SMS Spam Classifier

This notebook shows how to implement a basic spam classifier for SMS messages using Apache MXNet as deep learning framework. The idea is to use the SMS spam collection dataset available at https://archive.ics.uci.edu/ml/datasets/sms+spam+collection (https://archive.ics.uci.edu/ml/datasets/sms+spam+collection) to train and deploy a neural network model by leveraging on the built-in open-source container for Apache MXNet available in Amazon SageMaker.

Let's get started by setting some configuration variables and getting the Amazon SageMaker session and the current execution role, using the Amazon SageMaker high-level SDK for Python.

In [1]:

```
from sagemaker import get_execution_role

bucket_name = 'sage-maker-bkt'

role = get_execution_role()
bucket_key_prefix = 'sms-spam-classifier'
vocabulary_length = 9013

print(role)
```

```
arn:aws:iam::736691193834:role/service-role/AmazonSageMaker-ExecutionRole-2
0220505T211986
```

We now download the spam collection dataset, unzip it and read the first 10 rows.

In [2]:
```
!mkdir -p dataset
!curl https://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspam
!unzip -o dataset/smsspamcollection.zip -d dataset
!head -10 dataset/SMSSpamCollection
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Cur
rent
                                 Dload  Upload   Total   Spent    Left  Spe
ed
100  198k  100  198k    0     0   307k      0 --:--:-- --:--:-- --:--:--  3
07k
Archive:  dataset/smsspamcollection.zip
  inflating: dataset/SMSSpamCollection
  inflating: dataset/readme
ham     Go until jurong point, crazy.. Available only in bugis n great worl
d la e buffet... Cine there got amore wat...
ham     Ok lar... Joking wif u oni...
spam    Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005.
Text FA to 87121 to receive entry question(std txt rate)T&C's apply 0845281
0075over18's
ham     U dun say so early hor... U c already then say...
ham     Nah I don't think he goes to usf, he lives around here though
spam    FreeMsg Hey there darling it's been 3 week's now and no word back!
I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 t
o rcv
ham     Even my brother is not like to speak with me. They treat me like ai
ds patent.
ham     As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vetta
m)' has been set as your callertune for all Callers. Press *9 to copy your
friends Callertune
spam    WINNER!! As a valued network customer you have been selected to rec
eivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid
12 hours only.
spam    Had your mobile 11 months or more? U R entitled to Update to the la
test colour mobiles with camera for Free! Call The Mobile Update Co FREE on
08002986030
```

We now load the dataset into a Pandas dataframe and execute some data preparation. More specifically we have to:

- replace the target column values (ham/spam) with numeric values (0/1)
- tokenize the sms messages and encode based on word counts
- split into train and test sets
- upload to a S3 bucket for training

In [3]:
```
import pandas as pd
import numpy as np
import pickle
from sms_spam_classifier_utilities import one_hot_encode
from sms_spam_classifier_utilities import vectorize_sequences

df = pd.read_csv('dataset/SMSSpamCollection', sep='\t', header=None)
df[df.columns[0]] = df[df.columns[0]].map({'ham': 0, 'spam': 1})

targets = df[df.columns[0]].values
messages = df[df.columns[1]].values

# one hot encoding for each SMS message
one_hot_data = one_hot_encode(messages, vocabulary_length)
encoded_messages = vectorize_sequences(one_hot_data, vocabulary_length)

df2 = pd.DataFrame(encoded_messages)
df2.insert(0, 'spam', targets)

# Split into training and validation sets (80%/20% split)
split_index = int(np.ceil(df.shape[0] * 0.8))
train_set = df2[:split_index]
val_set = df2[split_index:]

train_set.to_csv('dataset/sms_train_set.gz', header=False, index=False, compr
val_set.to_csv('dataset/sms_val_set.gz', header=False, index=False, compressi
```

We have to upload the two files back to Amazon S3 in order to be accessed by the Amazon SageMaker training cluster.

In [4]:
```
import boto3

s3 = boto3.resource('s3')
target_bucket = s3.Bucket(bucket_name)

with open('dataset/sms_train_set.gz', 'rb') as data:
    target_bucket.upload_fileobj(data, '{0}/train/sms_train_set.gz'.format(bu

with open('dataset/sms_val_set.gz', 'rb') as data:
    target_bucket.upload_fileobj(data, '{0}/val/sms_val_set.gz'.format(bucket
```

```
/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/boto3/c
ompat.py:88: PythonDeprecationWarning: Boto3 will no longer support Python
3.6 starting May 30, 2022. To continue receiving service updates, bug fixe
s, and security updates please upgrade to Python 3.7 or later. More informa
tion can be found here: https://aws.amazon.com/blogs/developer/python-suppo
rt-policy-updates-for-aws-sdks-and-tools/ (https://aws.amazon.com/blogs/dev
eloper/python-support-policy-updates-for-aws-sdks-and-tools/)
  warnings.warn(warning, PythonDeprecationWarning)
```

Training the model with MXNet

We are now ready to run the training using the Amazon SageMaker MXNet built-in container. First let's have a look at the script defining our neural network.

In [5]:
```
!cat 'sms_spam_classifier_mxnet_script.py'
```

```
from __future__ import print_function

import logging
import mxnet as mx
from mxnet import gluon, autograd
from mxnet.gluon import nn
import numpy as np
import json
import time

import pip

try:
    from pip import main as pipmain
except:
    from pip._internal import main as pipmain

pipmain(['install', 'pandas'])
import pandas
```

We are now ready to run the training using the MXNet estimator object of the SageMaker Python SDK.

In [7]:

```
from sagemaker.mxnet import MXNet

output_path = 's3://{0}/{1}/output'.format(bucket_name, bucket_key_prefix)
code_location = 's3://{0}/{1}/code'.format(bucket_name, bucket_key_prefix)

m = MXNet('sms_spam_classifier_mxnet_script.py',
          role=role,
          train_instance_count=1,
          train_instance_type='ml.c5.2xlarge',
          output_path=output_path,
          base_job_name='sms-spam-classifier-mxnet',
          framework_version="1.2",
          code_location = code_location,
          py_version="py3",
          hyperparameters={'batch_size': 100,
                           'epochs': 20,
                           'learning_rate': 0.01})

inputs = {'train': 's3://{0}/{1}/train/'.format(bucket_name, bucket_key_prefi
 'val': 's3://{0}/{1}/val/'.format(bucket_name, bucket_key_prefix)}

m.fit(inputs)
```

```
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html (https://sagemaker.
readthedocs.io/en/stable/v2.html) for details.
train_instance_count has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html (https://sagemaker.
readthedocs.io/en/stable/v2.html) for details.
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html (https://sagemaker.
readthedocs.io/en/stable/v2.html) for details.

2022-05-06 01:39:08 Starting - Starting the training job...
2022-05-06 01:39:31 Starting - Preparing the instances for trainingProfiler
Report-1651801148: InProgress
......
2022-05-06 01:40:36 Downloading - Downloading input data...
2022-05-06 01:41:02 Training - Training image download completed. Training
in progress..2022-05-06 01:41:03,751 INFO - root - running container entryp
oint
2022-05-06 01:41:03,752 INFO - root - starting train task
2022-05-06 01:41:03,756 INFO - container_support.training - Training starti
ng
2022-05-06 01:41:04,646 WARNING - mxnet_container.train - #033[1;33mThis re
quired structure for training scripts will be deprecated with the next majo
r release of MXNet images. The train() function will no longer be required;
instead the training script must be able to be run as a standalone script.
 For more information, see https://github.com/aws/sagemaker-python-sdk/tre
e/master/src/sagemaker/mxnet#updating-your-mxnet-training-script.#033[1;0m
 (https://github.com/aws/sagemaker-python-sdk/tree/master/src/sagemaker/mxn
et#updating-your-mxnet-training-script.#033[1;0m)
2022-05-06 01:41:04,653 INFO - mxnet_container.train - MXNetTrainingEnviron
ment: {'job_name': 'sms-spam-classifier-mxnet-2022-05-06-01-39-08-411', 'ho
sts': ['algo-1'], 'resource_config': {'instance_groups': [{'instance_type':
'ml.c5.2xlarge', 'hosts': ['algo-1'], 'instance_group_name': 'homogeneousCl
uster'}], 'current_group_name': 'homogeneousCluster', 'hosts': ['algo-1'],
```

```
 'current_instance_type': 'ml.c5.2xlarge', 'network_interface_name': 'eth
0', 'current_host': 'algo-1'}, '_scheduler_host': 'algo-1', 'input_config_d
ir': '/opt/ml/input/config', 'enable_cloudwatch_metrics': False, 'output_di
r': '/opt/ml/output', 'channels': {'train': {'TrainingInputMode': 'File',
 'S3DistributionType': 'FullyReplicated', 'RecordWrapperType': 'None'}, 'va
l': {'TrainingInputMode': 'File', 'S3DistributionType': 'FullyReplicated',
 'RecordWrapperType': 'None'}}, '_ps_verbose': 0, 'channel_dirs': {'train':
'/opt/ml/input/data/train', 'val': '/opt/ml/input/data/val'}, 'container_lo
g_level': 20, 'base_dir': '/opt/ml', '_ps_port': 8000, 'model_dir': '/opt/m
l/model', 'sagemaker_region': 'us-east-1', 'user_requirements_file': None,
 'current_host': 'algo-1', 'available_cpus': 8, 'user_script_name': 'sms_sp
am_classifier_mxnet_script.py', '_scheduler_ip': '10.0.99.136', 'available_
gpus': 0, 'input_dir': '/opt/ml/input', 'hyperparameters': {'learning_rat
e': 0.01, 'sagemaker_submit_directory': 's3://sage-maker-bkt/sms-spam-class
ifier/code/sms-spam-classifier-mxnet-2022-05-06-01-39-08-411/source/sourced
ir.tar.gz', 'epochs': 20, 'sagemaker_region': 'us-east-1', 'sagemaker_progr
am': 'sms_spam_classifier_mxnet_script.py', 'sagemaker_job_name': 'sms-spam
-classifier-mxnet-2022-05-06-01-39-08-411', 'sagemaker_container_log_leve
l': 20, 'batch_size': 100}, 'user_script_archive': 's3://sage-maker-bkt/sms
-spam-classifier/code/sms-spam-classifier-mxnet-2022-05-06-01-39-08-411/sou
rce/sourcedir.tar.gz', 'output_data_dir': '/opt/ml/output/data/', 'code_di
r': '/opt/ml/code'}
Downloading s3://sage-maker-bkt/sms-spam-classifier/code/sms-spam-classifie
r-mxnet-2022-05-06-01-39-08-411/source/sourcedir.tar.gz to /tmp/script.tar.
gz
2022-05-06 01:41:04,969 INFO - mxnet_container.train - Starting distributed
training task
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/74/24/0cdbf8907e1e3bc
5a8da03345c23cbed7044330bb8f73bb12e711a640a00/pandas-0.24.2-cp35-cp35m-many
linux1_x86_64.whl (https://files.pythonhosted.org/packages/74/24/0cdbf8907e
1e3bc5a8da03345c23cbed7044330bb8f73bb12e711a640a00/pandas-0.24.2-cp35-cp35m
-manylinux1_x86_64.whl) (10.0MB)
Collecting pytz>=2011k (from pandas)
  Downloading https://files.pythonhosted.org/packages/60/2e/dec1cc18c51b8df
33c7c4d0a321b084cf38e1733b98f9d15018880fb4970/pytz-2022.1-py2.py3-none-any.
whl (https://files.pythonhosted.org/packages/60/2e/dec1cc18c51b8df33c7c4d0a
321b084cf38e1733b98f9d15018880fb4970/pytz-2022.1-py2.py3-none-any.whl) (503
kB)
Requirement already satisfied: python-dateutil>=2.5.0 in /usr/local/lib/pyt
hon3.5/dist-packages (from pandas) (2.7.4)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.5/di
st-packages (from pandas) (1.14.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.5/dist-pa
ckages (from python-dateutil>=2.5.0->pandas) (1.11.0)
Installing collected packages: pytz, pandas
Successfully installed pandas-0.24.2 pytz-2022.1
You are using pip version 18.1, however version 20.3.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
/usr/local/lib/python3.5/dist-packages/mxnet_container/train.py:190: Deprec
ationWarning: inspect.getargspec() is deprecated, use inspect.signature() i
nstead
  train_args = inspect.getargspec(user_module.train)
Train data path: /opt/ml/input/data/train
Validation data path: /opt/ml/input/data/val
[Epoch 0] Training: accuracy=0.731882
[Epoch 0] Validation: accuracy=0.745732
```

```
[Epoch 1] Training: accuracy=0.808391
[Epoch 1] Validation: accuracy=0.839173
[Epoch 2] Training: accuracy=0.852143
[Epoch 2] Validation: accuracy=0.846361
[Epoch 3] Training: accuracy=0.879067
[Epoch 3] Validation: accuracy=0.853549
[Epoch 4] Training: accuracy=0.888490
[Epoch 4] Validation: accuracy=0.874214
[Epoch 5] Training: accuracy=0.899035
[Epoch 5] Validation: accuracy=0.895777
[Epoch 6] Training: accuracy=0.909356
[Epoch 6] Validation: accuracy=0.890386
[Epoch 7] Training: accuracy=0.916536
[Epoch 7] Validation: accuracy=0.903863
[Epoch 8] Training: accuracy=0.920574
[Epoch 8] Validation: accuracy=0.899371
[Epoch 9] Training: accuracy=0.924613
[Epoch 9] Validation: accuracy=0.917341
[Epoch 10] Training: accuracy=0.925286
[Epoch 10] Validation: accuracy=0.914645
[Epoch 11] Training: accuracy=0.929773
[Epoch 11] Validation: accuracy=0.923630
[Epoch 12] Training: accuracy=0.933139
[Epoch 12] Validation: accuracy=0.914645
[Epoch 13] Training: accuracy=0.932690
[Epoch 13] Validation: accuracy=0.927224
[Epoch 14] Training: accuracy=0.935383
[Epoch 14] Validation: accuracy=0.932615
[Epoch 15] Training: accuracy=0.937177
[Epoch 15] Validation: accuracy=0.928122
[Epoch 16] Training: accuracy=0.939870
[Epoch 16] Validation: accuracy=0.929021
[Epoch 17] Training: accuracy=0.940319
[Epoch 17] Validation: accuracy=0.936208
[Epoch 18] Training: accuracy=0.939421
[Epoch 18] Validation: accuracy=0.932615
[Epoch 19] Training: accuracy=0.944133
[Epoch 19] Validation: accuracy=0.936208

2022-05-06 01:42:03 Uploading - Uploading generated training model
2022-05-06 01:42:03 Completed - Training job completed
Training seconds: 87
Billable seconds: 87
```

THE FOLLOWING STEPS ARE NOT MANDATORY IF YOU PLAN TO DEPLOY TO AWS LAMBDA AND ARE INCLUDED IN THIS NOTEBOOK FOR EDUCATIONAL PURPOSES.

Deploying the model

Let's deploy the trained model to a real-time inference endpoint fully-managed by Amazon SageMaker.

In [8]:
```
mxnet_pred = m.deploy(initial_instance_count=1,
                      instance_type='ml.m5.large')
```

```
-----!
```

Executing Inferences

Now, we can invoke the Amazon SageMaker real-time endpoint to execute some inferences, by providing SMS messages and getting the predicted label (SPAM = 1, HAM = 0) and the related probability.

In [9]:
```
from sagemaker.mxnet.model import MXNetPredictor
from sms_spam_classifier_utilities import one_hot_encode
from sms_spam_classifier_utilities import vectorize_sequences

# Uncomment the following line to connect to an existing endpoint.
# mxnet_pred = MXNetPredictor('<endpoint_name>')

test_messages = ["FreeMsg: Txt: CALL to No: 86888 & claim your reward of 3 ho
one_hot_test_messages = one_hot_encode(test_messages, vocabulary_length)
encoded_test_messages = vectorize_sequences(one_hot_test_messages, vocabulary

result = mxnet_pred.predict(encoded_test_messages)
print(result)
```

```
{'predicted_probability': [[0.9998699426651001]], 'predicted_label': [[1.
0]]}
```

Cleaning-up

When done, we can delete the Amazon SageMaker real-time inference endpoint.

In [ ]:
```
mxnet_pred.delete_endpoint()
```