCP ports of instances
- In order to connect each instance, open all TCP ports of instances



- Copy public key and private key into ~/.ssh/authorized_keys



Steps to configure a cluster

- Meet ERRORS
  - Unreachable IP address

- Step 4 : Configure the Jupyter Notebook Server
- In order to user Jupyter Notebook on a browser,
  - Generate password
  - Generate configuration file
  - Modify the configuration

```
(base) ubuntu@ip-172-26-1-240:~$ jupyter notebook password
Enter password:
Verify password:
[NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/jupyter_not
ebook_config.json
(base) ubuntu@ip-172-26-1-240:~$ cat ~/.jupyter/jupyter_notebook_config.json
{
  "NotebookApp": {
    "password": "sha1:7e76cc0d4b76:c812f8154101c219677ad03799401d71efb545fc"
```

```
}(base) ubuntu@ip-172-26-1-240:~$cat ~/.jupyter/jupyter_notebook_config.py
# Configuration file for jupyter-notebook.
c.NotebookApp.password = u'sha1:e8eae6105f42:31ee7d405b7e04f73de6f48e11f0024ade
ed3ec'
c.NotebookApp.ip = '*'
c.NotebookApp.port = 8888
```

- Reference tutorials
  - https://jupyter-notebook.readthedocs.io/en/stable/public_server.html

Now test the cluster

- Open browser and input IP address of host http://ip-address:8888/tree?
- Run cluster in a notebook

```
In [*]: !dask-ssh \
            --scheduler 3.15.27.177 \
            --nprocs 4 \
            --nthreads 1 \
            --ssh-username ubuntu \
            --ssh-private-key ~/.ssh/authorized_keys \
            --hostfile ~/list_of_server.txt
```

```
from dask.distributed import Client
import joblib
client = Client("3.15.27.177:8786")
client
```

Client
Scheduler: tcp://3.15.27.177:8786
Dashboard: http://3.15.27.177:8787/status

Cluster
Workers: 12
Cores: 12
Memory: 50.31 GB

- Run a test

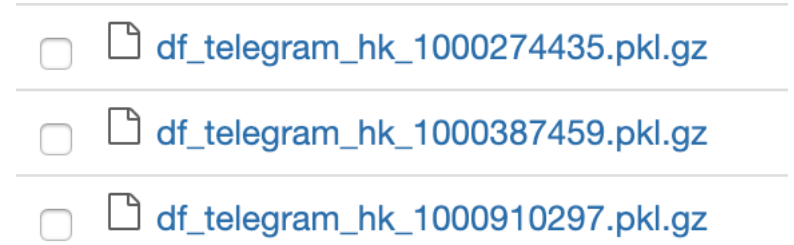
```
from joblib import Parallel, delayed
from math import sqrt

with joblib.parallel_backend('dask'):
    %time _ = Parallel(n_jobs=1)(delayed(sqrt)(param**2) for param in range(10))
```

```
CPU times: user 16 ms, sys: 12 ms, total: 28 ms
Wall time: 43.1 ms
```

Data Preprocessing

- Merge the data from different days and set each channel data into a sole file.
- The folder stored data is ~/data/channel_msg /2019-12-23 (use the date as an example)
  1. Read the daily file in the direction
  2. List the files need to process
  3. Obtain a processed list of files channel_done, if channel in channel_done, go to next step; otherwise, go to step 6.
  4. Read the data file according to the channel.id of each channel file.
  5. Match the channel.id and id and concatenate
  6. Save the datafram the name df_telegram_hk_[channel_id].pkl.gz in ~/data/channel_msg/channel direction
  7. Iterate until all channel have done



Language Detecting

- Because of the mixed languages in different channels, I try to detect languages.
  - many packages can be used
  - langdetect: not accurate
  - textblob: limited length 512
  - polyglot: cannot install “ERROR: no module named icu”
  - urllib: wrong detection {'encoding': 'utf-8', 'confidence': 0.99, 'language': ’’}
  - fasttext: can detect 176 languages (classifier: lid.176.ftz)
  - pycld3: Compact Language Detector v3 .No error

Length of values does not match length of index

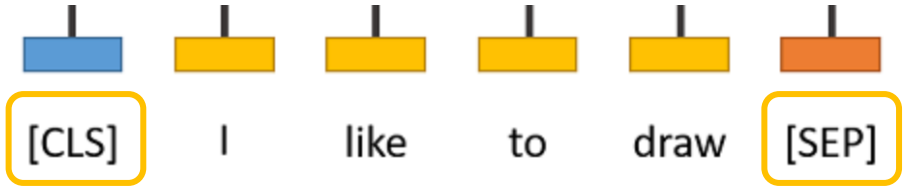
Try to use a pretrained-NLP model

- Google Colab
  - Link: https://drive.google.com/open?id=1kPR_Un4w83Vht3aKUWyAqm7GyALpen2Y
- Pytorch interface with BERT
  - transformers packages
  - torch
- Sentences and Labels
  - Text sequences
  - Predicted labels

https://mccormickml.com/2019/07/22/BERT-fine-tuning/#11-using-colab-gpu-for-training

Tokenizaition

- BERT Tokenizer
  - Split text into tokens, and mapping to the index in the tokenizer vocabulary.
  - using “bert-base-uncased”
- **tokenizer.encode_plus( )**
  - Add special tokens to the start and end of each sentence.

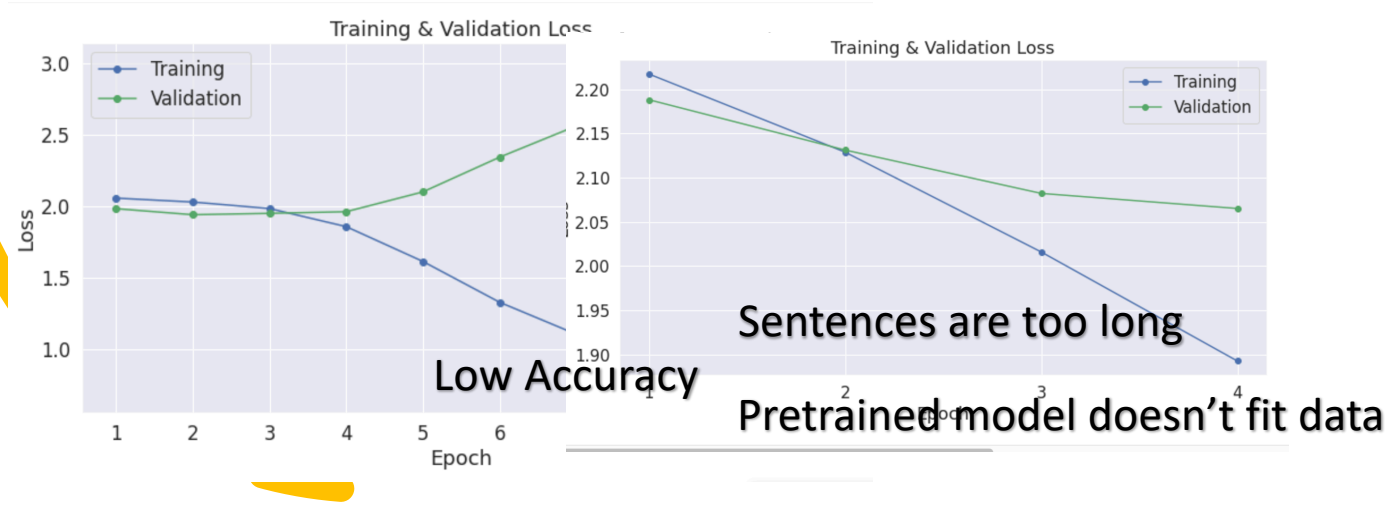


  - Length constraints: all sentences are fixed length && maximum length is 512 tokens (I use 320)
  - Add masks at the end of sentences

https://mccormickml.com/2019/07/22/BERT-fine-tuning/#11-using-colab-gpu-for-training

Training & Evaluation

- BERT BertForSequenceClassification Interface
  - 12 layers
  - using “bert-base-uncased” volabulary
- **Evaluation**



EORROE: Out of memory

- Limited text length: 512
- Training: ~1600
- Validation: ~400

https://mccormickml.com/2019/07/22/BERT-fine-tuning/#11-using-colab-gpu-for-training