[BYOC]
Structure:
container/
(some algorithm name)/: directory that contains the applications to run in the container
OK- nginx.conf: configuration for nginx master server that manages multiple workers
- **predictor.py**:
OK- serve: wrapper that starts the inference server
OK- train: program for TRAINING the model (algorithm)
OK- wsgi.py: the algorithm-specific inference server (own algorithm's code)
Dockerfile: how is the image built? What does it contain?
***build_and_push.sh**: script to build the Docker image >> push it to ECR >> deploy to SageMaker
local-test/: a directory containing scripts and setup for running training/ inference jobs locally (test)
- train-local.sh: instantiate the container configured for TRAINING
- serve-local.sh: instantiate the container configured for SERViNG
- predict.sh: run predictions against a locally instantiated server
- test-dir/: directory that gets mounted into the container w test data mounted < tree is mounted into the container and mimics the directory structure that SageMaker would create for running a container during training/ hosting
- input/config/hyperparaemters.json: hyper parameters for the TRAINING job
- input/data/training/leaf_train.csv: training data
- model: the directory where the algorithm writes the model file
- output: the directory where the algorithm can write its success/ failure file
- payload.csv: sample data for and used by predict.sh (test)

Entrypoint:
– SageMaker runs the container with the argument: TRAIN or SERVE
>> if the ENTRYPOINT is not defined (Dockerfile), Docker will run the command TRAIN at training time and SERVE at serving time (executable Python scripts)
>> if the ENTRYPOINT is defined, the first argument will be TRAIN or SERVE

- [x] Almost done with predictor.py
- [x] Need other files (opened) to be fixed
- [ ] ~~Training?~~
- [x] External modules?

[LOCAL TEST]
WINDOW1:
docker build -t product-tags10 .
docker run --rm -p 8080:8080 -v $(pwd)/local_test/test_dir:/opt/ml product-tags10 serve

WINDOW2:
curl http://127.0.0.1:8080/ping
chmod +x predict.sh
./predict.sh localhostc text/plain payload.txt

ipconfig getifaddr en0

```
docker run -d -p 56733:80 \
  --name=${app} \
  -v $PWD:/app ${app}
```

docker run -d --name mycontainer -p 80:80 myimage

[ERRORS]
~~AttributeError: 'bytes' object has no attribute 'readlines'~~
~~-> f = flask.request.data.decode("utf-8")~~

~~AttributeError: 'str' object has no attribute 'readlines'~~
~~-> f = flask.request.files['file']~~
>> text/html: the accepted content_type changed

Problem: command line으로 들어오는 파일이 0 (empty)

---

os.system()

Problems

1. Entry point approach

- Attempted to solve the issues with installing gluonnlp by using os.system() and 'apt-get install python3-dev'
- Created errors when installing dev package: attempted to solve this by updating apt-get and other solutions on the internet -> did not work

1. Entrypoint 사용 (notebook에 source directory를 만든 후 inference.py와 requirements.txt를 업로드)

- gluonnlp 모듈을 requirements.txt를 사용해서 install할 경우, python.h가 없어서 os 에러가 계속 발생.
- 해결 방법으로는 python3-dev를 설치하는 것. 하지만 이 package는 apt-get install 로만 가능.

- os.system()을 사용해서 cmd에 apt-get install python3-dev와 pip3 install gluonnlp 시도

-> 에러: E: Package 'python3-dev' has no installation candidate
-> 해결 방법으로 제시된 apt-get install을 update, python3-dev를 library로 설치: 계속 에러가 남

2. BYOC 사용
- 이메일에 있는 링크들 >> 구조와 튜토리얼을 따른 코드 정리
- predictor.py와 Dockerfile 새로 만듬: 코드 내에서는 에러가 없음 (다른 파일들은 예시에 있던 것들을 사용함)
- Local test를 위해 predictor.sh의 content type을 바꾸고 local text 실행

-> 모든 local test의 input 파일들의 content length가 0으로 나옴. Content type은 올바르게 나옴

1. BYOC
- Followed structure and tutorials from links provided in email
- Code-wise no error as of now
- However, all the LOCAL TEST inputs aka Flask.request have content-length 0. And because they are all BLANK, predictions are blank as well.

(I've only changed predictor.py, Dockerfile, and predict.sh. The Dockerfile and predictor.py shouldn't matter at this point)

>>

1) Figure out why the 'candidate' error is happening: test it out
2. Run example code > if it is running, compare the two to understand what's wrong with my program

>>

ModelHandler: google.protobuf.message.DecodeError: Error parsing message

[ERRORS]
1. Put model handling code inside a class (or a function)
2. Parsing issue: tested original code locally >> saved_model.pb doesn't work. Only output.pb does.
3. Version mismatch (TensorFlow)

ValueError: NodeDef mentions attr 'incompatible_shape_error' not in Op<name=Equal; signature=x:T, y:T -> z:bool; attr=T:type,allowed=[DT_BFLOAT16, DT_HALF, DT_FLOAT, DT_DOUBLE, DT_UINT8, ..., DT_QINT8, DT_QINT32, DT_STRING, DT_BOOL, DT_COMPLEX128]; is_commutative=true>; NodeDef: {{node Equal}}. (Check whether your GraphDef-interpreting binary is up to date with your GraphDef-generating binary.)

4. Set python to python 3.6 and install TensorFlow 1.15 (python 3.8 does not support pip3 install TensorFlow==1.15)-assuming the graph was trained

using TensorFlow 1.15...I think

>>

As of now, it works for 'output.pb'. Yay
HOWEVER, - not saved_model.pb (need to figure out how .load() works)
- one prediction takes about between 5-10 seconds (I think)
>> that means the 'keepalive_timeout' depends on the input length
>> with 10, nginx times out with the entire payload.txt file (370s would be needed)

1. gluonnlp installed-but can't be imported?!

Subprocess >> test it out~

>>

[ENDPOINT]
Endpoint-deployed (tar.gz + other things changed): because the image works locally
-> a plain text io approach: success
-> to invoke endpoint, it might be easier to get json io
-> a json approach: also a success
-> can now invoke endpoint and get predictions for 37 products (both)