# PE07: Programming Exercise

Fall 2024 by Verónica Elze

## NTM in SageMaker

Closely follow the textbook chapter 9 and Lecture 07.pptx to implement NTM in SageMaker:

- PE07-1 Data
  - Follow textbook 304 to get Enron emails
    - 39.861 emails and 28101 unique words
    - We will work with a subset of these emails – 3986 emails and 17524 unique words

EnronEmailsTopics.ipynb

Code git conda_python3

[2]:
```
# **Acknowledgements, Copyright Information, and Availability**
#Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and C
------
import pandas as pd
import numpy as np

from sklearn.preprocessing import normalize
from scipy.sparse import csr_matrix
import os
from sagemaker import get_execution_role
import boto3
from sagemaker.amazon.amazon_estimator import get_image_uri
import sagemaker
from sagemaker.session import s3_input
from sagemaker.model import Model
# from sagemaker.predictor import csv_serializer, json_deserializer
from sagemaker.serializers import CSVSerializer
from sagemaker.deserializers import JSONDeserializer
import warnings
warnings.simplefilter(action='ignore')
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

## Preprocessing

First, let's run user defined functions used to conduct common operations

[3]:
```
run bowemails.py
```

### Create Bag-of-Words and Vocabulary

We will only take 10% of emails to have a manageable dataset

[4]:
```
ip_fn = '../data/docword.enron.txt.gz'
percent_emails = .10 # get only a x% of emails to avoid memory errors
vocab_ip_fn = '../data/vocab.enron.txt'
vocab_op_fn = '../data/vocab.txt'

#Get bag-of-words from input of enron emails
# We will filter emails to reduce data size
# Create vocabulary based on the subset of emails that will be sent to training
pvt_emails = prepare_bow_vocab(ip_fn, percent_emails, vocab_ip_fn, vocab_op_fn)
```

[5]:
```
pvt_emails.head()
```

[5]:

| word_ID | 1 | 3 | 4 | 8 | 9 | 15 | 16 | 19 | 20 | 21 | ... | 28090 | 28091 | 28092 | 28093 | 28095 | 28096 | 28097 | 28098 | 28100 | 28101 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| email_ID | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 17524 columns

EnronEmailsTopics.ipynb

Markdown git conda_python3

[6]:
```
pvt_emails.shape
```

[6]:
```
(3986, 17524)
```

## TF-IDF Term Frequency Inverse Document Frequency

We assume that the words that help surface topics are those that are not repeated across all emails but are common within an email.

[7]:
```
tfidf_emails = TF_IDF(pvt_emails)
```

[8]:
```
# convert pivoted dataframe to compressed sparse row matrix
# compressed sparse row matrix contains row pointer, column index and values
sparse_emails = csr_matrix(pvt_emails, dtype=np.float32)
print(sparse_emails[:16].toarray())
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

[9]:
```
type(sparse_emails)
```

[9]:
```
scipy.sparse._csr.csr_matrix
```

## Create Training Validation and Test Datasets

[10]:
```
vol_train = int(0.8 * sparse_emails.shape[0])

# split train and test
train_data = sparse_emails[:vol_train, :]
test_data = sparse_emails[vol_train:, :]

# further split test set into validation set and test set
vol_test = test_data.shape[0]
val_data = test_data[:vol_test//2, :]
test_data = test_data[vol_test//2:, :]
```

[11]:
```
print(train_data.shape, test_data.shape, val_data.shape)
```

```
(3188, 17524) (399, 17524) (399, 17524)
```

)

EnronEmailsTopics.ipynb ● | Terminal 1 ×

Markdown | git | conda_python3

Set Hyperparameters

**feature_dim** - Represents the size of feature vector. It is set to vocabulary size
**num_topics** - Represents number of topics to extract. We can choose a number here and adjust it based on model performance on test set
**mini_batch_size** - Represents number of training examples to process before updating weights
**epochs** - Represents number of backward and forward passes
**num_patience_epochs** Represents maximum number of bad epochs (epochs where loss does not improve) executed before stopping
**optimizer** - We use Adadelta optimization algorithm. Adpative Delta gradient is an enhanced version of Adagrad (Adaptive Gradient), where learning rate decreases based on rolling window of gradient updates vs all past gradient updates
**tolerance** - Represents threshold for change in loss function - the trainining stops early if the change in loss within the last designated number of patience epochs falls below this threshold

[63]:
```
num_topics = 3
vocab_size = 17524 # from shape from pivoted emails dataframe
ntm_estmtr.set_hyperparameters(num_topics=num_topics,
                       feature_dim=vocab_size,
                       mini_batch_size=30,
                       epochs=150,
                       num_patience_epochs=5,
                       tolerance=.001)
```

[64]:
```
# Upload vocabulary file to auxiliary folder on S3 bucket -- this is used to identify words associated with latent topics
# aux_path = s3_aux_data + "/"
aux_path = s3loc_aux_data + "/"

!aws s3 cp $vocab_op_fn $aux_path
```

```
upload: ../data/vocab.txt to s3://ai620-in-aws/enronemails/aux/vocab.txt
```

[65]:
```
# s3_train = s3_input(s3loc_train_data, distribution='ShardedByS3Key', content_type='application/x-recordio-protobuf')
s3_train = sagemaker.inputs.TrainingInput(s3loc_train_data, distribution='ShardedByS3Key', content_type='application/x-recordio-protobuf')

# s3_val = s3_input(s3loc_val_data, distribution='FullyReplicated', content_type='application/x-recordio-protobuf')
s3_val = sagemaker.inputs.TrainingInput(s3loc_val_data, distribution='FullyReplicated', content_type='application/x-recordio-protobuf')

# s3_aux = s3_input(s3loc_aux_data, distribution='FullyReplicated', content_type='text/plain')
s3_aux = sagemaker.inputs.TrainingInput(s3loc_aux_data, distribution='FullyReplicated', content_type='text/plain')
# s3_aux = "s3://ai620-in-aws/enronemails/auxiliary/"
```

[66]:
```
ntm_estmtr.fit({'train': s3_train, 'validation': s3_val, 'auxiliary': s3_aux})
```

```
affiliates carr affiliate author sitting aug03
[11/24/2024 08:19:04 INFO 140341878044480] [0.30, 1.00] clicking web click represent sources chart message prohibited director receive instruction client phil
lip material august accurate solicitation offer corp link
[11/24/2024 08:19:04 INFO 140341878044480] [0.07, 1.00] yippee tail flip hookie resource sooo volunteer pending msa queue mgr coin sweet junior wearing approv
al flipping bidoffer pao league
[11/24/2024 08:19:04 INFO 140341878044480] Serializing model to /opt/ml/model/model_algo-2
[11/24/2024 08:19:04 INFO 140341878044480] Saved checkpoint to "/tmp/tmpkvyxt2eg/state-0001.params"
[11/24/2024 08:19:04 INFO 140341878044480] Test data is not provided.
#metrics {"StartTime": 1732522620.8432927, "EndTime": 1732522744.1579876, "Dimensions": {"Algorithm": "AWS/NTM", "Host": "algo-2", "Operation": "training"},
"Metrics": {"initialize.time": {"sum": 10778.23781967163, "count": 1, "min": 10778.23781967163, "max": 10778.23781967163}, "epochs": {"sum": 150.0, "count":
1, "min": 150, "max": 150}, "model.score.time": {"sum": 7505.507707595825, "count": 48, "min": 121.28925323486328, "max": 197.5710391998291}, "early_stop.tim
e": {"sum": 7425.658464431763, "count": 47, "min": 124.48740005493164, "max": 197.75700569152832}, "update.time": {"sum": 67394.36793327332, "count": 47, "mi
n": 1331.007719039917, "max": 1946.1092948913574}, "finalize.time": {"sum": 158.32161903381348, "count": 1, "min": 158.32161903381348, "max": 158.321619033813
48}, "model.serialize.time": {"sum": 3.3936500549316406, "count": 1, "min": 3.3936500549316406, "max": 3.3936500549316406}, "setuptime": {"sum": 49.3195056915
2832, "count": 1, "min": 49.31950569152832, "max": 49.31950569152832}, "totaltime": {"sum": 123382.44128227234, "count": 1, "min": 123382.44128227234, "max":
123382.44128227234}}}

2024-11-24 08:19:23 Completed - Training job completed
Training seconds: 764
Billable seconds: 764
```

[67]:
```
print('Training job name: {}'.format(ntm_estmtr.latest_training_job.job_name))
```

```
Training job name: ntm-2024-11-24-08-12-09-791
```

- PE07-2 Implementation in SageMaker
  - Follow textbook page 311 step 1 to 13

EnronEmailsTopics.ipynb ● | Terminal 1 ×

Markdown | git | conda_python3

## Model Hosting and Inference

[68]:
```
#You can either deploy the trained model using sagemaker estimator object or create a sagemaker and then invoke deploy() method

ntm_predctr = ntm_estmtr.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')

# OR

# current_job_name = 'ntm-2019-09-02-00-11-12-089'
# model_path = os.path.join('s3://', bucket, output_prefix, current_job_name, 'output/model.tar.gz')
# # ntm_model = Model(model_data=model_path, image=container, role=role, sagemaker_session=sess)
# # Correct the Model initialization
# ntm_model = Model(
#     model_data=model_path,
#     image_uri=container,  # Use image_uri instead of image
#     role=role,
#     sagemaker_session=sess
# )
# ntm_model.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
# ntm_predctr = sagemaker.predictor.RealTimePredictor(
#     ntm_model.endpoint_name,
#     sagemaker_session=sess
#      )
# Set up the predictor
ntm_predctr = sagemaker.predictor.Predictor(
    ntm_model.endpoint_name,
    sagemaker_session=sess
)
```

```
INFO:sagemaker:Creating model with name: ntm-2024-11-24-08-21-23-792
INFO:sagemaker:Creating endpoint-config with name ntm-2024-11-24-08-21-23-792
INFO:sagemaker:Creating endpoint with name ntm-2024-11-24-08-21-23-792
--------------!
```

[69]:
```
print('Endpoint name: {}'.format(ntm_model.endpoint_name))
```

```
Endpoint name: None
```

- PE07-3 Deploy the trained NTM model and running the inference
  - Follow textbook page 317 step 1 to 5, and plot the topics

## Inference with CSV

[70]:
```
ntm_predctr.content_type = 'text/csv'
ntm_predctr.serializer = CSVSerializer()
ntm_predctr.deserializer = JSONDeserializer()
```

[73]:
```
#Convert test vectors from compressed sparse matrix to dense matrix
if issparse(test_data):
    test_data = np.array(test_data.todense())
else:
    test_data = np.array(test_data)

print(type(test_data))
```

```
<class 'numpy.ndarray'>
```

[77]:
```
endpoint_name = "ntm-2024-11-24-08-21-23-792"
ntm_predctr = Predictor(endpoint_name=endpoint_name)
```

[78]:
```
print(ntm_predctr.endpoint_name)
```

```
ntm-2024-11-24-08-21-23-792
```

[81]:
```
# Convert NumPy array to CSV string
csv_buffer = io.StringIO()
np.savetxt(csv_buffer, test_data[1:6], delimiter=",")
serialized_data = csv_buffer.getvalue()
```

[84]:
```
results = ntm_predctr.predict(
    serialized_data.encode("utf-8"),
    initial_args={"ContentType": "text/csv"}
)
print(results)
```

```
b'{"predictions":[{"topic_weights":[0.067165561,0.8087602258,0.1240743324]},{"topic_weights":[0.0712335706,0.8568163514,0.0719500855]},{"topic_weights":[0.06307
96179,0.8224505782,0.1144697517]},{"topic_weights":[0.1324665844,0.6991248727,0.1684084982]},{"topic_weights":[0.1265615821,0.7397805452,0.133657977]}]}'
```

# ntm-2024-11-24-08-21-23-792

Clone | Delete | Apply to endpoint

## Endpoint configuration settings

| Name | ARN | Encryption key | Creation time |
|---|---|---|---|
| ntm-2024-11-24-08-21-23-792 | arn:aws:sagemaker:us-east-1:850995577816:endpoint-config/ntm-2024-11-24-08-21-23-792 | - | 11/24/2024, 11:21:24 AM |

## Data capture

| Enable data capture | Data capture options | S3 location to store data collected | Capture content type |
|---|---|---|---|
| No | - | | - |

Sampling percentage (%)
-

## Variants

Identifies a model that you want to host and the resources chosen to deploy for hosting it.

### P Production

| Model name | Training job | Variant name | Instance type | Elastic Inference | Initial instance count | Initial weight | Volume Size | Model Data |
|---|---|---|---|---|---|---|---|---|
| ntm-2024-11-24-08-21-23-792 | ntm-2024-11-24-08-12-09-791 | AllTraffic | ml.m4.xlarge | - | 1 | 1 | - | - |

EnronEmailsTopics.ipynb | Terminal 1

Markdown | git | conda_python3

[ ]:
```
# ntm_predctr.accept = ["text/csv"]
# results = ntm_predctr.predict(test_data[1:6])
# print(results)
```

[88]:
```
print(type(results))
```

```
<class 'bytes'>
```

[90]:
```
decoded_results = results.decode("utf-8")
print(decoded_results)
```


```
{"predictions":[{"topic_weights":[0.067165561,0.8087602258,0.1240743324]},{"topic_weights":[0.0712335706,0.8568163514,0.0719500855]},{"topic_weights":[0.0630796179,0.8224505782,0.1144697517]},{"topic_weights":[0.1324665844,0.6991248727,0.1684084982]},{"topic_weights":[0.1265615821,0.7397805452,0.133657977]}]}
```


[91]:
```
parsed_results = json.loads(decoded_results)
print(parsed_results)
```


```
{'predictions': [{'topic_weights': [0.067165561, 0.8087602258, 0.1240743324]}, {'topic_weights': [0.0712335706, 0.8568163514, 0.0719500855]}, {'topic_weights': [0.0630796179, 0.8224505782, 0.1144697517]}, {'topic_weights': [0.1324665844, 0.6991248727, 0.1684084982]}, {'topic_weights': [0.1265615821, 0.7397805452, 0.133657977]}]}
```


[ ]:
```
# topic_wts_res = np.array([prediction['topic_weights'] for prediction in results['predictions']])
# print(topic_wts_res)
```

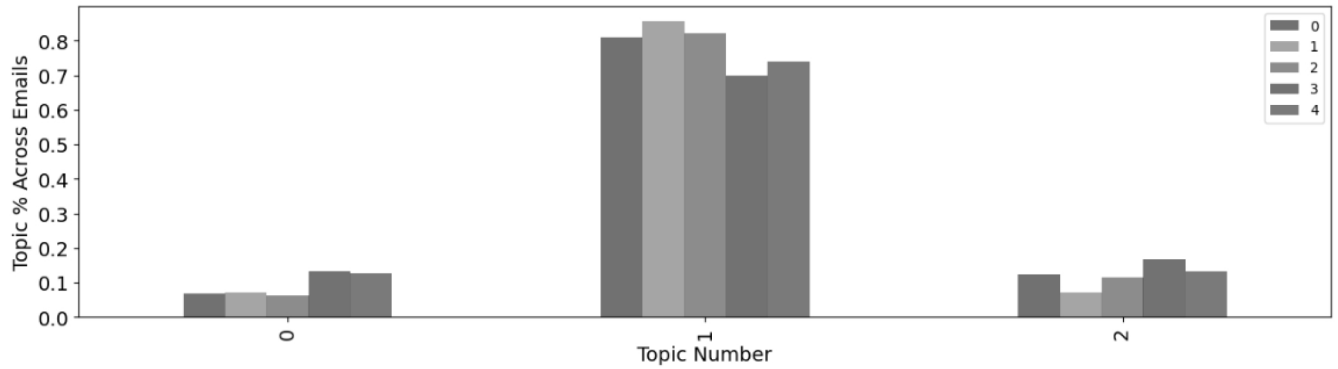
[92]:
```
topic_wts_res = np.array([prediction['topic_weights'] for prediction in parsed_results['predictions']])
print(topic_wts_res)
```

```
[[0.06716556 0.80876023 0.12407433]
 [0.07123357 0.85681635 0.07195009]
 [0.06307962 0.82245058 0.11446975]
 [0.13246658 0.69912487 0.1684085 ]
 [0.12656158 0.73978055 0.13365798]]
```

[93]:
```
import matplotlib.pyplot as plt
%matplotlib inline

fnt_sz=14

df_tpcwts=pd.DataFrame(topic_wts_res.T)

df_tpcwts.plot(kind='bar', figsize=(16,4), fontsize=fnt_sz)

plt.ylabel('Topic % Across Emails', fontsize=fnt_sz)
plt.xlabel('Topic Number', fontsize=fnt_sz)
```

```
INFO:matplotlib.font_manager:generated new fontManager
```

[93]:
```
Text(0.5, 0, 'Topic Number')
```



**Your task for PE07 is to:**

1. Complete above task

**Please make sure you remove all the running resources after finishing the task. Note that you will need to specifically shutdown the deployed inference.**

**HINT**:

- Should be a light PE so that you can focus on team project progress report. It took me 9 hours over 3 days to accomplish this PE with the help of the code provided by the textbook, instructor's video, and ChatGPT. I received ***more*** errors than the instructor experienced in the recorded video.
- Feel free to follow the textbook's GitHub codes if needed: https://github.com/PacktPublishing/Hands-On-Artificial-Intelligence-on-Amazon-Web-Services/tree/master/Chapter09
- Feel free to follow this step-by-step video to complete this PE

**Submit the items below to the PE submission page:**

- Please provide screenshots after completing task1 PE07-2, PE07-3
- Please provide around 100 words of overall feelings completing this whole PE.
  Completing this programming exercise was a rewarding yet challenging experience. It offered valuable hands-on exposure to working with AWS SageMaker, ECR, and deploying machine learning models. Troubleshooting errors, particularly around permissions and endpoint configurations, enhanced my understanding of cloud infrastructure intricacies. While the initial setup and debugging required persistence, the process deepened my appreciation for the practical applications of ML deployment. Successfully deploying the model and running predictions was a fulfilling moment that validated the effort invested. Overall, this exercise highlighted the importance of attention to detail, systematic problem-solving, and adaptability when working with cloud-based AI tools.

  - Make sure the PE module number and your name are written on the file name, e.g., "*PE07_Name.docx").*