**CHAPTER 10:** Create Machine Learning Inference Pipelines with Amazon SageMaker

- 23 pages

**Introduction**

Data scientists and Machine Learning Engineers spend significant amount of time conducting different steps – feature engineering, feature processing, training & deploying models, and obtaining inferences on data -- in Machine Learning Life Cycle. The data transformation logic used to process data during model training is same as the logic used to prepare data for obtaining inferences. Repeating the data transformation steps manually at the time of inference is redundant and not effective. Additionally, at inference time, the data processing logic has to be either coupled with inference logic or client application sending requests to trained models for scoring. The outcome is that your ability to iterate quickly becomes limited. If the data processing logic can be pulled out of the client application and inference logic, you can manage it independently, decreasing the complexity of the implementation.

The goal of this chapter is to walk you through how SageMaker and other AWS services can be employed to create Machine Learning Pipelines that can process big data, train algorithms, deploy trained models, and run inferences, all while using the same logic for data processing during model training and inference.

Let’s look at some of the AWS Services and SageMaker Python SDK (a library used for training and deploying machine learning models on SageMaker) required to build the pipeline. Amazon Glue is a fully-managed serverless ETL (Extract, Transform, and Load) service used to wrangle big data. The ETL jobs are run on Apache Spark environment, where Glue handles provisioning, configuration, and scaling of the resources required to run the jobs. In this chapter, we will write the ETL logic in Python and run PySpark jobs via Glue.

Also, we will use a component called MLeap, which is an open source Spark package designed to serialize Spark-trained transformers. The serialized models are used to transform data at the time of inference.

SageMaker Python SDK offers classes, such as Model, SparkMLModel, and PipelineModel, to create the Machine Learning Pipeline. The class Model is used to represent built-in SageMaker algorithms, whereas SparkMLModel is used for serialized SparkML (data transformer) jobs. PipelineModel class is leveraged to sequence the execution of data transformation and scoring.

To illustrate the concepts, we will use [ABC Millions Headlines](https://www.kaggle.com/therohk/million-headlines) dataset. The dataset contains approximately a million news headlines. Our main idea is to process the headlines using PySpark and train Neural Topic Model on the processed data to identify topics. The trained data transformer job and NTM model can be deployed as PipelineModel both as an endpoint for real-time predictions and in batch transform mode for batch predictions.

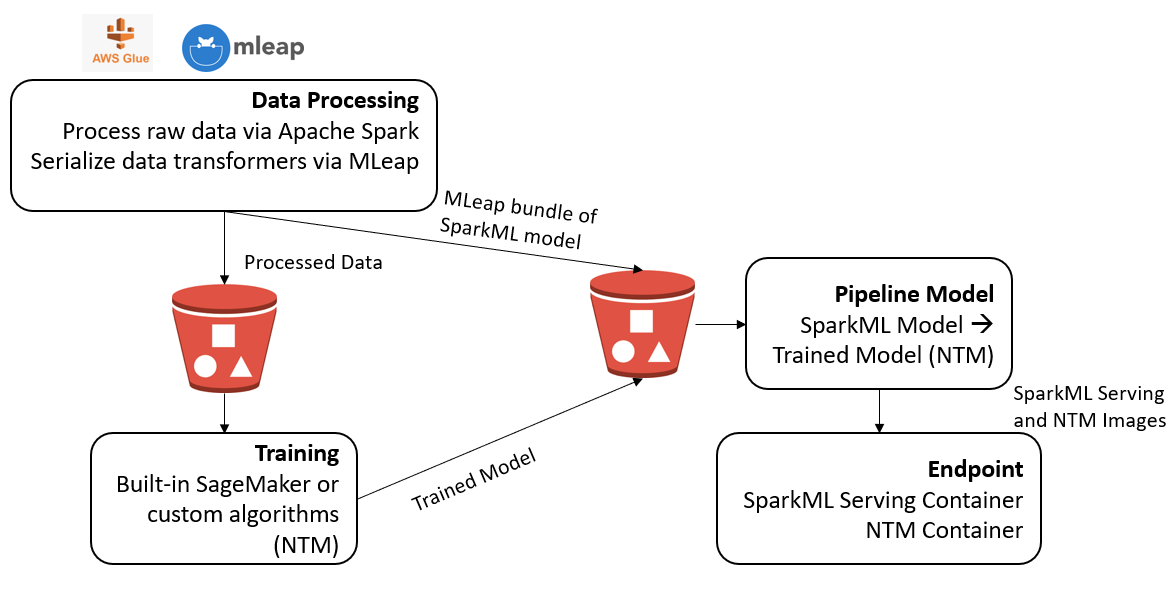
We will go through the below sections to illustrate key ideas:

* Create processed dataset using Amazon Glue ETL service to run SparkML jobs
* Identify topics in the processed dataset via training NTM (Neural Topic Model) algorithm
* Create inference pipeline consisting of SparkML and NTM models for real time predictions
* Create inference pipeline consisting of SparkML and NTM models for batch predictions

**Architecture of Inference Pipeline in SageMaker**

In the first step of the pipeline, we execute data processing logic on Apache Spark via AWS Glue. The data processing logic includes creating tokens from each of the news headlines, removing stop words, and counting frequency of each of the words in a given headline. The ETL logic is serialized into MLeap bundle, which can be used at the time of inference for data processing. Both the serialized SparkML model and processed input data are stored on S3 bucket. In second step, the NTM (Neural Topic Model) algorithm is trained on the processed data to discover topics. In step three, both the SparkML and trained NTM models are used to create a Pipeline Model, which is used to execute the models in the specified sequence. SparkML serving docker container and NTM docker container are provisioned as an endpoint for real-time model predictions. The same Pipeline Model can be used to run inferences in batch mode – i.e. score multiple news headlines in one go, discovering topics for each of them.

The following figure summarizes the steps we will implement:



**Feature Processing with Amazon Glue and SparkML in SageMaker**

The data processing via SparkML logic in encapsulated in a python script. This script is passed as input to AWS Glue job. Let’s review the script briefly.

The **getResolvedOptions** (args, options=argument names that you want to retrieve) gives you access to the arguments that are passed to the SparkML script when running a job. We will retrieve the location of input data (S3 bucket), in addition to noting the location of the job output, along with the location of the trained SparkML data transformers.

def main():

spark = SparkSession.builder.appName("AbcHeadlinesSpark").getOrCreate()

args = getResolvedOptions(sys.argv, ['S3\_INPUT\_BUCKET',

'S3\_INPUT\_KEY\_PREFIX',

'S3\_INPUT\_FILENAME',

'S3\_OUTPUT\_BUCKET',

'S3\_OUTPUT\_KEY\_PREFIX',

'S3\_MODEL\_BUCKET',

'S3\_MODEL\_KEY\_PREFIX'])

We will filter the ABC News Headlines dataset to include only 10% of the million headlines. This is to arrive at a manageable size for the dataset.

#Read the compressed text file containing enron emails encoded as table containing docID, wordID, and count

abcnewsdf = spark.read.option("header","true").csv(('s3://' + os.path.join(args['S3\_INPUT\_BUCKET'], args['S3\_INPUT\_KEY\_PREFIX'], args['S3\_INPUT\_FILENAME'])))

#Filter number of abc news headlines

#1,103,663 - headlines

hdl\_cnt = abcnewsdf.count()

#Filter the number of headlines

hdl\_fil\_cnt = hdl\_cnt \* .1

hdl\_fil\_cnt = int(hdl\_fil\_cnt)

abcnewsdf = abcnewsdf.limit(hdl\_fil\_cnt)

The filtered dataset is passed through a few feature transformers. Transfomers, in Spark, is an abstraction that includes features transformers and learned models. A transformer converts one DataFrame into another, generally by appending one or more columns. Tokenizer transformer takes headline as input and adds a column called “words” to the original dataset. Similarly, StopWordsRemover transformer removes stop words and adds a column called “filtered” to the input dataset. Also, CountVectorizer and IDF transformers create Term Frequency-Inverse document frequency column called “features” to the input dataset. In other words, the number of times a word occurs in a headline is weighted by its frequency of occurrence across the entire headline collection. A word that occurs more frequently as part of a headline, but occurs less frequently across all the headlines is indicative of a topic.

#Create features from text

#Tokenizer

tok = Tokenizer(inputCol="headline\_text", outputCol="words")

# stop words

swr = StopWordsRemover(inputCol="words", outputCol="filtered")

# Term frequency

ctv = CountVectorizer(inputCol="filtered", outputCol="tf", vocabSize=200, minDF=2)

#Term frequency is weighted by number of times the word appears across all docs in corpus

# Words that are unique to a headline have more weight - since they define the headline

idf = IDF(inputCol="tf", outputCol="features")

A pipeline is built with each of the transformers in a specific order. Once the pipeline is fit to the input dataset, Pipeline Model is created. The resulting Pipeline Model is used to create the final processed dataset.

# Build the pipeline

news\_pl = Pipeline(stages=[tok, swr, ctv, idf])

#Transformed dataset

news\_pl\_fit = news\_pl.fit(abcnewsdf)

news\_ftrs\_df = news\_pl\_fit.transform(abcnewsdf)

To save the processed headlines in csv format, the “features” column needs to be in a simple string format. CSV file format does not support storing arrays or lists in a column. We will define a user defined function, get\_str to convert dense vector to a string of comma separated tf-idf numbers. The length of features in CountVectorizer defines the number of words (vocabulary) used to create a representation of a headline.

def gen\_str(row\_arr):

return (','.join([str(elem) for elem in row\_arr.toArray()]))

gen\_str\_udf = F.udf(gen\_str, StringType())

#Convert Sparse vector to Dense vector

news\_formatted = news\_ftrs\_df.withColumn('result', gen\_str\_udf(news\_ftrs\_df.features))

#Save the Dense vector to csv file

news\_save = news\_formatted.select("result")

news\_save.write.option("delimiter", "\t").mode("append").csv('s3://' + os.path.join(args['S3\_OUTPUT\_BUCKET'], args['S3\_OUTPUT\_KEY\_PREFIX']))

Also, we will need to save vocabulary, words used to represent each headline, to a text file. The text file is used as an input to NTM algorithm. The words in the vocabulary are used to label the latent topics discovered from headlines.

#Save the vocabulary file

vocab\_list = news\_pl\_fit.stages[2].vocabulary

vocab\_df = spark.createDataFrame(vocab\_list, StringType())

vocab\_df = vocab\_df.coalesce(1)

vocab\_df.write.option("delimiter", "\n").format("text").mode("append").save('s3://' + os.path.join(args['S3\_OUTPUT\_BUCKET'], args['S3\_OUTPUT\_KEY\_PREFIX']))

The Pipeline Model is serialized via MLeap. The MLeap bundle, which is in .zip format, is converted to .tar.gz format and uploaded to S3 bucket. SageMaker requires SparkML models to be in .tar.gz format.

# Serialize the tokenizer via MLeap and upload to S3

SimpleSparkSerializer().serializeToBundle(news\_pl\_fit, "jar:file:/tmp/model.zip", news\_ftrs\_df)

# Unzip as SageMaker expects a .tar.gz file but MLeap produces a .zip file.

import zipfile

with zipfile.ZipFile("/tmp/model.zip") as zf:

zf.extractall("/tmp/model")

# Write back the content as a .tar.gz file

import tarfile

with tarfile.open("/tmp/model.tar.gz", "w:gz") as tar:

tar.add("/tmp/model/bundle.json", arcname='bundle.json')

tar.add("/tmp/model/root", arcname='root')

s3 = boto3.resource('s3')

file\_name = os.path.join(args['S3\_MODEL\_KEY\_PREFIX'], 'model.tar.gz')

s3.Bucket(args['S3\_MODEL\_BUCKET']).upload\_file('/tmp/model.tar.gz', file\_name)

if \_\_name\_\_ == "\_\_main\_\_":

main()

*SparkML Model Serialization*

We will need MLeap java package and python wrapper of MLeap readily available for AWS Glue job. These are the pre-requisites for serializing SparkML model into MLeap bundle.

For the serialization to work consistently, the required libraries are packaged into an assembly jar. Because MLeap is a Spark package, we need a python wrapper to invoke MLeap serialization as part of the data pre-processing logic (Python) executed by Glue job.

Upload the MLeap Spark Assembly and corresponding Python wrapper to the designated S3 location

python\_dep\_location = sess.upload\_data(path='python.zip', bucket=default\_bucket, key\_prefix='sagemaker/inference-pipeline/dependencies/python')

jar\_dep\_location = sess.upload\_data(path='mleap\_spark\_assembly.jar', bucket=default\_bucket, key\_prefix='sagemaker/inference-pipeline/dependencies/jar')

*Designate the output location for processed data and SparkML model*

from time import gmtime, strftime

import time

timestamp\_prefix = strftime("%Y-%m-%d-%H-%M-%S", gmtime())

# Input location of the data, We uploaded our train.csv file to input key previously

s3\_input\_bucket = default\_bucket

s3\_input\_key\_prefix = 'sagemaker/inference-pipeline/input'

s3\_input\_fn = 'abcnews-date-text.csv'

s3\_model\_key\_prefix = s3\_output\_key\_prefix + '/mleap'

# Output location of the data. The input data will be split, transformed, and

# uploaded to output/train and output/validation

s3\_output\_bucket = default\_bucket

s3\_output\_key\_prefix = 'sagemaker/inference-pipeline/output/' + timestamp\_prefix

# the MLeap serialized SparkML model will be uploaded to output/mleap

s3\_model\_bucket = default\_bucket

s3\_model\_key\_prefix = s3\_output\_key\_prefix + '/mleap'

*Create Glue Job*

We will now create Glue job using Boto, which is AWS SDK for Python. The SDK enables Python developers to create, configure, and manage AWS services. We tell boto session that we want to use *Glue* service. Once we’ve the *Glue* resource, we can create a ETL job by passing job name, description, IAM role, number of concurrent runs, script location of data processing logic, language in which data processing logic has been written, location of jars and python files required for execution.

boto\_session = sess.boto\_session

s3 = boto\_session.resource('s3')

glue\_client = boto\_session.client('glue')

job\_name = 'sparkml-abcnews-' + timestamp\_prefix

response = glue\_client.create\_job(

Name=job\_name,

Description='PySpark job to featurize the Enron Emails dataset',

Role=role, # you can pass your existing AWS Glue role here if you have used Glue before

ExecutionProperty={

'MaxConcurrentRuns': 1

},

Command={

'Name': 'glueetl',

'ScriptLocation': script\_location

},

DefaultArguments={

'--job-language': 'python',

'--extra-jars' : jar\_dep\_location,

'--extra-py-files': python\_dep\_location

},

AllocatedCapacity=10,

Timeout=60,

)

glue\_job\_name = response['Name']

print(glue\_job\_name)

*Start Glue Job*

Now that we have the job configured, we will start the run by passing input arguments: S3 input bucket, input file name containing headlines, output bucket, location to save MLeap bundle.

job\_run\_id = glue\_client.start\_job\_run(JobName=job\_name,

Arguments = {

'--S3\_INPUT\_BUCKET': s3\_input\_bucket,

'--S3\_INPUT\_KEY\_PREFIX': s3\_input\_key\_prefix,

'--S3\_INPUT\_FILENAME': s3\_input\_fn,

'--S3\_OUTPUT\_BUCKET': s3\_output\_bucket,

'--S3\_OUTPUT\_KEY\_PREFIX': s3\_output\_key\_prefix,

'--S3\_MODEL\_BUCKET': s3\_model\_bucket,

'--S3\_MODEL\_KEY\_PREFIX': s3\_model\_key\_prefix

})['JobRunId']

print(job\_run\_id)

*Check Glue Job Status*

While the ETL job is running, we can check the status of it. Listed below is the AWS Glue console after the job is executed.

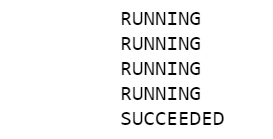
job\_run\_status = glue\_client.get\_job\_run(JobName=job\_name,RunId=job\_run\_id)['JobRun']['JobRunState']

while job\_run\_status not in ('FAILED', 'SUCCEEDED', 'STOPPED'):

job\_run\_status = glue\_client.get\_job\_run(JobName=job\_name,RunId=job\_run\_id)['JobRun']['JobRunState']

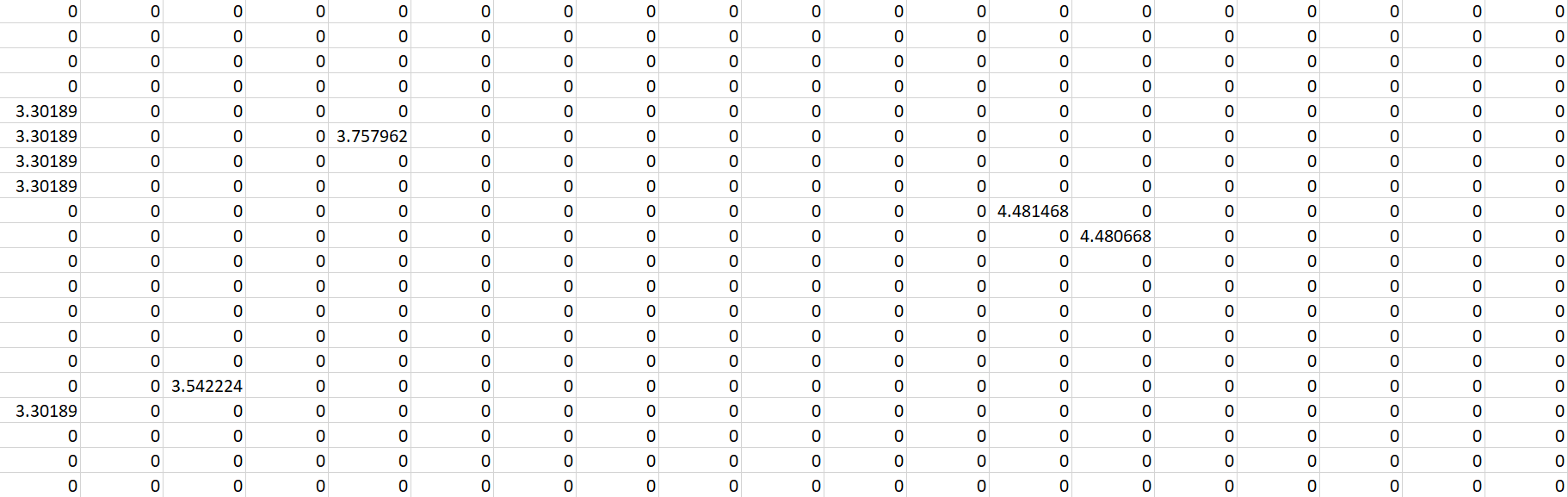
print (job\_run\_status)

time.sleep(30)



|  |  |
| --- | --- |
|  |  |
|  |  |

*Processed Data – Matrix of headlines and words*



*Processed Data – List of words in Vocabulary*

A few words from the generated vocabulary file:

|  |  |  |
| --- | --- | --- |
| police  us  govt  new  man  council  iraq  plan  court  says  nsw  claims  fire  killed | report  may  call  qld  water  back  urged  wa  boost  health  win  probe  face  death | hospital  crash  group  cup  set  two  world  pm  sydney  australia  war  vic  calls |

**Identify Topics by Training NTM Model using SageMaker**

Read the processed Abc News Headlines dataset from the output folder on the designated S3 bucket. This dataset is the only csv file in the output folder.

#Read the processed dataset

s3 = boto3.resource('s3')

my\_bucket = s3.Bucket(default\_bucket)

files = my\_bucket.objects.filter(Prefix=s3\_output\_key\_prefix)

for f in files:

if '.csv' in f.key:

#print(f.key)

abcnews\_df = pd.read\_csv(os.path.join('s3://', s3\_output\_bucket, f.key))

We will convert the dense vector representation of headlines to compressed sparse row matrix (CSR) format.

#convert dataframe (dense vector) to compressed sparse row matrix

abcnews\_csr = csr\_matrix(abcnews\_df, dtype=np.float32)

print(abcnews\_csr[:16].toarray())

We will go through the same NTM training process detailed in the chapter *Discovering Topics in Text Collection using Amazon SageMaker*

Refer to the notebook associated with this chapter for details. We will upload training and validation vectors to the designated location on S3 bucket

#Upload training and validation vectors

split\_convert\_upload(train\_vectors, bucket=default\_bucket, prefix=train\_prefix, fname\_template='data\_part{}.pbr', n\_parts=8)

split\_convert\_upload(val\_vectors, bucket=default\_bucket, prefix=val\_prefix, fname\_template='val\_part{}.pbr', n\_parts=3)

We will now download the vocabulary file created by SparkML job, rename it, and upload to the same to the auxiliary path location.

#Download the vocabulary file from s3 bucket

s3 = boto3.resource('s3')

files = my\_bucket.objects.filter(Prefix=s3\_output\_key\_prefix)

for f in files:

if '.txt' in f.key:

s3.Bucket(default\_bucket).download\_file(f.key, 'vocab.txt')

#Upload vocabulary file to auxiliary path

vocabFN\_location = sess.upload\_data(path='vocab.txt', bucket=default\_bucket, key\_prefix=aux\_prefix)

We will train the built-in NTM algorithm on the processed dataset.

s3\_train = s3\_input(s3\_train\_data, distribution='ShardedByS3Key')

s3\_aux = s3\_input(s3\_aux\_data, distribution='FullyReplicated', content\_type='text/plain')

ntm.fit({'train': s3\_train, 'test': s3\_val\_data, 'auxiliary': s3\_aux})

From training, below are the 5 topics uncovered from the headlines.

War and Crime

[0.56, 0.93] dies charged found dead man crash woman murder accident missing search three car killed attack injured charge death four baghdad

Legal Challenges and Sports

[0.61, 0.82] lead final wins cup takes test win first england world open face title take second top court day one charges

Clashes and Conflict

[0.49, 0.78] back set pakistan residents win appeal top clash home england rain world killed title crash water return bid south record

Crime and Investigation (Public Safety)

[0.32, 0.88] deal probe wa new death record denies power says australian police open fire council home hits rejects face qld vic

International Politics and Conflict

[0.49, 1.00] un pm report war calls plans australia iraq warns inquiry troops urged election tour us iraqi may bush defends claim

The variational autoencoder implemented as part of NTM is optimized by minimizing the loss in a) reconstructing the original document from topic embeddings (recons) and b) building a stochastic representation of the input document (aka Kullback Leibler Divergence - kld), which is nothing but topic embeddings.

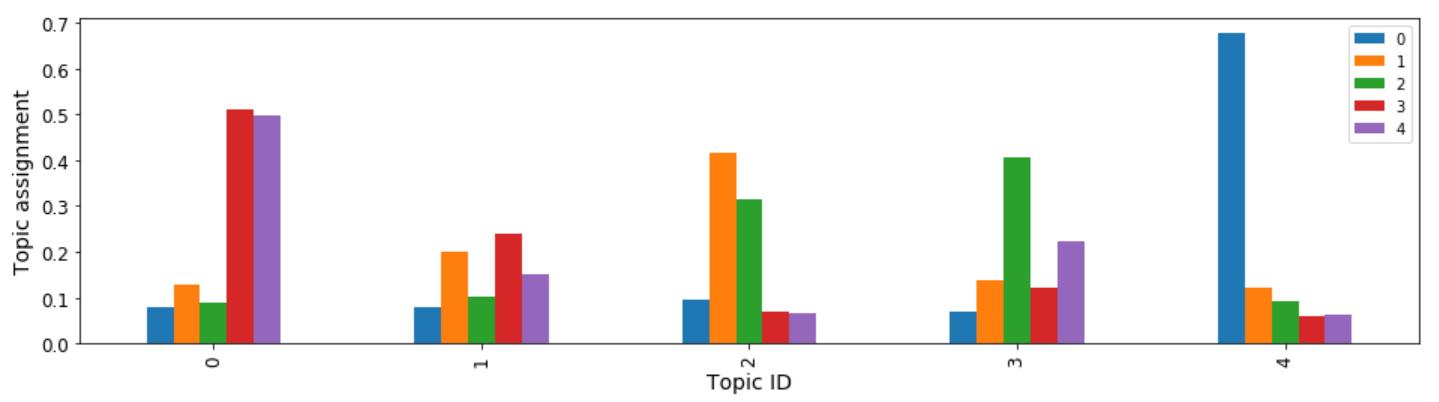
[02/17/2019 23:30:41 INFO 140262573676352] Loss (name: value) total: 2.93362134402

[02/17/2019 23:30:41 INFO 140262573676352] Loss (name: value) kld (representation of input documents): 0.627517072295

[02/17/2019 23:30:41 INFO 140262573676352] Loss (name: value) recons (reconstruction of original document from topic embedding): 2.30610427302

[02/17/2019 23:30:41 INFO 140262573676352] Loss (name: value) logppx: 2.93362134402

The below diagram showcases how each of the 5 topics are represented across 5 headlines selected from test dataset (refer to the notebook for further details). For example, Topic 0 is prevalent in headlines 4 and 5



**Create Inference Pipeline with SparkML and NTM models for Real-Time Predictions**

In this section, we will build a pipeline where we re-use the serialized SparkML model for data pre-processing and employ trained NTM model to derive topics from pre-processed headlines. SageMaker Python SDK provides classes, such Model, SparkMLModel, & PipelineModel, to create an inference pipeline that can be used to conduct feature processing and then score the processed data using trained algorithm. The PipelineModel created can be deployed as an endpoint for real time inferences. Additionally, the PipelineModel can also be deployed in batch mode (Batch Transform), to get inferences for large volume of data points.

SparkMLModel class, which extends org.apache.spark.ml.Model, is used to represent SparkML data transformers. Model, on the other hand, is for representing trained NTM model. PipelineModel further uses these two models as stages in the pipeline to conduct data pre-processing and inference.

The SparkMLModel requires structure of the input and processed output dataset.

import json

schema = {

"input": [

{

"name": "headline\_text",

"type": "string"

}

],

"output":

{

"name": "features",

"type": "double",

"struct": "vector"

}

}

schema\_json = json.dumps(schema)

print(schema\_json)

*Real-time Predictions*

In this section, we will deploy the SageMaker PipelineModel as an endpoint. Our first step is to initialize SparkMLModel by passing S3 location holder for MLeap serialized SparkML model, along with schema of input and output. As our second step, we will create NTM model from the latest NTM docker image and the trained NTM model.

#Get MLeap serialized model

s3\_ntm\_output\_key\_prefix = 'sagemaker/inference-pipeline/output'

#Get the data location of the trained ntm model

#modeldataurl = 's3://{}/{}/{}/{}'.format(s3\_model\_bucket, s3\_ntm\_output\_key\_prefix, ntm.latest\_training\_job.job\_name, 'output/model.tar.gz')

sparkml\_data = 's3://{}/{}/{}'.format(s3\_model\_bucket, s3\_model\_key\_prefix, 'model.tar.gz')

ntm\_model = Model(model\_data=modeldataurl, image=container)

# passing the schema defined above by using an environment variable that sagemaker-sparkml-serving understands

sparkml\_model = SparkMLModel(model\_data=sparkml\_data, env={'SAGEMAKER\_SPARKML\_SCHEMA' : schema\_json})

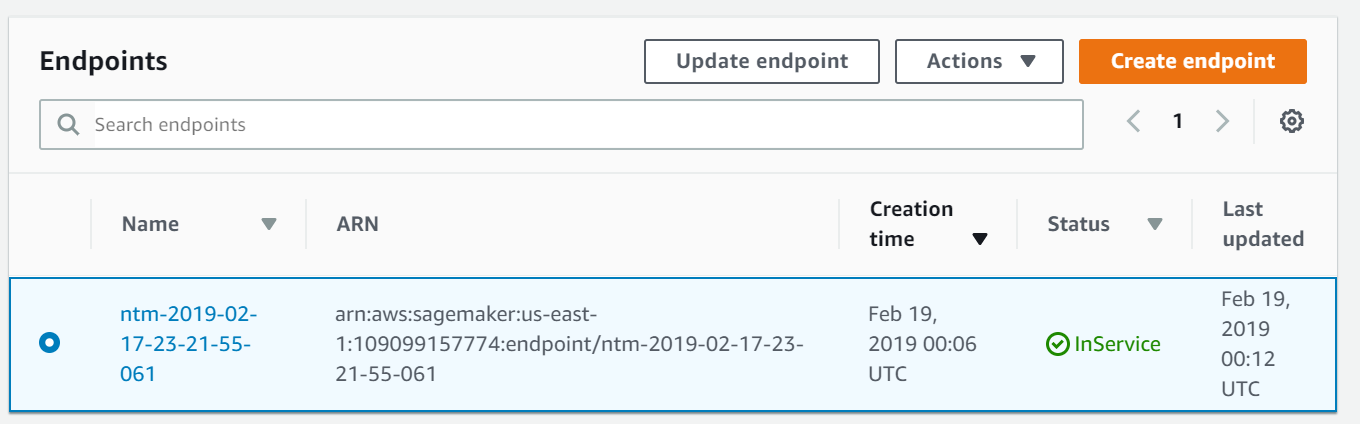
model\_name = 'inference-pipeline-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

sm\_model = PipelineModel(name=model\_name, role=role, models=[sparkml\_model, ntm\_model])

As our final step, we will deploy the PipelineModel as an endpoint.

endpoint\_name = 'inference-pipeline-ep-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

sm\_model.deploy(initial\_instance\_count=1, instance\_type='ml.c4.xlarge', endpoint\_name=endpoint\_name)



Now that the PipelineModel is deployed, it is time to pass data to the inference endpoint. We can choose to overwrite the input and output schema passed at the time of initializing SparkMLModel. If we choose to override, we can pass the schema along with the data that needs to be scored.

We will use JSON format (instead of CSV format) to pass input to the Pipeline model.

payload = {

"schema": {

"input": [

{

"name": "headline\_text",

"type": "string"

},

],

"output":

{

"name": "features",

"type": "double",

"struct": "vector"

}

},

"data": [

#["murder conviction court of criminal appeal"]

#["is dabiq captured opposition forces"]

["lisa scaffidi public hearing possible over expenses scandal"]

]

}

We will now initialize *RealTimePredictor* by passing the newly created endpoint. This predictor can be used to score new set of headlines, which are not part of training, validation or test datasets.

predictor = RealTimePredictor(endpoint=endpoint\_name, sagemaker\_session=sess, serializer=json\_serializer,

content\_type=CONTENT\_TYPE\_JSON, accept=CONTENT\_TYPE\_CSV)

print(predictor.predict(payload))

Here are the topics weights for title: "lisa scaffidi public hearing possible over expenses scandal"



We can see that the headline has 3 prominent topics: International Politics and Conflict followed by Sports & Legal Challenges and War & Crime

*A little bit of context:* Lisa Scaffidi was the Lord Mayor of Perth, Western Australia. She was charged with inappropriate use of her position – failure to declare gifts and travel worth tens and thousands of dollars. Therefore, this headline aptly has a mixture of topics: International Politics and Conflict (38%) followed by Legal Challenges (27%) and then by War & Crime (20%)

*As a reminder, here are the topics discovered from trained NTM model*

War and Crime

[0.56, 0.93] dies charged found dead man crash woman murder accident missing search three car killed attack injured charge death four baghdad

Sports and Legal Challenges

[0.61, 0.82] lead final wins cup takes test win first england world open face title take second top court day one charges

Clashes and Conflict

[0.49, 0.78] back set pakistan residents win appeal top clash home england rain world killed title crash water return bid south record

Crime and Investigation (Public Safety)

[0.32, 0.88] deal probe wa new death record denies power says australian police open fire council home hits rejects face qld vic

International Politics and Conflict

[0.49, 1.00] un pm report war calls plans australia iraq warns inquiry troops urged election tour us iraqi may bush defends claim

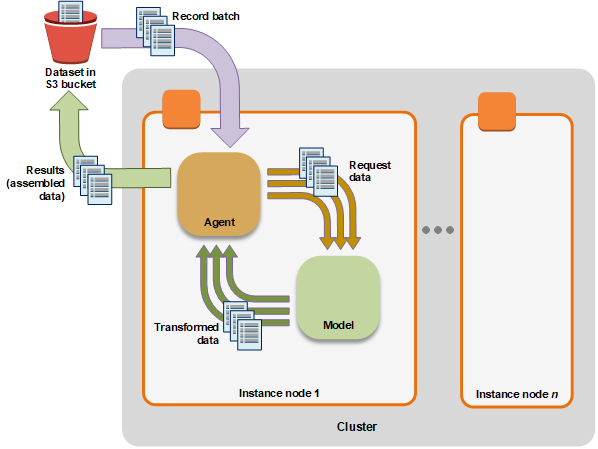
**Create Inference Pipeline with SparkML and NTM models for Batch Predictions**

In this section, we will turn our attention from real-time predictions to batch predictions. In real-world production scenarios, we typically come across two situations: 1) Obtain inferences in real time or in online mode and/or 2) Obtain inferences in batch or in offline mode.

To illustrate, in the case of using recommender system as part of a web/mobile app, real time inferences can be used when you want to personalize item suggestions based on in-app activity. The in-app activity, such as items you browsed, items left in shopping cart and not checked out, can be sent as input to an online recommender system. On the other hand, if you want to present item suggestions to your customers even before they engage with your web/mobile app, then you can send data related to their historical consumption behavior to an offline recommender system to obtain item suggestions for your entire customer base in one shot.

To address the need of deploying trained models in offline mode, SageMaker offers *Batch Transform.* Batch Transform is a newly released high performance and through-put feature where inferences can be obtained for the entire dataset. Both the input and output data are stored in S3 bucket. The *Batch Transform* service manages compute resources necessary to score the input data, given the trained model.

Below is architecture diagram of how *Batch Transform* service works. The service ingests large volumes of input data (from S3 bucket), transforms it, and generates inferences. The inferences produced are deposited back to the designated S3 bucket.



*Source: AWS Documentation (*[*Link*](https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-batch.html)*)*

In our instance, the trained model is a Pipeline Model encompassing both SparkML model (data transformer) and trained NTM model

The batch transform job needs the following information:

* The path to the S3 bucket where the input data is stored
* The path to the S3 bucket where the output data is stored
* The names of the trained Pipeline Model and of the Batch Transform job
* The compute resources required to run the Pipeline Model. For example, EC2 instance type and number
* The input and output data format – CSV, JSON

*Transformer* class from SageMaker.transformer is initialized with the parameters defined above. We then call *transform* method of *Transformer* object by passing location of the input dataset and the input format.

#Number of headlines - choose 10-15

input\_data\_path = 's3://{}/{}/{}'.format(default\_bucket, 'sagemaker/inference-pipeline/batch', 'abcnews-batch-input.csv')

output\_data\_path = 's3://{}/{}/{}'.format(default\_bucket, 'sagemaker/inference-pipeline/batch\_output/abcnews', timestamp\_prefix)

job\_name = 'serial-inference-batch-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())

#Define the SageMaker PipelineModel Name

model\_name = 'inference-pipeline-2019-02-22-01-15-00'

transformer = sagemaker.transformer.Transformer(

# This was the model created using PipelineModel and it contains feature processing and NTM stages

model\_name = model\_name,

instance\_count = 1,

instance\_type = 'ml.m4.xlarge',

strategy = 'SingleRecord',

assemble\_with = 'Line',

output\_path = output\_data\_path,

base\_transform\_job\_name='serial-inference-batch',

sagemaker\_session=sess,

accept = CONTENT\_TYPE\_CSV

)

transformer.transform(data = input\_data\_path,

job\_name = job\_name,

content\_type = CONTENT\_TYPE\_CSV,

split\_type = 'Line')

transformer.wait()

The transformer job waits until all the input data is processed.

*Analyze the results*: As you can see below, each headline has a mixture of 2 to 3 topics. For example, in the headline, “norman moore calls for perth freight link decision from liberals”, International Politics & Conflict; Legal Challenges topics are predominant. These topics are very relevant, since the headline has to do with Australian politics; in this case, the opposition party (Liberals) are asking Government to decide on a freight link.

*List of headlines and corresponding mixture of topics:*

|  |  |
| --- | --- |
| Headline | War and Crime, Legal Challenges, Clashes and Conflict, International Politics & Conflict |
| murder conviction court of criminal appeal | [0.4160057902,0.1207091212,0.3973485529,0.0274028089,0.0385336392] |
| myster drug overdose gold coast one man remains critical | [0.3838274777,0.2403284609,0.1550810933,0.1542729735,0.0664899871] |
| nauru detention complaints 'largely minor' department says | [0.1731015444,0.2305851877,0.0811567158,0.3308500051,0.1843065023] |
| nauru detention policy a 'breach of human rights' amnesty | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| nbl podcast: week 2 | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| new royal adelaide hospital paper records foi | [0.247835502,0.068220146,0.2128179073,0.2778810859,0.1932453364] |
| nick kaldas appointed to committee investigating ira murders | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| nick kyrgios booted off atp tour for extended period | [0.0595307648,0.2493837774,0.0686811805,0.0535562448,0.5688480139] |
| norman moore calls for perth freight link decision from liberals | [0.0865799785,0.1025102213,0.2175595313,0.0734829158,0.5198673606] |
| nrn gmid basin report | [0.0440420732,0.0567624345,0.1520529538,0.2077625841,0.5393798947] |
| nsw considers 'no body; no parole' law | [0.0830427706,0.2022803426,0.1351942867,0.3568308055,0.2226518393] |
| nt royal commission in the tiwi islands | [0.2511883974,0.3452801108,0.0931304321,0.2273406237,0.0830604509] |
| pacific ocean garbage patch plastic much worse | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| panda worlds oldest dies in captivity aged 38 | [0.7684386373,0.0589178354,0.0715033337,0.046733249,0.0544069484] |
| paper plane championships teach children about community | [0.2065631598,0.2110857964,0.2137157619,0.1713560969,0.1972791702] |
| park rangers remove stuck tyre from rhino's mouth | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| penny wong questions george brandis diplomatic posting | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| petrol at some perth stations falls to 11 year lows | [0.3278017044,0.0844318494,0.1126552299,0.3827249706,0.0923862681] |
| plantation forestry growth used to target protected species | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| pm seeks to switch focus to industrial relations | [0.1974816322,0.0698087513,0.0816718265,0.3307147026,0.3203230798] |
| png captain assad vala leads from the front as | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| police questioning couple over critical lithgow toddler | [0.275745362,0.0724028349,0.3066048324,0.2734159827,0.0718309507] |
| police video shows car ramming officers motorcycle | [0.3554754555,0.0465094075,0.3460674286,0.2011484653,0.0507992655] |
| politics live october 17 | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| property market value to drop | [0.2724763155,0.1913396269,0.0972012505,0.1640012711,0.2749815285] |
| qld government plan to help save sea turtle eggs criticised | [0.2110161334,0.1764959097,0.2231868953,0.1801718622,0.2091292143] |
| queensland police hunt car hit police motorcycle | [0.3372704387,0.0470974259,0.3044666052,0.2448058128,0.0663596094] |

**References**:

<https://blog.insightdatascience.com/spark-pipelines-elegant-yet-powerful-7be93afcdd42>

<https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-batch.html>

<https://medium.com/ml2vec/topic-modeling-is-an-unsupervised-learning-approach-to-clustering-documents-to-discover-topics-fdfbf30e27df>

<https://spark.apache.org/docs/1.6.0/ml-guide.html#transformers>