AI 620 Emerging Topics in Artificial Intelligence

HOS07A Neural Topic Model (NTM) in SageMaker

03/20/2023 Developed by Yared Shewarade

09/21/2024 Updated by Anh Nguyen

9/15/2024 Reviewed by Jonathan Koerber

School of Technology and Computing (STC) @City University of Seattle (CityU)

Before You Start

- The directory path shown in screenshots may be different from yours.
- Some steps are not explained in the tutorial**.** If you are not sure what to do:
    1. Consult the resources listed below.
    2. If you cannot solve the problem after a few tries, the courses student worker for help.

Learning Outcomes

Students will be able to learn:

- Introduction to Neural Topic Model (NTM)
- Setup Notebook for SageMaker
- Training NTM in SageMaker
- Deploying NTM and running inference

Resources

- Tripuraneni, S., & Song, C. (2019). *Hands-on artificial intelligence on amazon web services: Decrease the time to market for AI and ML applications with the power of AWS* (1st ed.). Packt.

Introduction to Neural Topic Model (NTM)

Topic modeling is the process of learning, recognizing, and extracting topics which is one of the most useful ways to understand text. Understanding topics in text can be used in the legal industry to surface themes from contracts, in the retails industry to identify broad trends in social media conversations, product innovation-introduce new merchandise into online and physical stores, to inform others of product assortment, and so on.

Structured and unstructured data are being generated at an unprecedented rate which is challenging to make the data useful. More than 80% of the data in enterprise is unstructured data that needs the right tools to organize, search, and understand this vast amount of information.

Text analysis is the process of converting unstructured text into meaningful data for analysis to support fact-based decision making. There are different techniques used for text analytics, such as topic modeling, entity and key phrases extraction, sentiment analysis, and coreference resolution.

Topic Modeling is used to organize a corpus of documents into "topics" which is a grouping based on a statistical distribution of words within the documents themselves. The technical definition of topic modeling is that each topic is a distribution of words, and each document is a mixture of topics across a set of documents. For example, a collection of documents that contains frequent occurrences of words such as 'bike', 'car', 'mile', 'brake' , and 'speed' are likely to share a topic on "transportation". It can be used to summarize documents based on topic similarities.

The Neural Topic Model (NTM) is a generative document model that produces multiple representations of a document based on the vibrational autoencoder architecture. It generates two outputs:

- The topic mixture for a document
- A list of keywords that explain a topic, for all the topics across an entire corpus.

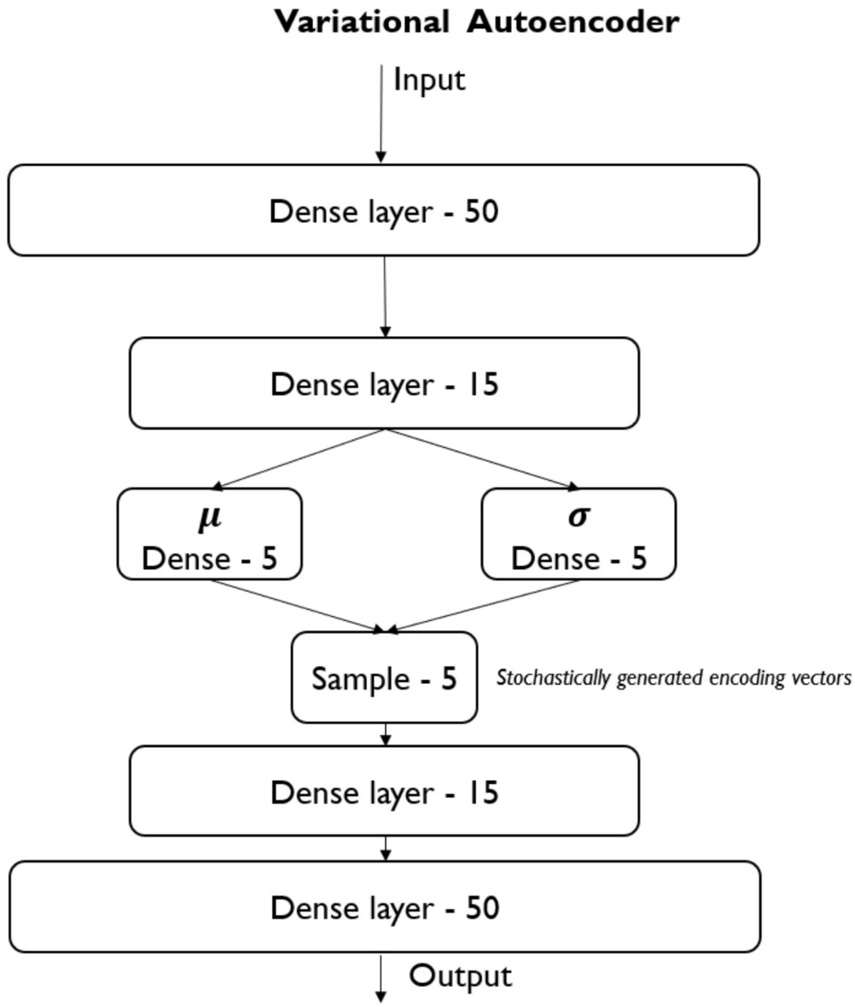
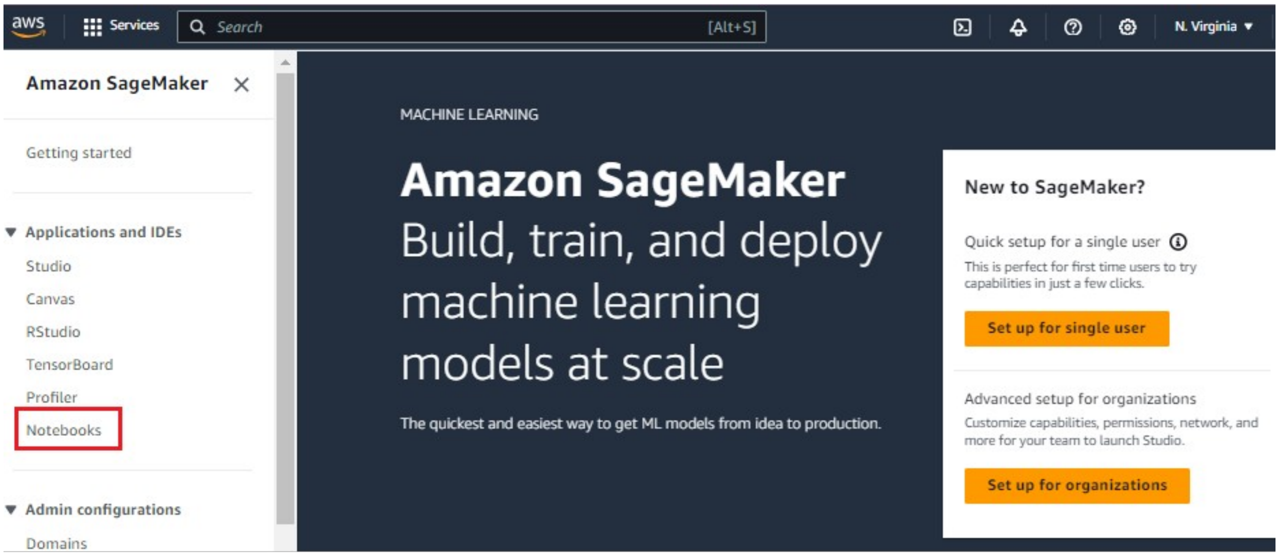


Fig. 1. The working structure of Neural Topic Model (NTM)

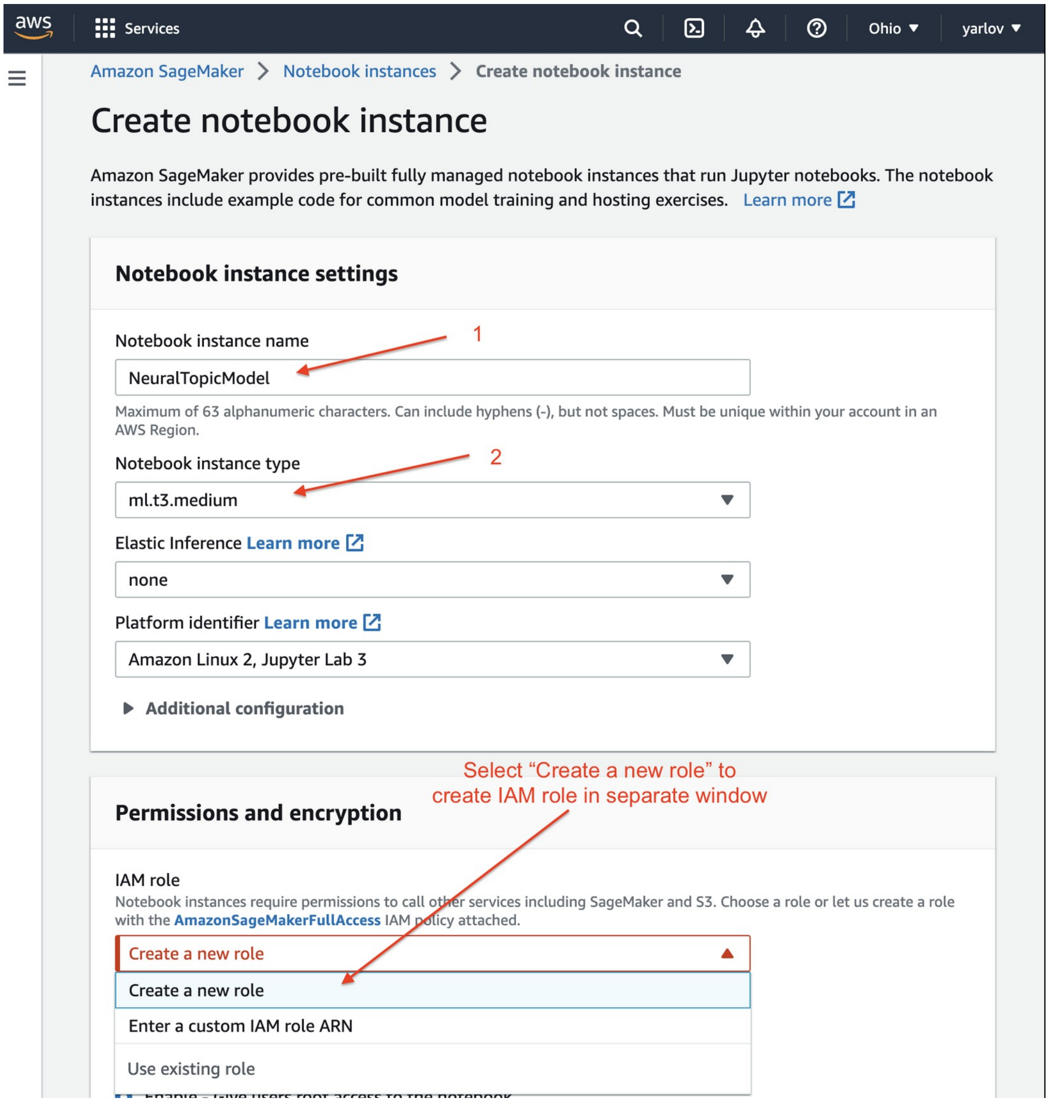
Note: For submission, take the screenshot for all steps and save it in your local repository along with your code.

Setup Notebook for NTM

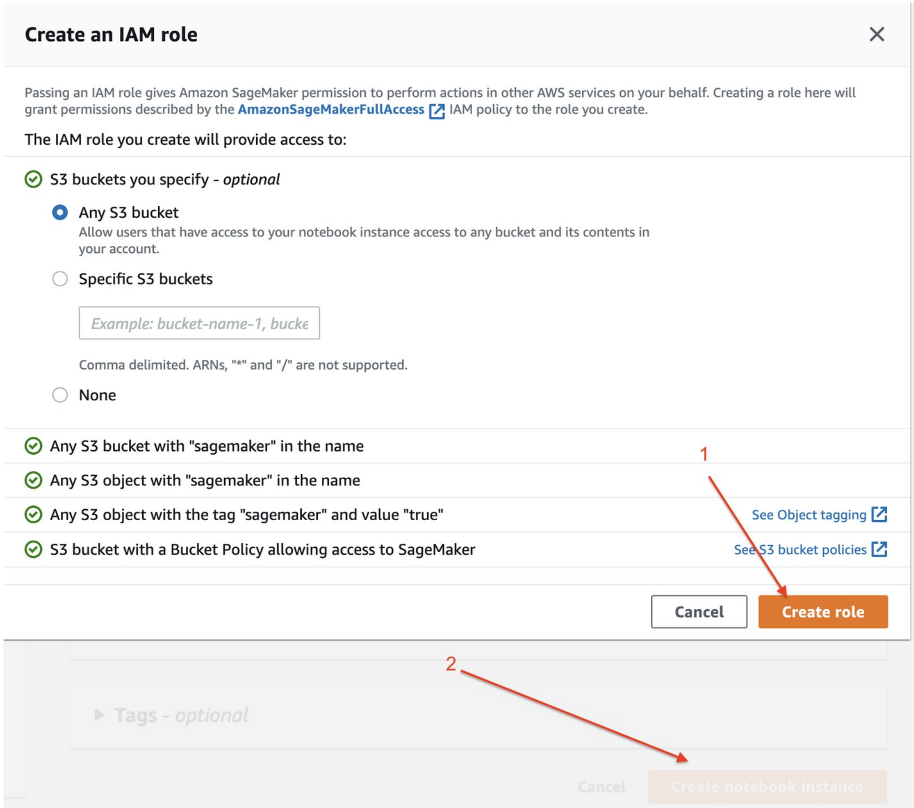
A. Go to your AWS account, Amazon SageMaker, and click on Notebook. Click Create notebook instance



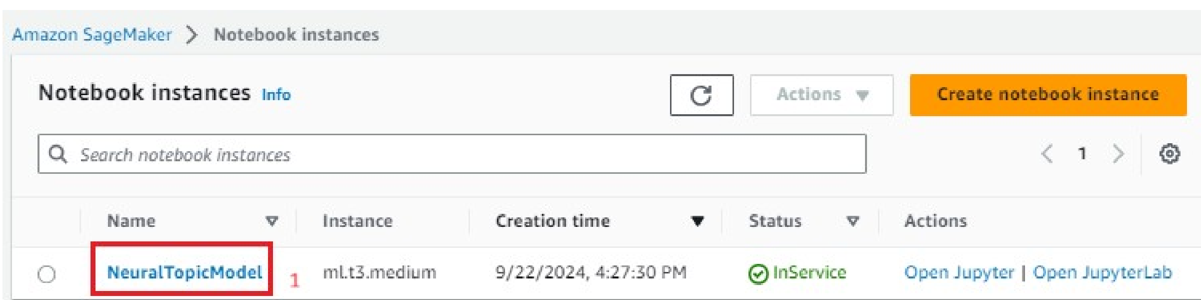
B. Give Notebook instance name and select Notebook instance type: ml.t3.medium



C. Follow the steps to create an IAM role. Then select the **Create notebook instance** button at the end of the page.

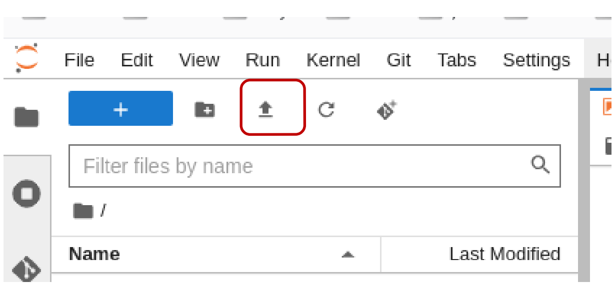


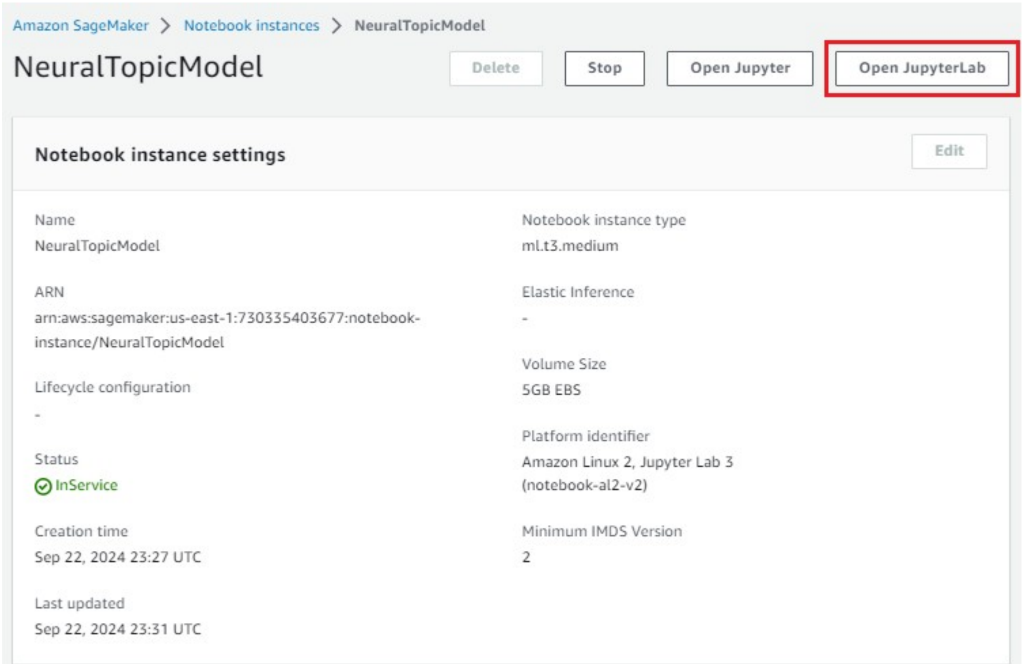
Wait until the notebook instance’s Status changes to InService. This can take a little while.

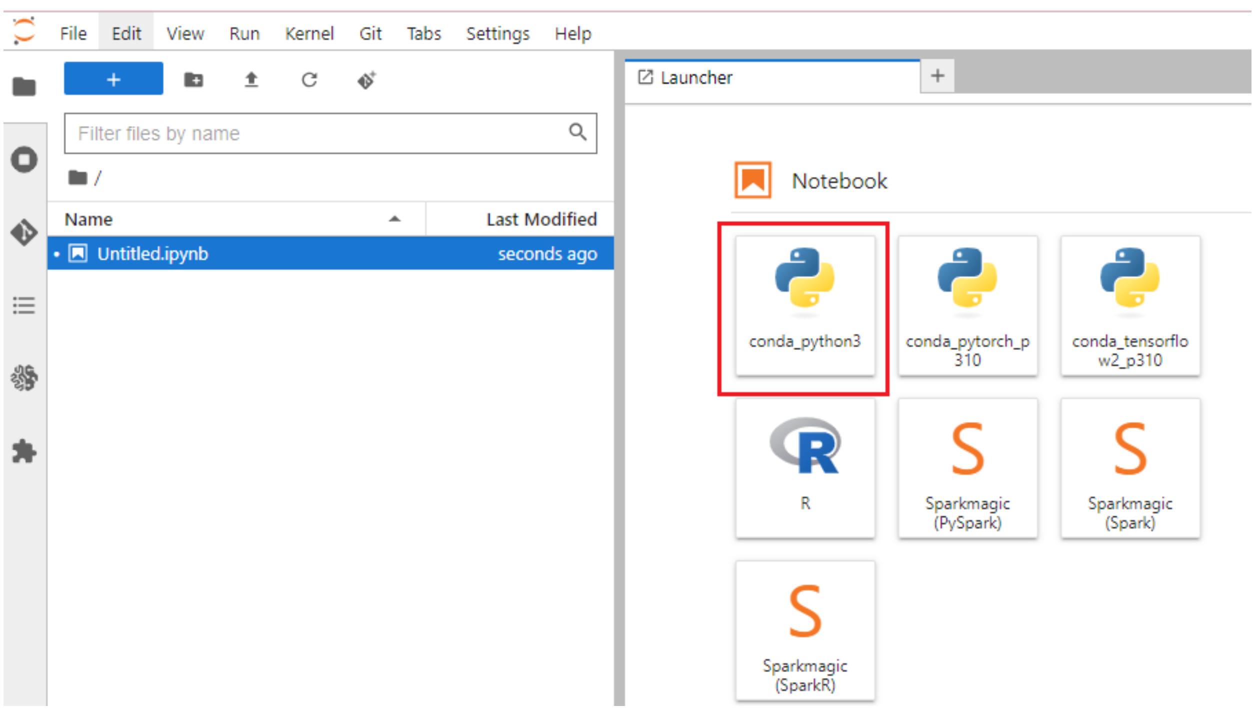


Click Open JupyterLab > conda_python3 to create a new notebook

Your assignment repo comes with a starter notebook. You can upload this notebook to complete the assignment.







Training NTM in SageMaker

1. ## Fetching Data Set

First let’s define the folder to hold the data and clean the content in it which might be from previous experiments.

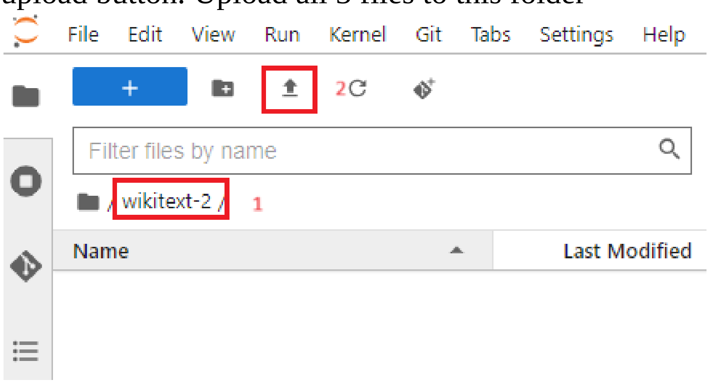
```
In [11]: import os
         import shutil


         def check_create_dir(dir):
             if os.path.exists(dir):  # cleanup existing data folder
                 shutil.rmtree(dir)
             os.mkdir(dir)


         dataset = "wikitext-2"
         current_dir = os.getcwd()
         data_dir = os.path.join(current_dir, dataset)
         check_create_dir(data_dir)
         os.chdir(data_dir)
         print("Current directory: ", os.getcwd())

         Current directory:  /home/ec2-user/SageMaker/wikitext-2/wikitext-2
```

Let's download the wikitext-2 data from Kaggle and unzip it to your local computer.

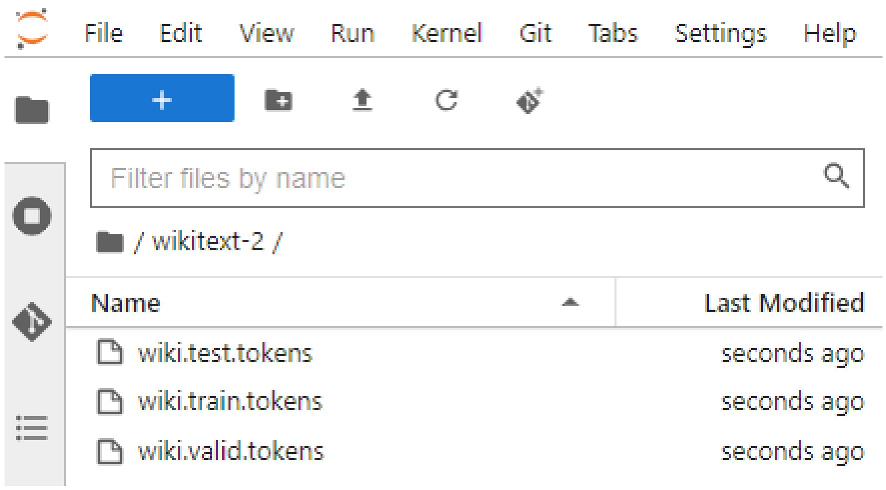
There should be 3 files:

- wiki.train.tokens
- wiki.val.tokens
- Wiki.test.tokens

Upload these files to the SageMaker notebook. On the Jupyter notebook, select the wikitext-2 folder and select the upload button. Upload all 3 files to this folder



Your wikitext-2 should look like this:



2. Preprocessing

Let's first parse the input files into separate documents. We can identify each document by its title in level-1 handling.

[3]:
```
def is_document_start(line):
    if len(line) < 4:
        return False
    if line[0] == "=" and line[-1] == "=":
        if line[2] != "=":
            return True
        else:
            return False
    else:
        return False
```

[4]:
```
def token_list_per_doc(input_dir, token_file):
    lines_list = []
    line_prev = ""
    prev_line_start_doc = False
    with open(os.path.join(input_dir, token_file), "r", encoding="utf-8") as f:
        for line in f:
            line = line.strip()
            if prev_line_start_doc and line:
                # the previous line should not have been the start of the document
                lines_list.pop()
                lines_list[-1] = lines_list[-1] + " " + line_prev
            if line:
                if is_document_start(line) and not line_prev:
                    lines_list.append(line)
                    prev_line_start_doc = True
                else:
                    lines_list[-1] = lines_list[-1] + " " + line
                    prev_line_start_doc = False
            else:
                prev_line_start_doc = False
            line_prev = line
    print("{} documents parsed!".format(len(lines_list)))
    return lines_list
```

[5]:
```
train_file = "wiki.train.tokens"
val_file = "wiki.valid.tokens"
test_file = "wiki.test.tokens"
train_doc_list = token_list_per_doc(data_dir, train_file)
val_doc_list = token_list_per_doc(data_dir, val_file)
test_doc_list = token_list_per_doc(data_dir, test_file)
```

```
600 documents parsed!
60 documents parsed!
60 documents parsed!
```

Let’s install and import nltk.

```
[6]: !pip install nltk

Requirement already satisfied: nltk in /home/ec2-user/anaconda3/envs/python3/lib/python3.
Requirement already satisfied: click in /home/ec2-user/anaconda3/envs/python3/lib/python3
Requirement already satisfied: joblib in /home/ec2-user/anaconda3/envs/python3/lib/pythor
Requirement already satisfied: regex>=2021.8.3 in /home/ec2-user/anaconda3/envs/python3/l
Requirement already satisfied: tqdm in /home/ec2-user/anaconda3/envs/python3/lib/python3.
```

```
[7]: import nltk

# nltk.download("punkt")
nltk.download("wordnet")
from nltk.stem import WordNetLemmatizer
import re

token_pattern = re.compile(r"(?u)\b\w\w+\b")

class LemmaTokenizer(object):
    def __init__(self):
        self.wnl = WordNetLemmatizer()

    def __call__(self, doc):
        return [
            self.wnl.lemmatize(t)
            for t in doc.split()
            if len(t) >= 2 and re.match("[a-z].*", t) and re.match(token_pattern, t)
        ]

/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/nltk/metrics/associati
iPy (detected version 1.22.4)
  from scipy.stats import fisher_exact
[nltk_data] Downloading package wordnet to /home/ec2-user/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Let’s perform lemmatizing and counting next.

```
[11]: import time
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

print("Lemmatizing and counting, this may take a few minutes...")
start_time = time.time()
vectorizer = CountVectorizer(
    input="content",
    analyzer="word",
    stop_words="english",
    # tokenizer=LemmaTokenizer(),
    max_df=0.9,
    min_df=3,
)

train_vectors = vectorizer.fit_transform(train_doc_list)
val_vectors = vectorizer.transform(val_doc_list)
test_vectors = vectorizer.transform(test_doc_list)

vocab_list = vectorizer.get_feature_names_out()
vocab_size = len(vocab_list)
print("vocab size:", vocab_size)
print("Done. Time elapsed:: {:.2f}s".format(time.time() - start_time))

Lemmatizing and counting, this may take a few minutes...
vocab size: 20439
Done. Time elapsed:: 2.19s
```

Define train, test, and val to train the algorithm.

```
[13]: import scipy.sparse as sparse

def shuffle_and_dtype(vectors):
    idx = np.arange(vectors.shape[0])
    np.random.shuffle(idx)
    vectors = vectors[idx, :]
    vectors = sparse.csr_matrix(vectors, dtype=np.float32)
    print(type(vectors), vectors.dtype)
    return vectors

train_vectors = shuffle_and_dtype(train_vectors)
val_vectors = shuffle_and_dtype(val_vectors)
test_vectors = shuffle_and_dtype(test_vectors)
```

```
<class 'scipy.sparse._csr.csr_matrix'> float32
<class 'scipy.sparse._csr.csr_matrix'> float32
<class 'scipy.sparse._csr.csr_matrix'> float32
```

The NTM algorithm accepts data in RecordIO Protobuf format. Inside this helper functions we use write_spmatrix_to_sparse_tensor function provided by SageMaker to convert scipy spare matrix into RecordIO Protobuf format.

```
import io
import sagemaker.amazon.common as smac

def split_convert(sparray, prefix, fname_template="data_part{}.pbr", n_parts=2):
    chunk_size = sparray.shape[0] // n_parts
    for i in range(n_parts):

        # Calculate start and end indices
        start = i * chunk_size
        end = (i + 1) * chunk_size
        if i + 1 == n_parts:
            end = sparray.shape[0]

        # Convert to record protobuf
        buf = io.BytesIO()
        smac.write_spmatrix_to_sparse_tensor(array=sparray[start:end], file=buf, labels=None)
        buf.seek(0)

        fname = os.path.join(prefix, fname_template.format(i))
        with open(fname, "wb") as f:
            f.write(buf.getvalue())
        print("Saved data to {}".format(fname))

train_data_dir = os.path.join(data_dir, "train")
val_data_dir = os.path.join(data_dir, "validation")
test_data_dir = os.path.join(data_dir, "test")

check_create_dir(train_data_dir)
check_create_dir(val_data_dir)
check_create_dir(test_data_dir)

split_convert(train_vectors, prefix=train_data_dir, fname_template="train_part{}.pbr", n_parts=4)
split_convert(val_vectors, prefix=val_data_dir, fname_template="val_part{}.pbr", n_parts=1)
split_convert(test_vectors, prefix=test_data_dir, fname_template="test_part{}.pbr", n_parts=1)
```

```
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
Saved data to /home/ec2-user/SageMaker/wikitext-2/train/train_part0.pbr
Saved data to /home/ec2-user/SageMaker/wikitext-2/train/train_part1.pbr
Saved data to /home/ec2-user/SageMaker/wikitext-2/train/train_part2.pbr
Saved data to /home/ec2-user/SageMaker/wikitext-2/train/train_part3.pbr
Saved data to /home/ec2-user/SageMaker/wikitext-2/validation/val_part0.pbr
Saved data to /home/ec2-user/SageMaker/wikitext-2/test/test_part0.pbr
```

Let’s save the text file with the name vocab.txt in the auxiliary directory.

```
In [36]: aux_data_dir = os.path.join(data_dir, "auxiliary")
         check_create_dir(aux_data_dir)
         with open(os.path.join(aux_data_dir, "vocab.txt"), "w", encoding="utf-8") as f:
             for item in vocab_list:
                 f.write(item + "\n")
```

3. Store Data on S3

Specify data locations and access roles. The S3 bucket and prefix that you want to use for training and model data. The IAM role is used to give training and hosting access to your data.

In [37]:
```python
import os
import sagemaker

role = sagemaker.get_execution_role()

bucket = sagemaker.Session().default_bucket()  # <or insert your own bucket name>#
prefix = "ntm/" + dataset

train_prefix = os.path.join(prefix, "train")
val_prefix = os.path.join(prefix, "val")
aux_prefix = os.path.join(prefix, "auxiliary")
test_prefix = os.path.join(prefix, "test")
output_prefix = os.path.join(prefix, "output")

s3_train_data = os.path.join("s3://", bucket, train_prefix)
s3_val_data = os.path.join("s3://", bucket, val_prefix)
s3_aux_data = os.path.join("s3://", bucket, aux_prefix)
s3_test_data = os.path.join("s3://", bucket, test_prefix)
output_path = os.path.join("s3://", bucket, output_prefix)
print("Training set location", s3_train_data)
print("Validation set location", s3_val_data)
print("Auxiliary data location", s3_aux_data)
print("Test data location", s3_test_data)
print("Trained model will be saved at", output_path)
```

```
Training set location s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/train
Validation set location s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/val
Auxiliary data location s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/auxiliary
Test data location s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/test
Trained model will be saved at s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/output
```

Upload the input directories to S3

In [38]:
```python
import subprocess

cmd_train = "aws s3 cp " + train_data_dir + " " + s3_train_data + " --recursive"
p = subprocess.Popen(cmd_train, shell=True, stdout=subprocess.PIPE)
p.communicate()
```

Out[38]:
```
(b'Completed 256.0 KiB/2.3 MiB (1.3 MiB/s) with 4 file(s) remaining\rCompleted 512.0 KiB/2.3 MiB (2.5 MiB/s) with 4 f
ile(s) remaining\rCompleted 768.0 KiB/2.3 MiB (3.5 MiB/s) with 4 file(s) remaining\rCompleted 1.0 MiB/2.3 MiB (4.7 Mi
B/s) with 4 file(s) remaining  \rCompleted 1.2 MiB/2.3 MiB (5.8 MiB/s) with 4 file(s) remaining  \rCompleted 1.5 MiB/
2.3 MiB (6.9 MiB/s) with 4 file(s) remaining  \rCompleted 1.8 MiB/2.3 MiB (7.8 MiB/s) with 4 file(s) remaining  \rCom
pleted 2.0 MiB/2.3 MiB (8.8 MiB/s) with 4 file(s) remaining  \rCompleted 2.1 MiB/2.3 MiB (7.7 MiB/s) with 4 file(s) r
emaining  \rupload: train/train_part3.pbr to s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/train/train_part3.p
br\nCompleted 2.1 MiB/2.3 MiB (7.7 MiB/s) with 3 file(s) remaining\rCompleted 2.1 MiB/2.3 MiB (7.3 MiB/s) with 3 file
(s) remaining\rupload: train/train_part1.pbr to s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/train/train_part
1.pbr\nCompleted 2.1 MiB/2.3 MiB (7.3 MiB/s) with 2 file(s) remaining\rCompleted 2.2 MiB/2.3 MiB (7.1 MiB/s) with 2 f
ile(s) remaining\rupload: train/train_part2.pbr to s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/train/train_p
art2.pbr\nCompleted 2.2 MiB/2.3 MiB (7.1 MiB/s) with 1 file(s) remaining\rCompleted 2.3 MiB/2.3 MiB (7.1 MiB/s) with
1 file(s) remaining\rupload: train/train_part0.pbr to s3://sagemaker-us-east-2-931175847565/ntm/wikitext-2/train/trai
n_part0.pbr\n',
 None)
```

In [39]:
```python
cmd_val = "aws s3 cp " + val_data_dir + " " + s3_val_data + " --recursive"
p = subprocess.Popen(cmd_val, shell=True, stdout=subprocess.PIPE)
p.communicate()
```

Out[39]:
```
(b'Completed 238.8 KiB/238.8 KiB (1.2 MiB/s) with 1 file(s) remaining\rupload: validation/val_part0.pbr to s3://sagem
aker-us-east-2-931175847565/ntm/wikitext-2/val/val_part0.pbr\n',
 None)
```

In [40]:
```python
cmd_test = "aws s3 cp " + test_data_dir + " " + s3_test_data + " --recursive"
p = subprocess.Popen(cmd_test, shell=True, stdout=subprocess.PIPE)
p.communicate()
```

Out[40]:
```
(b'Completed 247.9 KiB/247.9 KiB (1.4 MiB/s) with 1 file(s) remaining\rupload: test/test_part0.pbr to s3://sagemaker-
us-east-2-931175847565/ntm/wikitext-2/test/test_part0.pbr\n',
 None)
```

In [41]:
```python
cmd_aux = "aws s3 cp " + aux_data_dir + " " + s3_aux_data + " --recursive"
p = subprocess.Popen(cmd_aux, shell=True, stdout=subprocess.PIPE)
p.communicate()
```

Out[41]:
```
(b'Completed 164.1 KiB/164.1 KiB (1.3 MiB/s) with 1 file(s) remaining\rupload: auxiliary/vocab.txt to s3://sagemaker-
us-east-2-931175847565/ntm/wikitext-2/auxiliary/vocab.txt\n',
 None)
```

4. Model Training

Let's configure a SageMaker training job to use the NTM algorithm on the data we prepared. SageMaker uses Amazon Elastic Container Registry (ECR) docker container to host the NTM training image. ** in below screen show(instance_type = 'ml.c4.xlarge')

```
[28]: import boto3
      from sagemaker.image_uris import retrieve

      container = retrieve("ntm", boto3.Session().region_name)
```

```
[31]: sess = sagemaker.Session()
      ntm = sagemaker.estimator.Estimator(
          container,
          role,
          instance_count=1,
          instance_type="ml.c4.xlarge",
          output_path=output_path,
          sagemaker_session=sess,
      )
```

```
[32]: num_topics = 20
      ntm.set_hyperparameters(
          num_topics=num_topics, feature_dim=vocab_size, mini_batch_size=60, epochs=50, sub_sample=0.7
      )
```

```
[34]: from sagemaker.inputs import TrainingInput

      s3_train = TrainingInput(
          s3_train_data, distribution="ShardedByS3Key", content_type="application/x-recordio-protobuf"
      )
      s3_val = TrainingInput(
          s3_val_data, distribution="FullyReplicated", content_type="application/x-recordio-protobuf"
      )
      s3_test = TrainingInput(
          s3_test_data, distribution="FullyReplicated", content_type="application/x-recordio-protobuf"
      )
      s3_aux = TrainingInput(s3_aux_data, distribution="FullyReplicated", content_type="text/plain")
```

Now, it is ready to run the training job. Again, we will notice in the log that the top words are printed together with the WETC and TU scores.

```
[24]: ntm.fit({
    "train": s3_train,
    "validation": s3_val,
    "auxiliary": s3_aux,
    "test": s3_test,
})
```

```
INFO:sagemaker:Creating training-job with name: ntm-2024-09-23-07-50-16-271
2024-09-23 07:50:17 Starting - Starting the training job...
2024-09-23 07:50:32 Starting - Preparing the instances for training...
2024-09-23 07:51:03 Downloading - Downloading input data...
2024-09-23 07:51:23 Downloading - Downloading the training image.......................
2024-09-23 07:55:46 Training - Training image download completed. Training in progress...Docker entrypoint called with argument(s): train
Running default environment configuration script
/opt/amazon/lib/python3.8/site-packages/mxnet/model.py:97: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if num_device is 1 and 'dist' not in kvstore:
[09/23/2024 07:55:57 INFO 140718653077312] Reading default configuration from /opt/amazon/lib/python3.8/site-packages/algorithm/default-in
atch_size': '256', 'epochs': '50', 'encoder_layers_activation': 'sigmoid', 'optimizer': 'adadelta', 'tolerance': '0.001', 'num_patience_ep
gradient': '1.0', 'clip_gradient': 'Inf', 'weight_decay': '0.0', 'learning_rate': '0.01', 'sub_sample': '1.0', '_tuning_objective_metric':
'auto', '_num_kv_servers': 'auto', '_kvstore': 'auto_gpu'}
um": 1177.0391464233398, "count": 52, "min": 20.028114318847656, "ma
ax": 40.000200271606445}, "update.time": {"sum": 23881.894826889038,
ount": 1, "min": 121.92082405090332, "max": 121.92082405090332}, "mo
"setuptime": {"sum": 34.88755226135254, "count": 1, "min": 34.887552
6, "max": 34610.774517059326}}}

2024-09-23 07:56:49 Uploading - Uploading generated training model
2024-09-23 07:56:49 Completed - Training job completed
Training seconds: 346
Billable seconds: 346
```

Once the job is completed, you can view information about and the status of a training job using the AWS SageMaker console.

```
print("Training job name: {}".format(ntm.latest_training_job.job_name))
```

```
Training job name: ntm-2024-09-23-07-50-16-271
```

HOS submission instructions:

1. Please install the GitHub Desktop: https://cityuseattle.github.io/docs/git/github_desktop/

2. Clone, organize, and submit your work through GitHub Desktop: https://cityuseattle.github.io/docs/hoporhos