Balancing the Dataset - AWS Glue Job Script (& III)

Write samples in S3

```
# Buid a Spark DataFrame that is the union of all the samples
balanced_df = samples[0]

for sample in samples[1:]:
     balanced_df = balanced_df.union(sample)

# Convert the DataFrame to a Glue DynamicFrame
balanced = DynamicFrame.fromDF(balanced_df, glueContext, "balanced")

# Write the data on the target bucket using parquet format
sampled_data_sink = glueContext.write_dynamic_frame.from_options(
     frame = balanced,
     connection_type='s3',
     connection_options={"path": target, "partitionKeys": ["product_category"]},
     format="parquet")
```


aws

SageMaker BlazingText

Two modes:

- Unsupervised
  - Highly optimized implementation of the Word2vec
  - Used for converting words to vectors (aka *word embeddings*)
- Supervised
  - Extends the fastText text classifier
  - Used for multi class/label text classification

aws

Target Format - BlazingText

- Single preprocessed text file
- Space separated tokens
- Single sentence per line
- Labels alongside the sentence
- A label is a word prefixed by the string __label__
- Can use training and validation channels

TODOs:

- Select only the used fields
- Tokenize review body and convert it to a space separated string and prepend the string with the product category label
- Split the dataset into training, validation, and test subsets
- Write each subset to a single object in S3

Preparing the Dataset - AWS Glue Job Script (I)

Select only the used fields

```
# Read from the data source
datasource = glueContext.create_dynamic_frame.from_catalog(
     database = database,
     table_name = table)

# Select only the fields that are going to be used
select = SelectFields.apply(
     frame = datasource,
     paths = ["product_category", "review_body"])
```


aws

Preparing the Dataset - AWS Glue Job Script (II)

Tokenize review body and convert it to a space separated string

```
# Transform the reviews text by applying the tokenize function to each row in the DynamicFrame
tokenized = Map.apply(frame = select, f = tokenize, transformation_ctx = "tokenized")

def tokenize(dynamicRecord):
    category = dynamicRecord['product_category'].lower()
    dynamicRecord['product_category'] = '__label__'+dynamicRecord['product_category'].lower
    dynamicRecord['review'] = transform_review_body(dynamicRecord.get('review_body', ''))
    return dynamicRecord

def transform_review_body(review_body):
    from nltk.tokenize import TweetTokenizer
    tknzr = TweetTokenizer()
    body = tknzr.tokenize(remove_tags(review_body.lower()))

    return(' '.join(body))
```


aws

Preparing the Dataset - AWS Glue Job Script (III)

Split the dataset into training, validation, and test subsets

```
# Split the sample into train, test, and validation sets. A Spark DataFrame is needed for th
df = tokenized.toDF()
train, validation, test = df.randomSplit(weights = [.6, .2, .2], seed = 42)
```

Preparing the Dataset - AWS Glue Job Script (IV)

Write each subset to a single object in S3

```
# Repartition the data frames to store each set into a single file and convert to DynamicFra
train_set = DynamicFrame.fromDF(train.repartition(1), glueContext, "train")
validation_set = DynamicFrame.fromDF(validation.repartition(1), glueContext, "validation")
test_set = DynamicFrame.fromDF(test.repartition(1), glueContext, "test")

# Write each set with a data sink
train_datasink = glueContext.write_dynamic_frame.from_options(
     frame = train_set,
     connection_type = "s3",
     connection_options = {"path": "{}/train".format(target)},
     format = "csv",
     format_options = {"separator": " ", "writeHeader": False, "quoteChar": "-1" })
```


aws

Using the generic Amazon SageMaker Estimator

- Part of the SageMaker Python SDK
- Needs to be configured with:
    - Container for BlazingText algorithm
    - Training and validation channels
    - IAM Role
    - Training instance configuration
    - Hyperparameters
    - Output location for the training artifacts

aws

from sagemaker python SDK.

In [15]:
```
import sagemaker
import json
import boto3

sess = sagemaker.Session()

role = sagemaker.get_execution_role()
print(role) # This is the role that SageMaker would use to leverage AWS resources (S3, CloudWat

bucket = 'sagemaker-reviews-demo'

job_run_id = 'jr_17371f644effb76dfb62c4a4bd3f305e09ee02f87a7723a5c83d83a5e2e7579b'

prefix = 'transformed/' + job_run_id #Replace with the prefix under which you want to store the
```

```
arn:aws:iam::[redacted]:role/service-role/AWSGlueServiceSageMakerNotebookRole-TextClassific
ation
```

You are going to need the nltk library, which is not included in the notebook instance, so let's use pip to install it (this is only needed once).

In [2]:
```
!pip install nltk
```

```
prefix = 'transformed/' + job_run_id #Replace with the prefix under which you want to store the
```

```
arn:aws:iam::[redacted]:role/service-role/AWSGlueServiceSageMakerNotebookRole-TextClassific
ation
```

You are going to need the nltk library, which is not included in the notebook instance, so let's use pip to install it (this is only needed once).

In [2]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/si
te-packages (3.3)
Requirement already satisfied: six in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/sit
e-packages (from nltk) (1.11.0)
You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [3]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/ec2-user/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[3]:
```
True
```

Out[3]: True

Data channels

Amazon SageMaker requires a channel for reading training data and another for validation data. To set this up, you need to create the sagemaker.session.s3_input objects from the data channels. These objects are then put in a simple dictionary, which the algorithm consumes.

In [4]:
```
train_channel = prefix + '/train/'
validation_channel = prefix + '/validation/'

s3_train_data = 's3://{}/{}'.format(bucket, train_channel)
s3_validation_data = 's3://{}/{}'.format(bucket, validation_channel)

train_data = sagemaker.session.s3_input(s3_train_data, distribution='FullyReplicated',
                        content_type='text/plain', s3_data_type='S3Prefix')

validation_data = sagemaker.session.s3_input(s3_validation_data, distribution='FullyReplicated',
                             content_type='text/plain', s3_data_type='S3Prefix')

data_channels = {'train': train_data, 'validation': validation_data}
```

Next you need to setup an output location at S3, where the model artifact will be dumped. These artifacts are also the output of the algorithm's traning job

the algorithm's traning job.

In [5]:
```
s3_output_location = 's3://{}/output/{}'.format(bucket, job_run_id)
```

Training

Now that you are done with all the setup that is needed, you are ready to train our object detector. To begin, create a `sageMaker.estimator.Estimator` object. This estimator will launch the training job.

In [6]:
```
region_name = boto3.Session().region_name
```

In [7]:
```
container = sagemaker.amazon.amazon_estimator.get_image_uri(region_name, "blazingtext", "latest")
print('Using SageMaker BlazingText container: {} ({})'.format(container, region_name))
```

```
Using SageMaker BlazingText container: 685385470294.dkr.ecr.eu-west-1.amazonaws.com/blazingte
xt:latest (eu-west-1)
```

Training the BlazingText model for supervised text classification

Similar to the original implementation of Word2Vec, SageMaker BlazingText provides an efficient implementation of the continuous bag-of-words (CBOW) and skip-gram architectures using Negative Sampling, on CPUs and additionally on GPU[s].

Modes	cbow (supports subwords training)	skipgram (supports subwords training)	batch_skipgram	supervised
Single CPU instance	✔	✔	✔	✔
Single GPU instance	✔	✔	✔ (Instance with 1 GPU only)	
Multiple CPU instances		✔		

Now, let's define the SageMaker `Estimator` with resource configurations and hyperparameters to train Text Classification on the prepared dataset, using "supervised" mode on a `p3.2xlarge` instance.

In [8]:
```
bt_model = sagemaker.estimator.Estimator(container,
                                         role,
                                         train_instance_count=1,
                                         train_instance_type='ml.c5.4xlarge',
                                         train_volume_size = 30,
                                         train_max_run = 360000,
                                         input_mode= 'File',
                                         output_path=s3_output_location,
                                         sagemaker_session=sess)
```

Please refer to [algorithm documentation] (https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext_hyperparameters.html) for the complete list of hyperparameters.

	Multiple CPU instances		✔

Now, let's define the SageMaker `Estimator` with resource configurations and hyperparameters to train Text Classification on the prepared dataset, using "supervised" mode on a `p3.2xlarge` instance.

In [8]:
```
bt_model = sagemaker.estimator.Estimator(container,
                                         role,
                                         train_instance_count=1,
                                         train_instance_type='ml.c5.4xlarge',
                                         train_volume_size = 30,
                                         train_max_run = 360000,
                                         input_mode= 'File',
                                         output_path=s3_output_location,
                                         sagemaker_session=sess)
```

Please refer to [algorithm documentation] (https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext_hyperparameters.html) for the complete list of hyperparameters.

In [9]:
```
bt_model.set_hyperparameters(mode='supervised',
                             min_epochs=5,
                             epochs=100,
                             min_count=2,
                             learning_rate=0.010817706468703524,
```

```
                                    train_instance_type='ml.c5.4xlarge',
                                    train_volume_size = 30,
                                    train_max_run = 360000,
                                    input_mode= 'File',
                                    output_path=s3_output_location,
                                    sagemaker_session=sess)
```

Please refer to [algorithm documentation]
(https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext_hyperparameters.html) for the complete list of hyperparameters.

In [9]:
```
bt_model.set_hyperparameters(mode='supervised',
                             min_epochs=5,
                             epochs=100,
                             min_count=2,
                             learning_rate=0.010817706468703524,
                             vector_dim=134,
                             early_stopping=True,
                             patience=30,
                             word_ngrams=5)
```

We will use SageMaker hyperparameter tuning to automate the searching process effectively. Specifically, we specify a range, or a list of possible values in the case of categorical hyperparameters, for each of the hyperparameter that we plan to tune. SageMaker hyperparameter tuning will automatically launch multiple training jobs with different hyperparameter settings,

```
patience=30,
word_ngrams=5)
```

We will use SageMaker hyperparameter tuning to automate the searching process effectively. Specifically, we specify a range, or a list of possible values in the case of categorical hyperparameters, for each of the hyperparameter that we plan to tune. SageMaker hyperparameter tuning will automatically launch multiple training jobs with different hyperparameter settings, evaluate results of those training jobs based on a predefined "objective metric", and select the hyperparameter settings for future attempts based on previous results. For each hyperparameter tuning job, we will give it a budget (max number of training jobs) and it will complete once that many training jobs have been executed.

First we define the ranges for some of the BlaingText supervised mode algorithm

In [10]:
```
from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, Hyperpa

hyperparameter_ranges = {'min_count': IntegerParameter(2, 4),
                         'learning_rate': ContinuousParameter(0.01, 0.08),
                         'vector_dim': IntegerParameter(100, 200),
                         'word_ngrams': IntegerParameter(2, 5)}
```

Then define the objective metric

In [11]:
```
objective_metric_name = 'validation:accuracy'
```

We will use SageMaker hyperparameter tuning to automate the searching process effectively. Specifically, we specify a range, or a list of possible values in the case of categorical hyperparameters, for each of the hyperparameter that we plan to tune. SageMaker hyperparameter tuning will automatically launch multiple training jobs with different hyperparameter settings, evaluate results of those training jobs based on a predefined "objective metric", and select the hyperparameter settings for future attempts based on previous results. For each hyperparameter tuning job, we will give it a budget (max number of training jobs) and it will complete once that many training jobs have been executed.

First we define the ranges for some of the BlaingText supervised mode algorithm

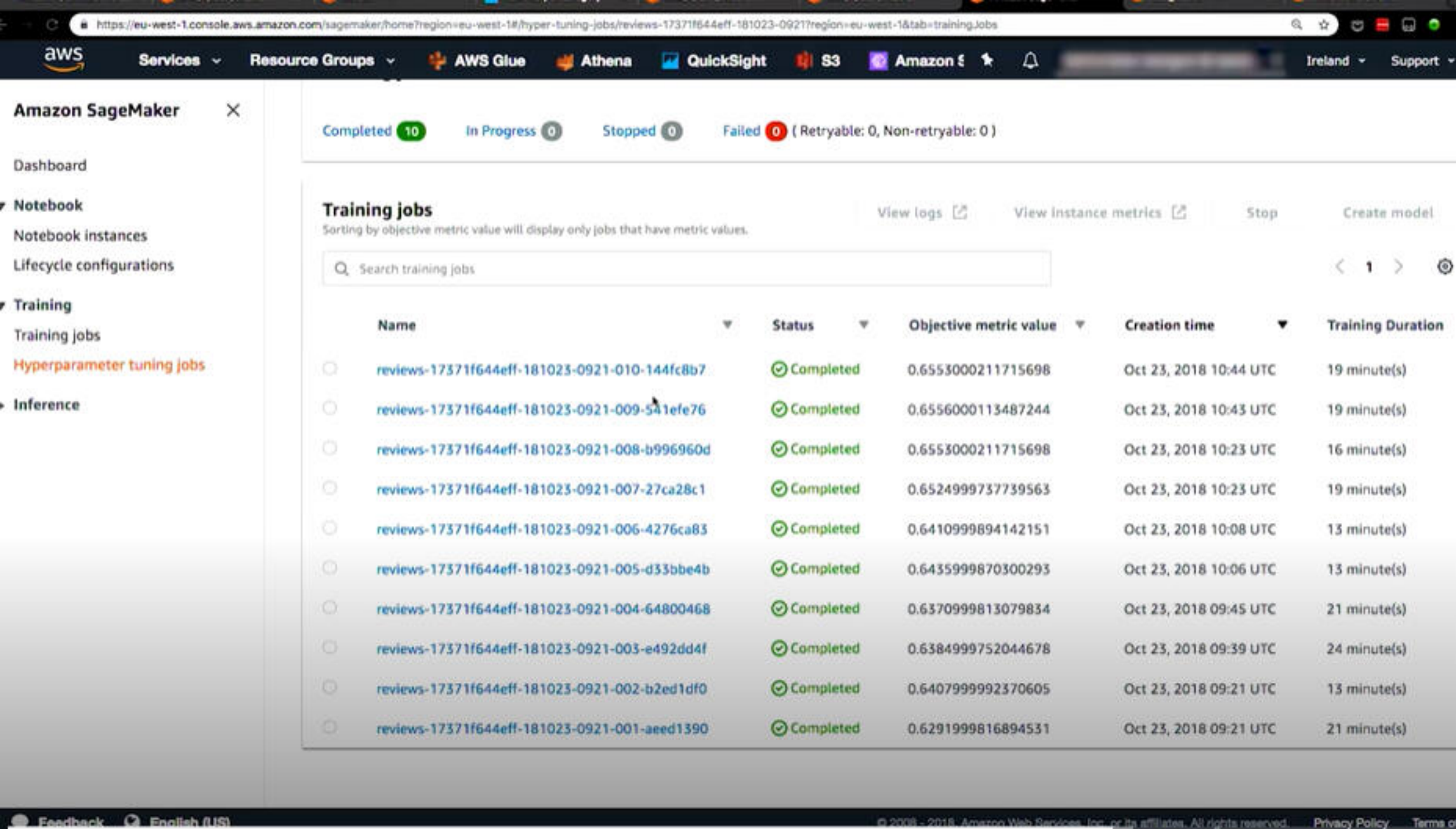
In [10]:
```
from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, Hyperp

hyperparameter_ranges = {'min_count': IntegerParameter(2, 4),
                         'learning_rate': ContinuousParameter(0.01, 0.08),
                         'vector_dim': IntegerParameter(100, 200),
                         'word_ngrams': IntegerParameter(2, 5)}
```

Then define the objective metric

In [11]:
```
objective_metric_name = 'validation:accuracy'
```

Now, you'll create a HyperparameterTuner object, to which you pass:

- The BlazingTest estimator created above


https://eu-west-1.console.aws.amazon.com/sagemaker/home?region=eu-west-1#/hyper-tuning-jobs/reviews-17371f644eff-181023-0921?region=eu-west-1&tab=trainingJobs
aws
Services
Resource Groups
AWS Glue
Athena
QuickSight
S3
Amazon S
Ireland
Support
Amazon SageMaker
Dashboard
Notebook
Notebook instances
Lifecycle configurations
Training
Training jobs
Hyperparameter tuning jobs
Inference
Completed 10
In Progress 0
Stopped 0
Failed 0 (Retryable: 0, Non-retryable: 0)
Training jobs
Sorting by objective metric value will display only jobs that have metric values.
View logs
View instance metrics
Stop
Create model
Search training jobs
1
Name
Status
Objective metric value
Creation time
Training Duration
reviews-17371f644eff-181023-0921-010-144fc8b7 Completed 0.6553000211715698 Oct 23, 2018 10:44 UTC 19 minute(s)
reviews-17371f644eff-181023-0921-009-541efe76 Completed 0.6556000113487244 Oct 23, 2018 10:43 UTC 19 minute(s)
reviews-17371f644eff-181023-0921-008-b996960d Completed 0.6553000211715698 Oct 23, 2018 10:23 UTC 16 minute(s)
reviews-17371f644eff-181023-0921-007-27ca28c1 Completed 0.6524999737739563 Oct 23, 2018 10:23 UTC 19 minute(s)
reviews-17371f644eff-181023-0921-006-4276ca83 Completed 0.6410999894142151 Oct 23, 2018 10:08 UTC 13 minute(s)
reviews-17371f644eff-181023-0921-005-d33bbe4b Completed 0.6435999870300293 Oct 23, 2018 10:06 UTC 13 minute(s)
reviews-17371f644eff-181023-0921-004-64800468 Completed 0.6370999813079834 Oct 23, 2018 09:45 UTC 21 minute(s)
reviews-17371f644eff-181023-0921-003-e492dd4f Completed 0.6384999752044678 Oct 23, 2018 09:39 UTC 24 minute(s)
reviews-17371f644eff-181023-0921-002-b2ed1df0 Completed 0.6407999992370605 Oct 23, 2018 09:21 UTC 13 minute(s)
reviews-17371f644eff-181023-0921-001-aeed1390 Completed 0.6291999816894531 Oct 23, 2018 09:21 UTC 21 minute(s)
Feedback
English (US)
Privacy Policy
Terms of

Amazon SageMaker

Dashboard

Notebook

Notebook instances

Lifecycle configurations

Training

Training jobs

Hyperparameter tuning jobs

Inference

Name	Status	Objective metric	Value
reviews-17371f644eff-181023-0921-009-541efe76	Completed	validation:accuracy	0.6556000113487244

Best training job hyperparameters

Name	Type	Value
_tuning_objective_metric	FreeText	validation:accuracy
early_stopping	FreeText	True
epochs	FreeText	100
learning_rate	Continuous	0.01
min_count	Integer	2
min_epochs	FreeText	5
mode	FreeText	supervised
patience	FreeText	30
sagemaker_estimator_class_name	FreeText	"Estimator"
sagemaker_estimator_module	FreeText	"sagemaker.estimator"
vector_dim	Integer	183

Four phases of ML

Discovery

Integration

Training

Deployment

aws

https://aws-glue-textclassfication.notebook.eu-west-1.sagemaker.aws/notebooks/text-classification/sagemaker-notebooks/reviews-classification-hpt.ipynb

jupyter reviews-classification-hpt Last Checkpoint: a day ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Markdown

In [ ]:
```
boto3.client('sagemaker').describe_hyper_parameter_tuning_job(
    HyperParameterTuningJobName=tuner.latest_tuning_job.job_name)['HyperParameterTuningJobStatus
```

Hosting / Inference

Once the training is done, we can deploy the trained model as an Amazon SageMaker real-time hosted endpoint. This will allow us to make predictions (or inference) from the model. Note that we don't have to host on the same type of instance that we used to train. Because instance endpoints will be up and running for long, it's advisable to choose a cheaper instance for inference. **If you trained the model using the hyperparameter tuning, use the console**

In [ ]:
```
tuner.deploy(initial_instance_count = 1, instance_type = 'ml.m4.xlarge', endpoint_name='textclas
#text_classifier = bt_model.deploy(initial_instance_count = 1,instance_type = 'ml.m4.xlarge')
```

Use JSON format for inference

BlazingText supports `application/json` as the content-type for inference. The payload should contain a list of sentences with the key as "**instances**" while being passed to the endpoint.

In [ ]:
```
with
sentences = ["disappointed, stopped working due to water after only two weeks",
             "10 year old loved it!"]
```

jupyter test_model Last Checkpoint: 23/10/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted conda_python3

Run Code

```
# using the same nltk tokenizer that we used during data preparation for training
tokenized_sentences = [' '.join(nltk.word_tokenize(sent)) for sent in sentences]

payload = {"instances" : tokenized_sentences}

response = text_classifier.predict(json.dumps(payload))

predictions = json.loads(response)
print(json.dumps(predictions, indent=2))
```

```
[nltk_data] Downloading package punkt to /home/ec2-user/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[
  {
    "prob": [
      0.39650389552116394
    ],
    "label": [
      "__label__lawn_and_garden"
    ]
  },
  {
    "prob": [
      0.5274955630302429
    ],
    "label": [
      "__label__toys"
    ]
```

jupyter test_model Last Checkpoint: 23/10/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted conda_python3

Run Code

In [15]:
```
payload = {"instances" : tokenized_sentences,
           "configuration": {"k": 3}}

response = text_classifier.predict(json.dumps(payload))

predictions = json.loads(response)
print(json.dumps(predictions, indent=2))
```


```
[
  {
    "prob": [
      0.39650389552116394,
      0.19785909354686737,
      0.1759667694568634
    ],
    "label": [
      "__label__lawn_and_garden",
      "__label__watches",
      "__label__major_appliances"
    ]
  },
  {
    "prob": [
      0.5274955630302429,
      0.31088608503341675,
      0.05716495215892792
    ],
```

jupyter test_model Last Checkpoint: 23/10/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | conda_python3

Run Code

```
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

AWS Glue + Amazon SageMaker Cost

Development endpoint:

- 2 DPUs x 3 hours x $0.44 DPU-hour = $2.64

Sampling Job:

- 30 DPUs x 6 min (minimum 10) x $0.44 DPU-hour = $2.2

Preparation Job:

- 20 DPUs x 25 min x $0.44 DPU-hour = $3.67

SageMaker Notebook instance:

- 1 ml.t2.medium x 3 hours x $0.05 = $0.15

Hyperparameter Tuning Job:

- 1 ml.c5.4xlarge x $1.075 x 2 h 58 min total training = $3.19

Total: $11.85

AWS Glue + Amazon SageMaker Cost

Development endpoint:

- 2 DPUs x 3 hours x $0.44 DPU-hour = $2.64

Sampling Job:

- 30 DPUs x 6 min (minimum 10) x $0.44 DPU-hour = $2.2

Preparation Job:

- 20 DPUs x 25 min x $0.44 DPU-hour = $3.67

SageMaker Notebook instance:

- 1 ml.t2.medium x 3 hours x $0.05 = $0.15

Hyperparameter Tuning Job:

- 1 ml.c5.4xlarge x $1.075 x 2 h 58 min total training = $3.19

Total: $11.85

AWS Glue + Amazon SageMaker Cost

Development endpoint:

- 2 DPUs x 3 hours x $0.44 DPU-hour = $2.64

Sampling Job:

- 30 DPUs x 6 min (minimum 10) x $0.44 DPU-hour = $2.2

Preparation Job:

- 20 DPUs x 25 min x $0.44 DPU-hour = $3.67

SageMaker Notebook instance:

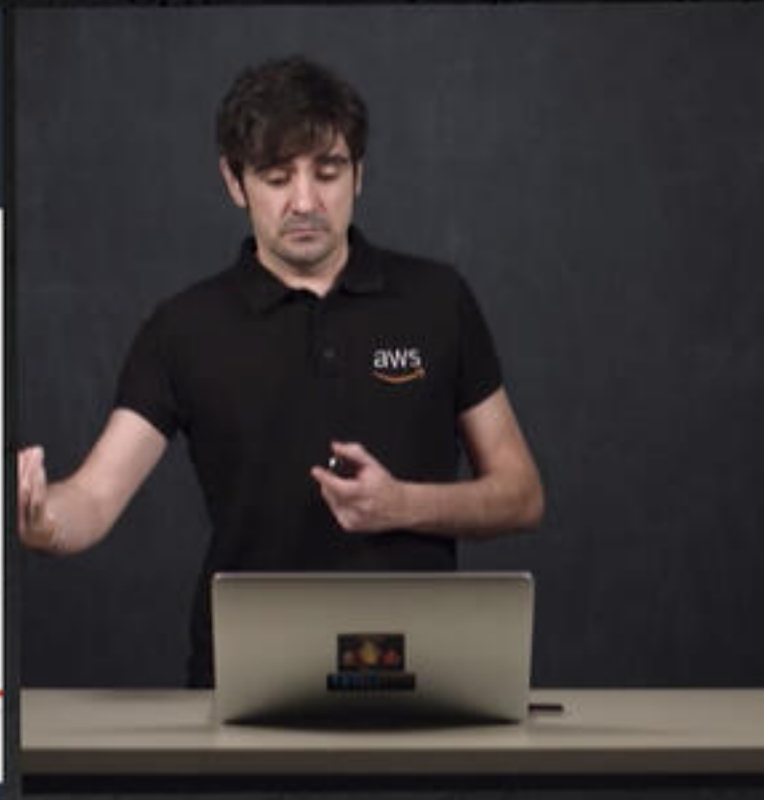
- 1 ml.t2.medium x 3 hours x $0.05 = $0.15

Hyperparameter Tuning Job:

- 1 ml.c5.4xlarge x $1.075 x 2 h 58 min total training = $3.19

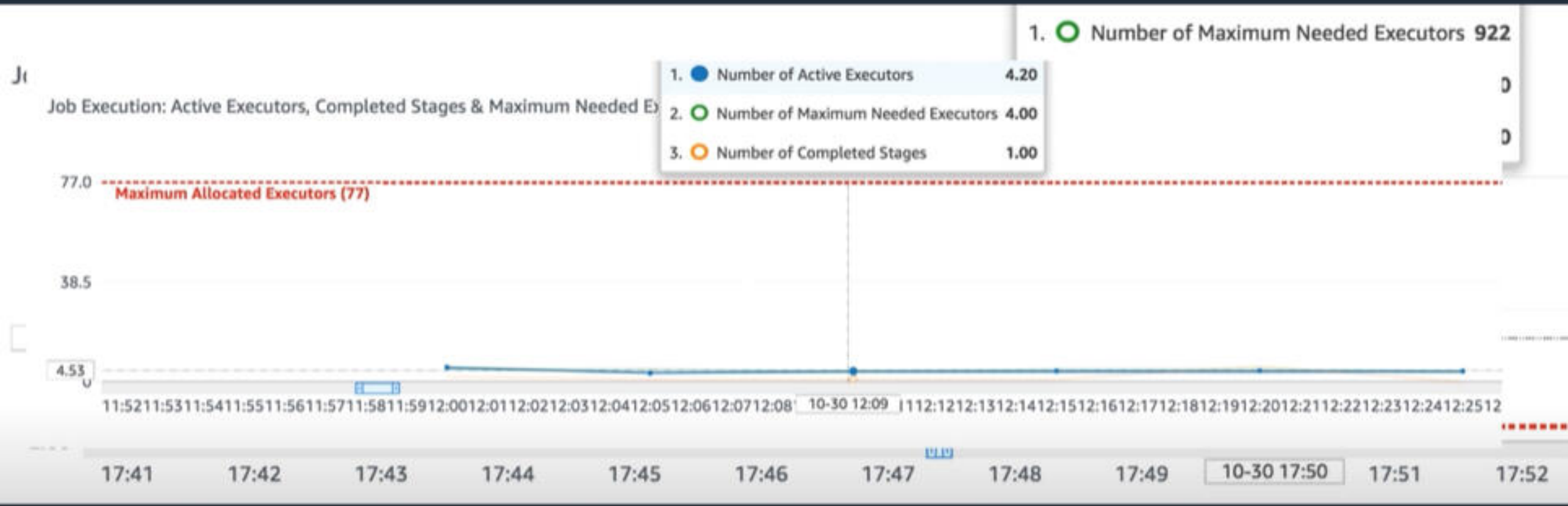
Total: $11.85

Glue Jobs Optimization



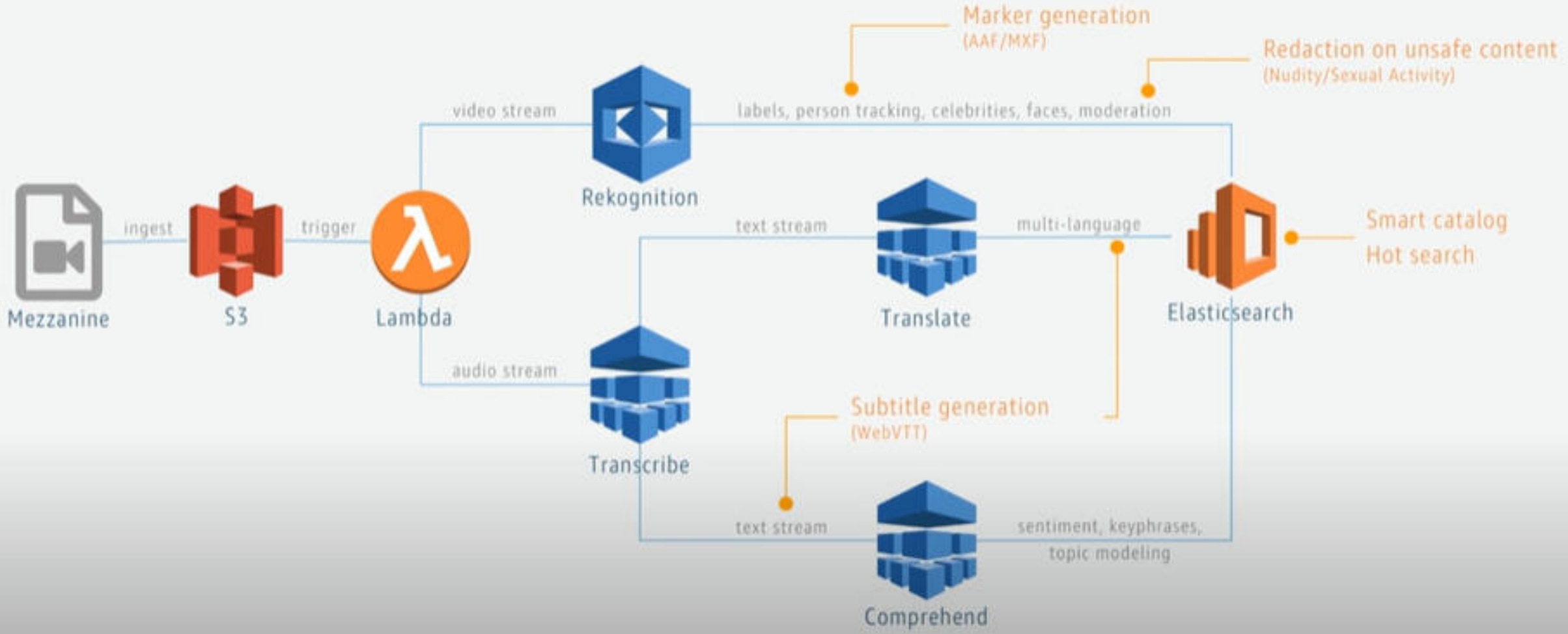
aws

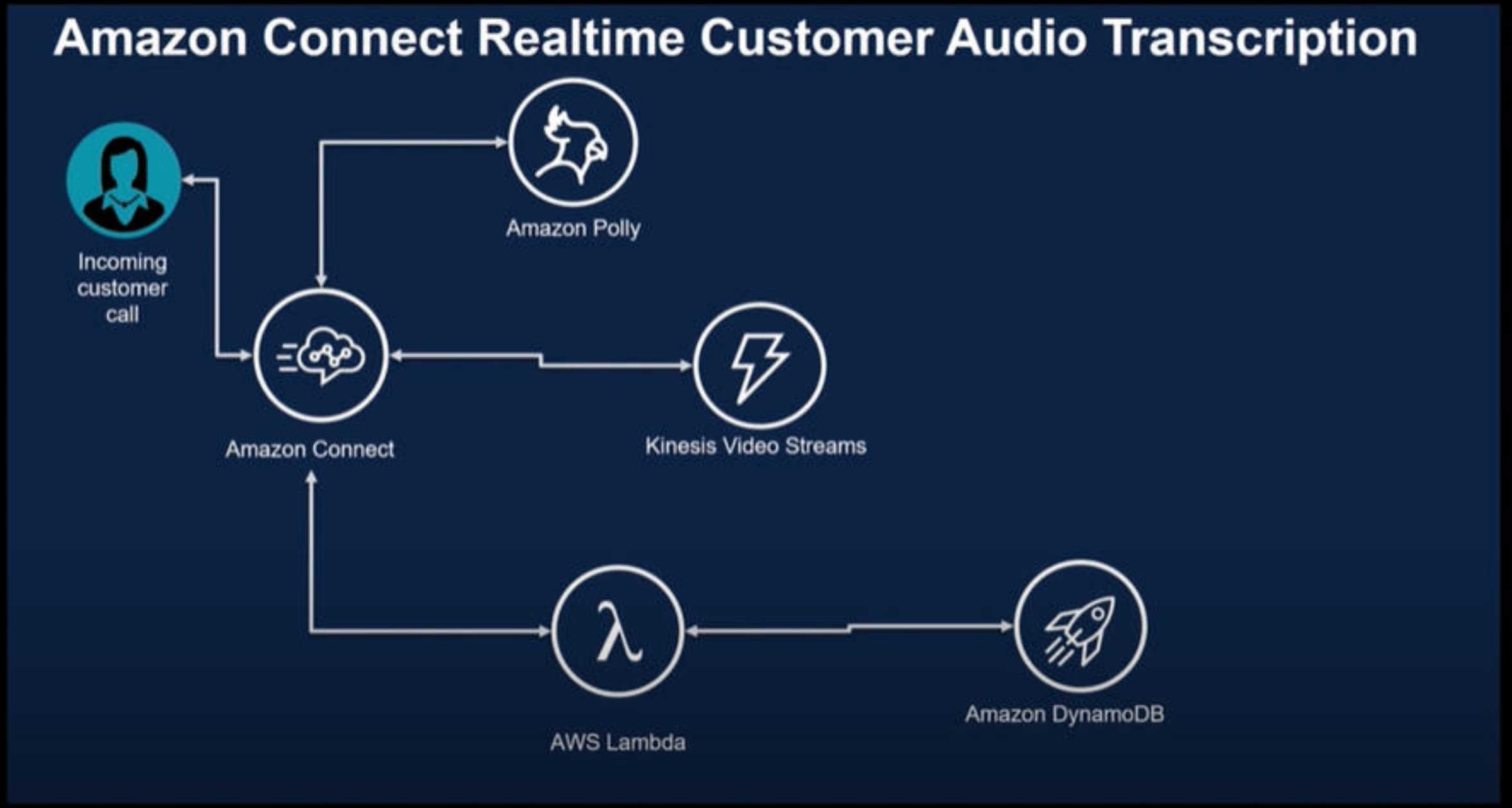
Glue Jobs Optimization



aws

AI/ML use cases in M&E field



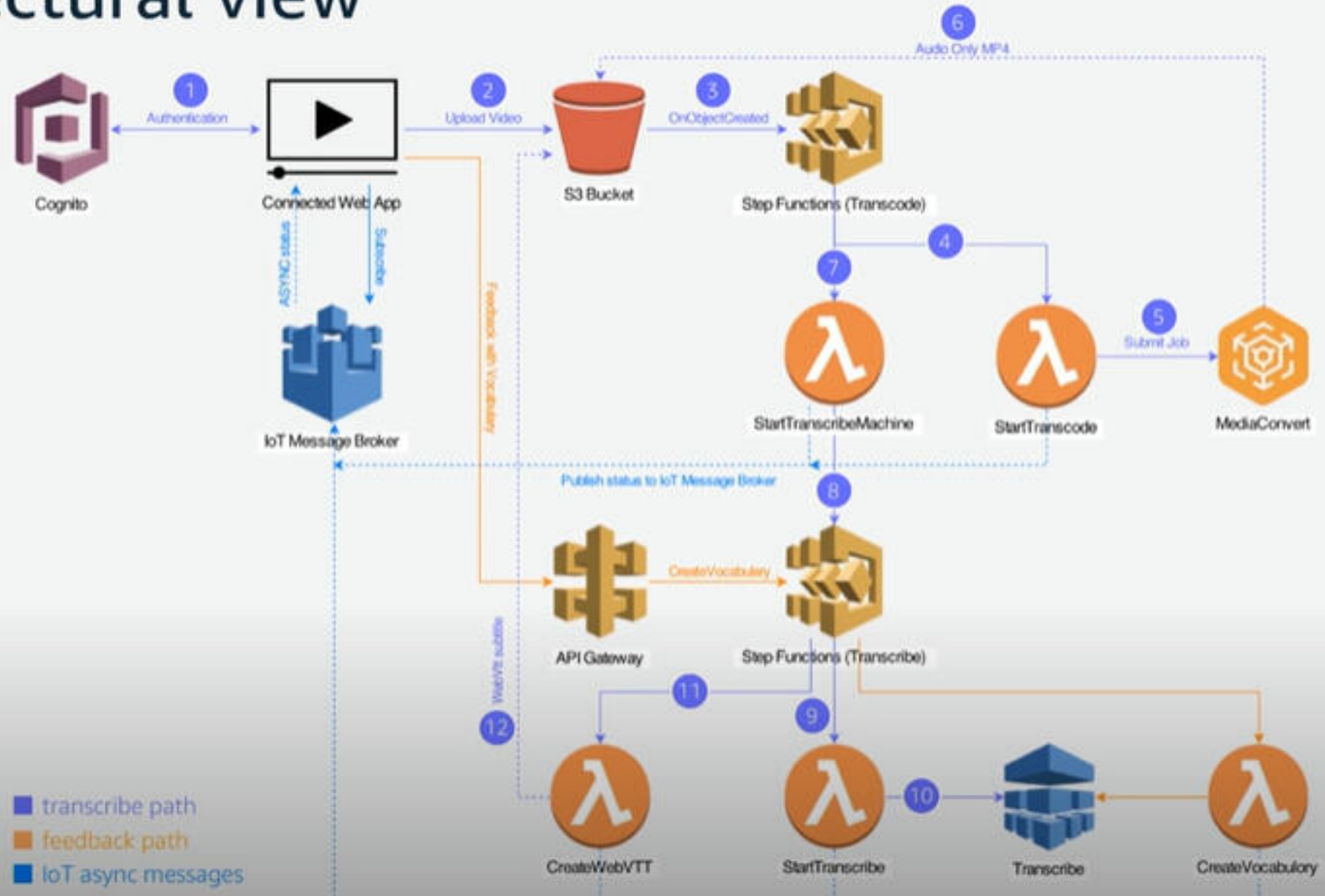


details like the ARN, the

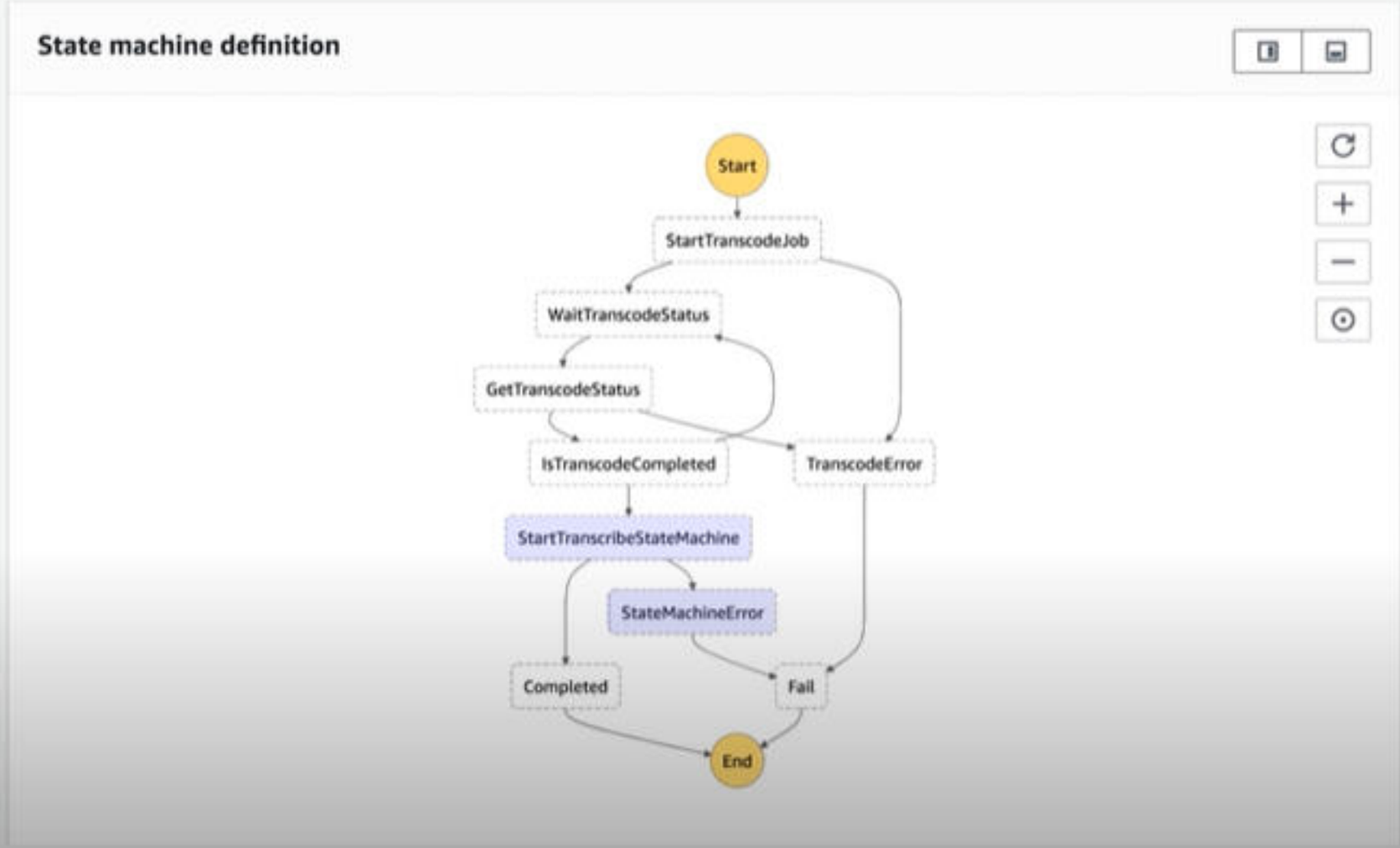
MediaConvert: submit job (nodeJS)

```
async function submitJob(template) {
  try {
    /* create a temp instance to find out the endpoint */
    const temporary = new AWS.MediaConvert({ apiVersion: '2017-08-29' });
    const data = await temporary.describeEndpoints().promise();
    const { Endpoints } = data;
    if (!Endpoints || !Endpoints.length) {
      throw new Error('no endpoint found!');
    }
    /* IMPORTANT: create a new instance and use the endpoint from describeEndpoint */
    const [{ Url: endpoint }] = Endpoints;
    const instance = new AWS.MediaConvert({
      apiVersion: '2017-08-29',
      endpoint,
    });
    const response = await instance.createJob(template).promise();
    return response;
  } catch (e) {
    throw e;
  }
}

const template = { /* Job JSON template */ };
const result = await submitJob(template);
```

Architectural view



Step Functions: transcoding state machine



Transcribe: start transcribe service

```
const Basename = 'SOME_AUDIO_FILE';

const params = {
  LanguageCode: 'en-US',
  MediaFormat: 'mp4',
  Media: { MediaFileUri: mediaFileUri },
  /* generate a unique transcribe name */
  TranscriptionJobName: `${Basename}-${new Date().toISOString().replace(/[-:.]/g, '')}`,
};

/* if vocabulary name exists, uses it */
if (Vocabularies) {
  params.Settings = { VocabularyName: Basename };
}
const transcribe = new TranscribeService({ apiVersion: '2017-10-26' });
const response = await transcribe.startTranscriptionJob(params).promise();
```

Transcribe: webvtt subtitle

```
WEBVTT

1
00:00:00.040 --> 00:00:03.040
Tell us which <c.five>ws</c> services
2
00:00:03.480 --> 00:00:06.480
Absolutely. I love young <c.six>lumber</c> yard, which is a game

3
00:00:06.600 --> 00:00:09.600
engine, that's really good, most like history, because

4
00:00:09.800 --> 00:00:12.800
storage is fantastic, and, of course, you should really <c.five>lam</c>

...
```

IoT: mqtt message broker


```
# Define your message topic.
anyTopicName/status
```

IoT: mqtt message broker


```
# Attach policy to Cognito user (identity)
aws iot attach-policy \
--policy-name anyThingPolicy \
--target "us-east-1:00000000-0000-0000-0000-000000000000"
```

aws

IoT: publisher

```
class IotStatus {
  static async publish(endpoint, topic, payload) {
    try {
      const iotData = new AWS.IotData({ apiVersion: '2015-05-28', endpoint });
      const params = { topic, payload, qos: 0 };
      const response = await iotData.publish(params).promise();
    } catch (e) {
      e.message = `IotStatus.publish: ${e.message}`;
    }
  }
}

/* Backend state machine to publish messages */
const endpoint = 'https://xxxxxxxx-ats.iot.eu-west-2.amazonaws.com';
IotStatus.publish(endpoint, 'anyTopicName/status', 'test message');
```

IoT: subscriber

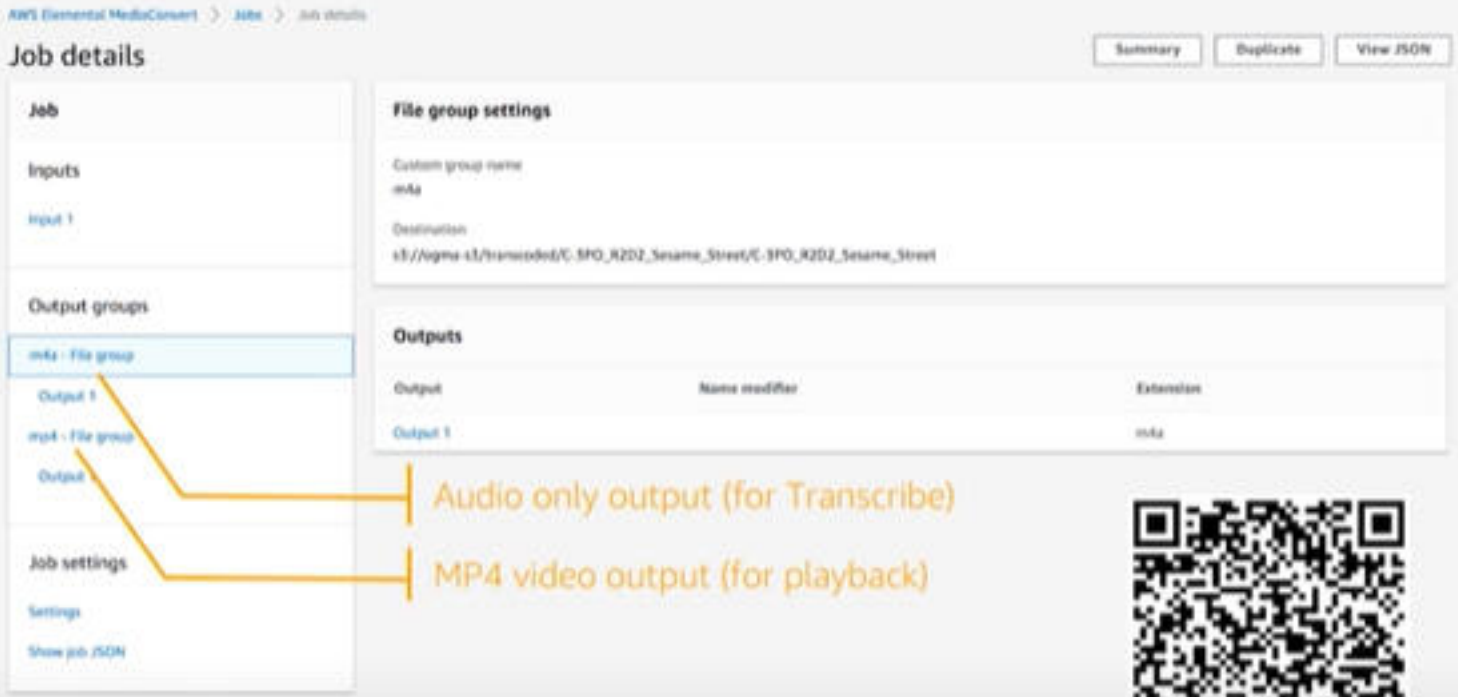
```
/* Download IoT JS SDK https://github.com/aws/aws-iot-device-sdk-js */

const subscriber = AWSIoTData.device({
  host: 'xxxxxxxx-ats.iot.us-east-1.amazonaws.com',
  region: 'us-east-1',
  clienId: 'cognito-user',
  protocol: 'wss',
  debug: false,
  accessKeyId,
  secretKey,
  sessionToken,
});

subscriber.on('connect', () => {
  subscriber.subscribe('anyTopicName/status');
});

subscriber.on('message', (topic, payload) => {
  console.log(`Received: topic ${topic}: payload: ${payload.toString()}`);
  const message = JSON.parse(payload.toString());
  /* do something */
});
```

MediaConvert: transcoding

MediaConvert: IAM service role

```
# IAM role to allow MediaConvert to access S3/API resources
aws iam create-role \
--role-name MediaConvertServiceRole \
--assume-role-policy-document \
'{"Version":"2012-10-
17","Statement":[{"Sid":"","Effect":"Allow","Principal":{"Service":"mediaconvert.amazonaws.
com"},"Action":"sts:AssumeRole"}]}'


# Attach S3 policy
aws iam attach-role-policy \
--policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess \
--role-name MediaConvertServiceRole


# Attach API Gateway policy
aws iam attach-role-policy \
--policy-arn arn:aws:iam::aws:policy/AmazonAPIGatewayInvokeFullAccess \
--role-name MediaConvertServiceRole
```

MediaConvert: define a job template



Sample JSON

aws

MediaConvert: submit job

```
# First, find out the per-region, per-account endpoint
aws mediaconvert describe-endpoints --region eu-west-1

{
  "Endpoints": [{
    "Url": "https://xxxxxxxx.mediaconvert.eu-west-1.amazonaws.com"
  }]
}


# submit job to the endpoint
aws mediaconvert create-job \
--role arn:aws:iam::000000000000:role/MediaConvertServiceRole \
--settings file://sampleJob.json \
--endpoint https://xxxxxxxx.mediaconvert.eu-west-1.amazonaws.com \
--region eu-west-1

{
  JSON Job data...
}
```

aws

MediaConvert: submit job

```
# First, find out the per-region, per-account endpoint
aws mediaconvert describe-endpoints --region eu-west-1

{
  "Endpoints": [{
    "Url": "https://xxxxxxxx.mediaconvert.eu-west-1.amazonaws.com"
  }]
}


# submit job to the endpoint
aws mediaconvert create-job \
--role arn:aws:iam::000000000000:role/MediaConvertServiceRole \
--settings file://sampleJob.json \
--endpoint https://xxxxxxxx.mediaconvert.eu-west-1.amazonaws.com \
--region eu-west-1

{
  JSON Job data...
}
```

MediaConvert: submit job (nodeJS)

```
async function submitJob(template) {
  try {
    /* create a temp instance to find out the endpoint */
    const temporary = new AWS.MediaConvert({ apiVersion: '2017-08-29' });
    const data = await temporary.describeEndpoints().promise();
    const { Endpoints } = data;
    if (!Endpoints || !Endpoints.length) {
      throw new Error('no endpoint found!');
    }
    /* IMPORTANT: create a new instance and use the endpoint from describeEndpoint */
    const [{ Url: endpoint }] = Endpoints;
    const instance = new AWS.MediaConvert({
      apiVersion: '2017-08-29',
      endpoint,
    });
    const response = await instance.createJob(template).promise();
    return response;
  } catch (e) {
    throw e;
  }
}

const template = { /* Job JSON template */ };
const result = await submitJob(template);
```

A few gotcha…

1GB file size limit

use MediaConvert, ETS, or ffmpeg to extract audio stream from source file

2 hours duration limit

split source file and post-process to stitch (modify timecode of) the transcription result

working with acronym

put a space in between acronym. A W S, E M R, and etc

working with numbers

pre/post-process to convert numbers before and after Transcribe

Evaluation of Accuracy: Likelihood

aws training and certification

Likelihood: Likelihood is probability of some observed data—e.g. test, given a model.

$$P(\mathcal{E}_{test};\ \theta) = \prod_{E\ \epsilon \mathcal{E}_{test}} P(E;\theta)$$

Example

$|\mathcal{E}_{test}| = 10^6$

E1 = I am from pittsburg, $P(E;\theta) = 2.1 \times 10^{-6}$ *

E2 = I study at a university $P(E;\theta) = 1.1 \times 10^{-8}$

E3 = my mother is from utah, $P(E;\theta) = 2 \times 10^{-9}$

$$P(\mathcal{E}_{test};\ \theta) = \prod_{E\ \epsilon \mathcal{E}_{test}} P(E;\theta) = 2.1 \times 10^{-6} \times 1.1 \times 10^{-8} \times 2.0 \times 10^{-9} = 4.62 \times 10^{-23}$$

*probabilities are arbitrary just to demonstrate the range

Evaluation of Accuracy: Log Likelihood

aws training and certification

- $Theorem: \log_b \prod_{i=1}^{n} x_i = \sum_{i=1}^{n} \log_b x_i; Example: \log(x.y) = \log(x) + \log(y)$
- We have seen probabilities of likelihood are very tiny. Multiplying these small values make the probabilities even smaller numbers. It is easier to work with log probabilities for measuring accuracy.

$$\log\big(P(\mathcal{E}_{test};\theta)\big) = \log_b \prod_{E\,\epsilon\,\mathcal{E}_{test}} P(E;\theta) = \sum_{E\,\epsilon\,\mathcal{E}_{test}} \log(\,P(E;\theta))\,.$$

- **Example:**

$|\mathcal{E}_{test}| = \mathbf{10^6}$

E1 = I am from pittsburg, $\boldsymbol{P(E;\theta)} = 2.1 \times 10^{-6} \therefore \log(\mathrm{P(E;}\theta) = \log(2.1) + (-6) = -5.67$ *

E2 = I study at a university $\boldsymbol{P(E;\theta)} = 1.1 \times 10^{-8} \therefore \log(\mathrm{P(E;}\theta) = \log(1.1) + (-8) = -7.96$

E3 = my mother is from utah, $\boldsymbol{P(E;\theta)} = 2 \times 10^{-9} \therefore \log(\mathrm{P(E;}\theta) = \log(2.0) + (-9) = -8.70$

$$\boldsymbol{P(\mathcal{E}_{test};\,\theta) = \prod_{E\,\epsilon\,\mathcal{E}_{test}} P(E;\theta) = \sum_{E\,\epsilon\,\mathcal{E}_{test}} \log(\,P(E;\theta)) = -5.67 - 7.96 - 8.70 = -22.33}$$

10 09:00 / 42:33

*probabilities are arbitrary just to demonstrate the range

Evaluation of Accuracy: Perplexity

- It is common to divide log-likelihood by the number of words in the corpus. This makes cross-corpora measurement easier.

$$\text{length}(\mathcal{E}_{test}) = \sum_{E \,\epsilon\, \mathcal{E}_{test}} |E|$$

$$ppl(\mathcal{E}_{test}; \theta) = e^{-(\log(P(\mathcal{E}_{test};\theta))/lengt(\mathcal{E}_{test})}$$

- This accuracy indicator is called **perplexity**.

NMT - Motivation

aws training and certification

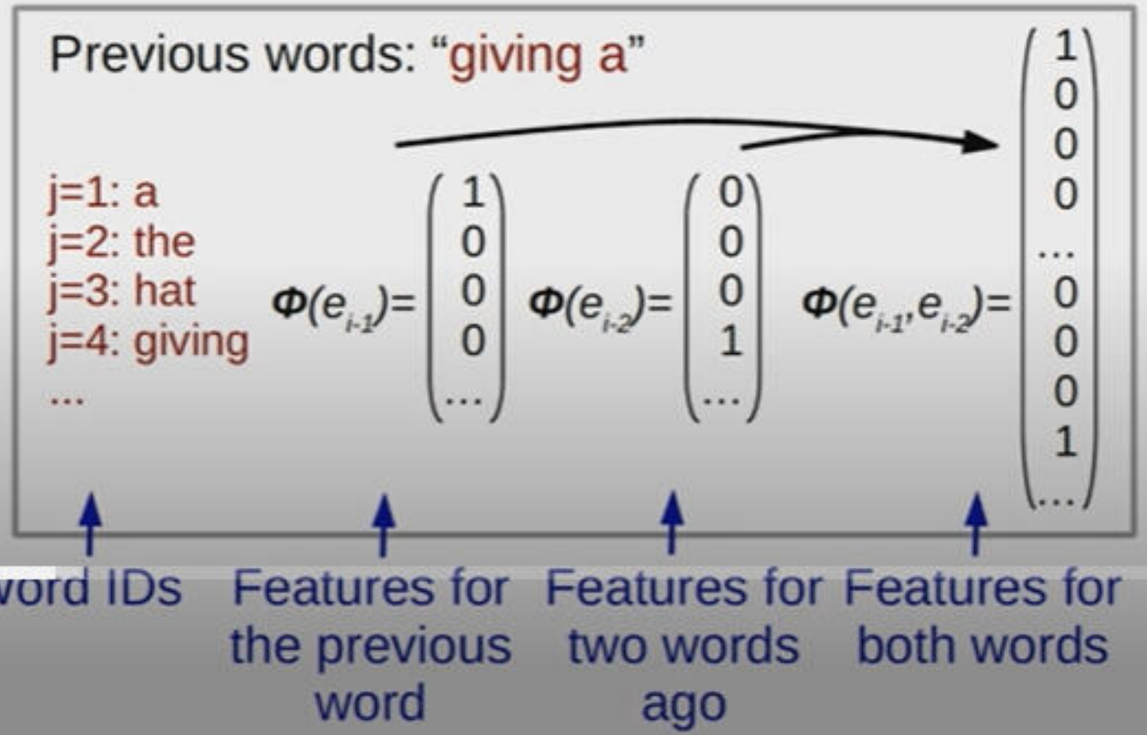
farmers eat	steak → **high** hay → **low**	cows eat	steak → **low** hay → **high**
farmers grow	steak → **low** hay → **high**	cows grow	steak → **low** hay → **low**

- In separate contexts e_{t-2} = "farmers" is compatible with e_t = "hay" and e_{t-1} = "eats".
- If we are using feature sets depending on e_{t-1} and e_{t-2}, then a sentence like "farmers eat hey" cannot be ruled out, even though it is an unnatural sentence or perhaps an unnatural farmer.

Step1: Word Embedding

aws training and certification

- In Neural Language Models we represent words as word embedding, **a vector representation of words**.
- Then we concatinate these words to the length of the context (3 for trigrams).



Step3: Score and Probability

aws training and certification

- Next, we calculate the score vector for each word. This is done by performing an affine* transform of the hidden vector h with a weight matrix and adding a bias vector.
- Finally, we get a probability estimate p by running the calculated scores through a softmax function.

$$\boldsymbol{m} = \text{concat}(M_{\cdot,e_{t-2}}, M_{\cdot,e_{t-1}})$$
$$\boldsymbol{h} = \tanh(W_{mh}\boldsymbol{m} + \boldsymbol{b}_h)$$
$$\boldsymbol{s} = W_{hs}\boldsymbol{h} + \boldsymbol{b}_s$$
$$\boldsymbol{p} = \text{softmax}(\boldsymbol{s})$$

* An affine transformation is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation). Ref: Wolfram MathWorld

Benefits of Neural Models Contd.

aws training and certification

- **Ability to skip previous words:** n-gram models refer to all sequential words depending the length of the context. Neural Models can skip words.

- **Simplistic Models:** SMT models require complex data-structures, in production often using 80 GBs or more of heap space. High quality NMT production models are generally less than 1 GB.

Dependencies: Reference and Context Cont.

aws training and certification

- **The Spanish King** officially abdicated in **favour** of his **son**, **Felipe.** **Felipe** will be confirmed tomorrow as the new Spanish king. As the new **king, he** will be the soverign.

Reference | Context | Reference & Context

Dependencies: Reference and Context Cont.

aws training and certification

- **The Spanish King** officially abdicated in **favour** of his **son, Felipe**. **Felipe** will be confirmed tomorrow as the new Spanish **king**. As the new **king, he** will be the soverign.
- Their duo was known as "Buck and Bubbles". Bubbles tapped along as Buck **played on the stride piano and sang** until Bubbles was totally tapped out.

Reference | Context | Reference & Context

Problems of Translation

aws training and certification

To perform translation we need to solve the following problems:

Modeling: Decide what our model $P(E|F;\theta)$ will look like. What parameters will it have, and how will the parameters specify a probability distribution?

Learning: We need a method to learn appropriate values for parameters from training data.

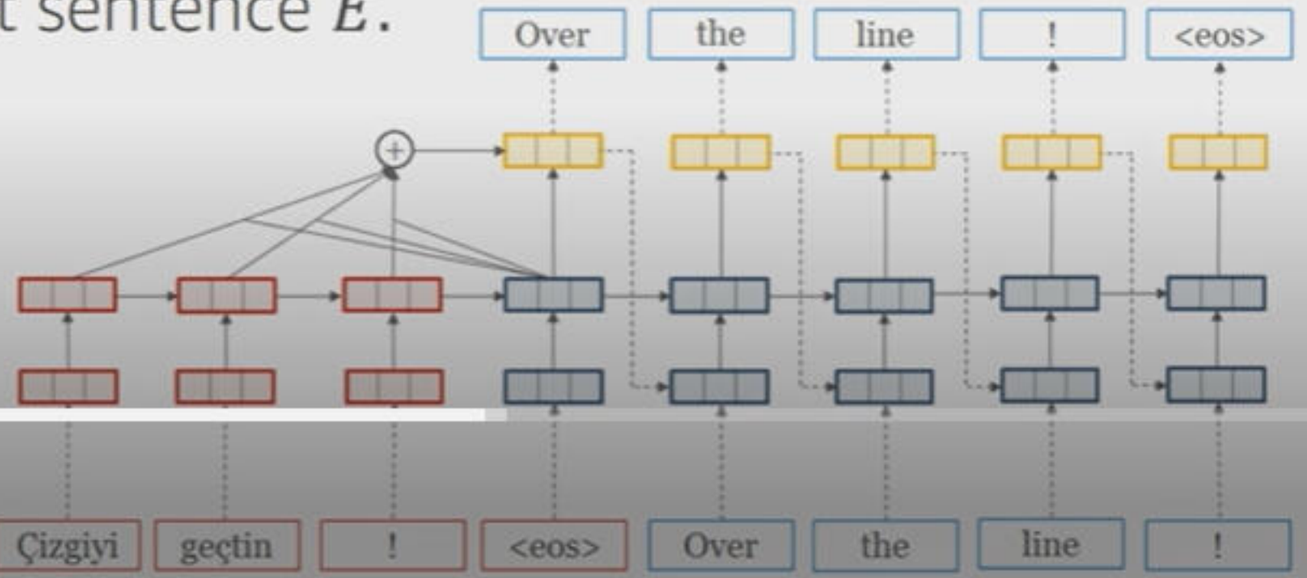
Search: We need to find the most probable sentence (solving "argmax"). This process of searching for the best hypothesis is often called decoding.

Inferring on Sequence from Another

aws training and certification

We still intend to calculate $P(E;\theta)$, but before doing so we would like to calculate the initial state of the target model based on another RNN over the surce sentence F.

First RNN "encodes" source sentence F, and second RNN “decodes” the initial state into target sentence E.

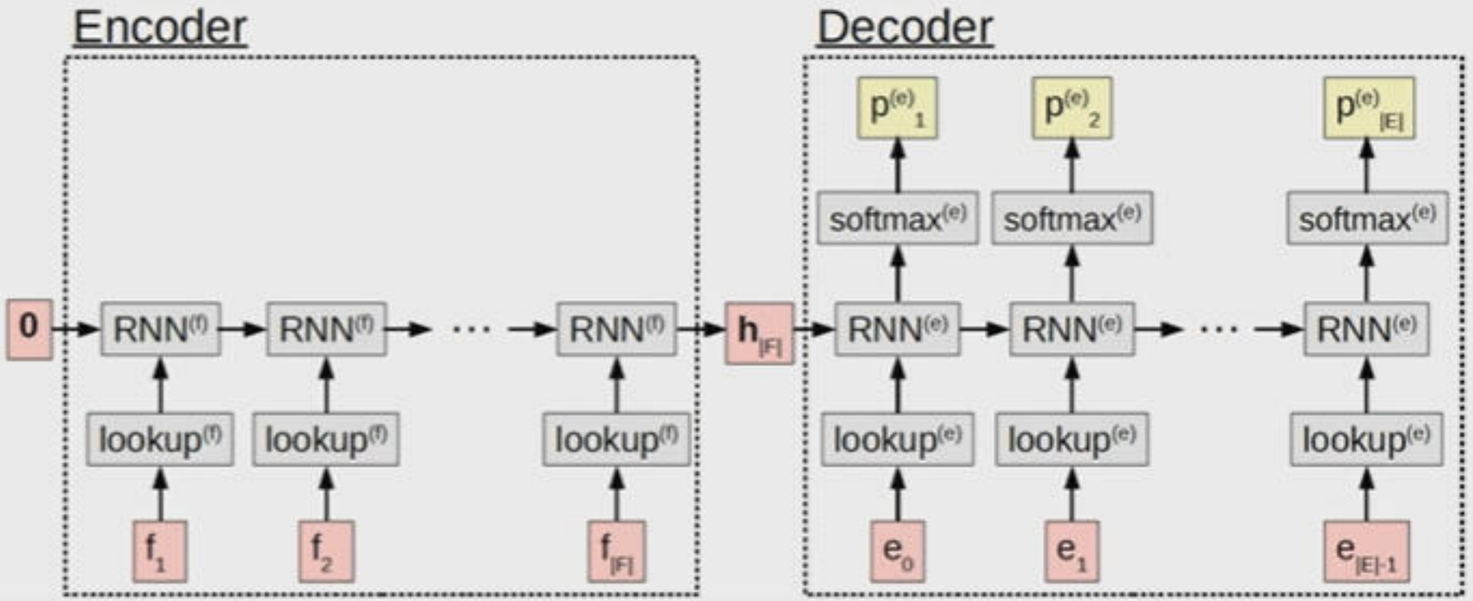


http://opennmt.net/

Encoder-Decoder Architecture

aws training and certification

A model based on two LSTM networks, one that takes a sentence and captures it into an output layer that is to be used as input for another LSTM that performs translation



$$\boldsymbol{m}_t^{(f)} = M_{\cdot,f_t}^{(f)}$$

$$\boldsymbol{h}_t^{(f)} = \begin{cases} \mathrm{RNN}^{(f)}(\boldsymbol{m}_t^{(f)}, \boldsymbol{h}_{t-1}^{(f)}) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{m}_t^{(e)} = M_{\cdot,e_{t-1}}^{(e)}$$

$$\boldsymbol{h}_t^{(e)} = \begin{cases} \mathrm{RNN}^{(e)}(\boldsymbol{m}_t^{(e)}, \boldsymbol{h}_{t-1}^{(e)}) & t \geq 1, \\ \boldsymbol{h}_{|F|}^{(f)} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t^{(e)} = \mathrm{softmax}(W_{hs}\boldsymbol{h}_t^{(e)} + b_s)$$

10 21:35 / 42:33

Generating Output

aws training and certification

- So far we have seen how to make a probability model. In order to perform translation, we need to generate output. Generally this is performed using search in **parallel corpora**, the corpora of target language.

- Search is performed based on several Criteria:
  - **Random Sampling:** Randomly selecting an output E from a probability model. $\hat{E} \sim P(E|F)$
  - **Greedy 1-Best Search:** Finding E that maximizes $P(E|F)$; $\hat{E} = argmax_E\ P\ (E|F)$
  - **Beam Search:** Finding the n options with the highest probability $P\ (E|F)$

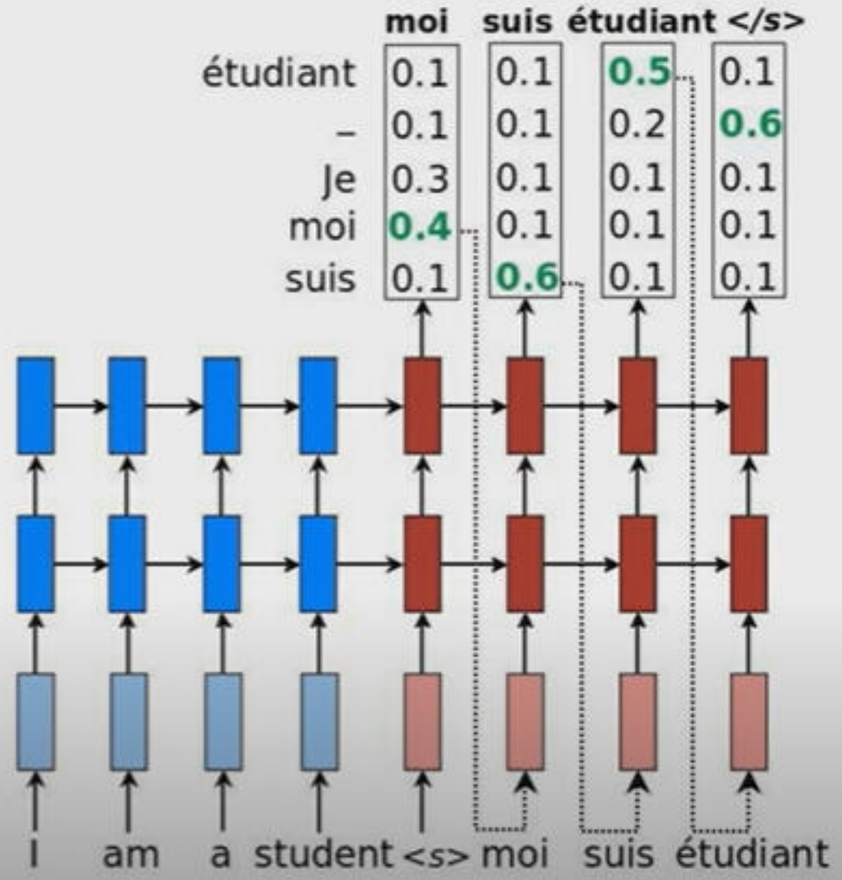
Greedy (Best) Search

aws training and certification

- This is very similar to random sampling with one difference: The goal is to search for the word in target corpora that maximizes probability.

$$\hat{e}_t = argmax_i \ \log(p_{t,i}^{(e)})$$

- Greedy search is not guaranteed to find the translation with highest probability, but is efficient in terms of memory and computation.

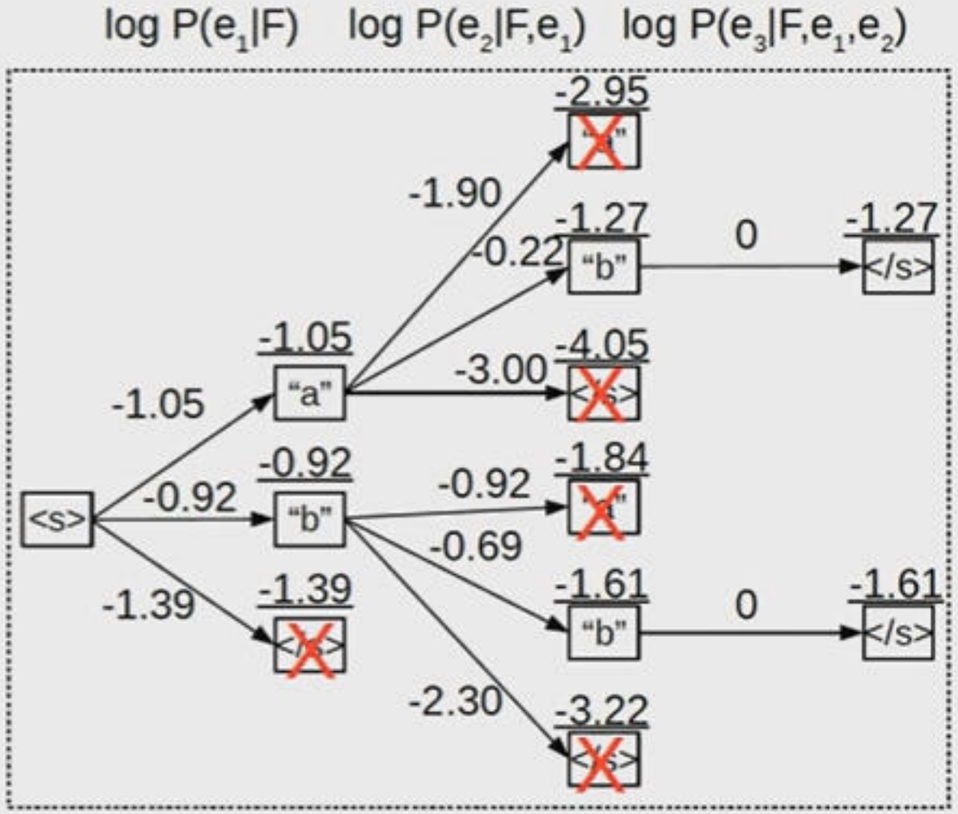


https://github.com/tensorflow/nmt

Beam Search

aws training and certification

- Beam Search is a search method that searches for *b* best hypothesis at each time step. *b* is called width of the beam.
- Larger beam size has a strong length bias.
- *b* is a hyperparameter that should be selected to maximize accuracy.
- Not very efficient.



Limitations

aws training and certification

- Long distance dependencies.
- Storing variable length sentences in fixed size vectors.

He purchased a car, so that he could drive to work and not spend his time on train.

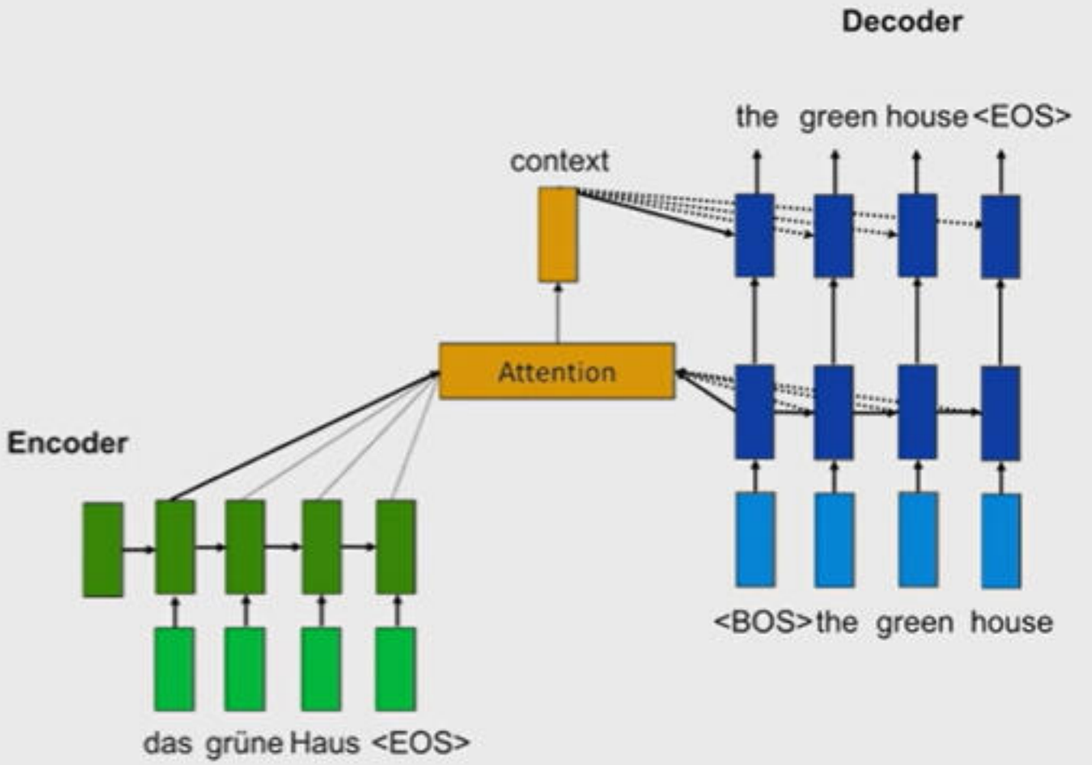
Er kaufte ein Auto, damit er zur Arbeit fahren und seine Zeit nicht im Zug verbringen konnte

Attention

aws training and certification

- The attention model comes between the encoder and the decoder and helps the decoder to pick only the encoded inputs that are important for each step of the decoding process.

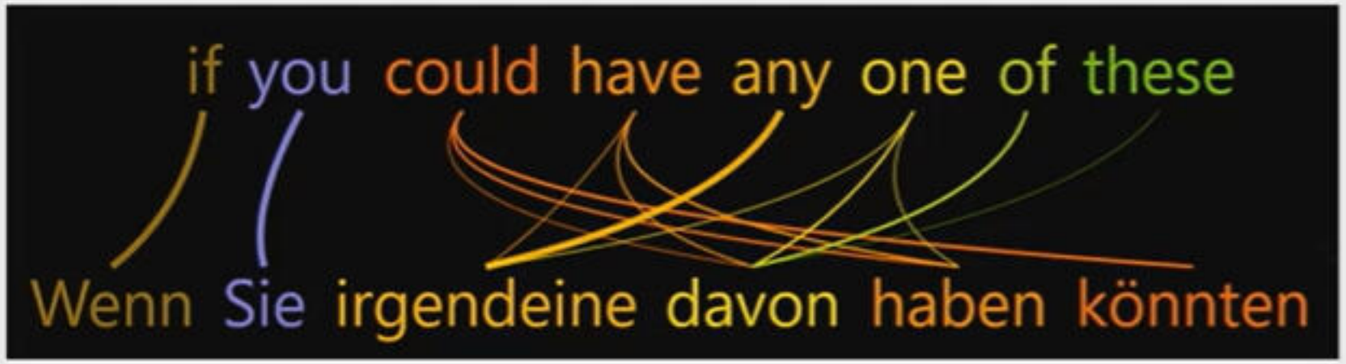
- For each encoded input from the encoder RNN, the attention mechanism calculates its importance.



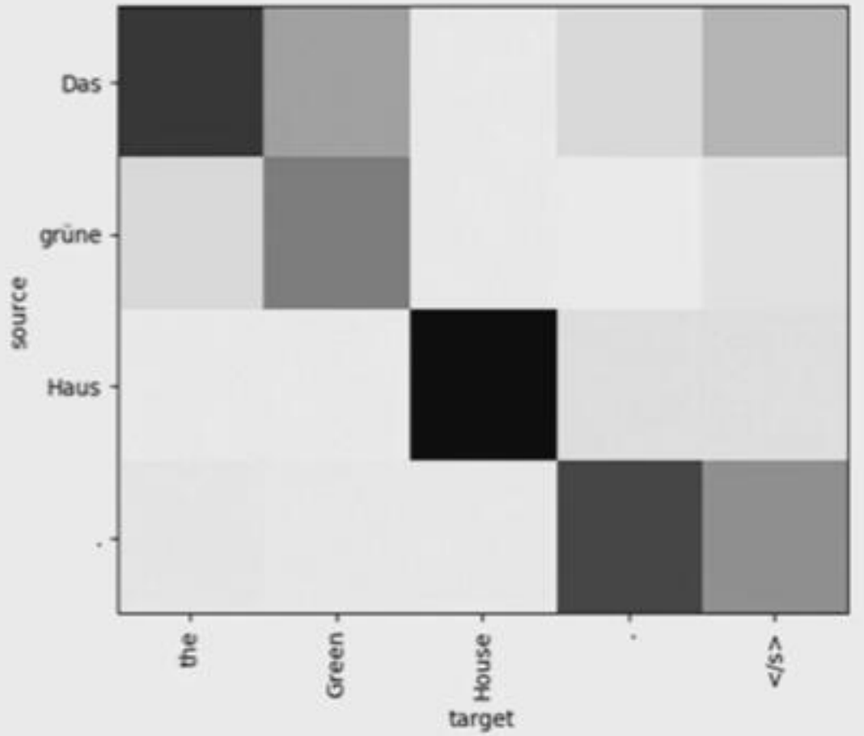
https://aws.amazon.com/blogs/ai/train-neural-machine-translation-models-with-sockeye/

Visualization of Importance

aws training and certification



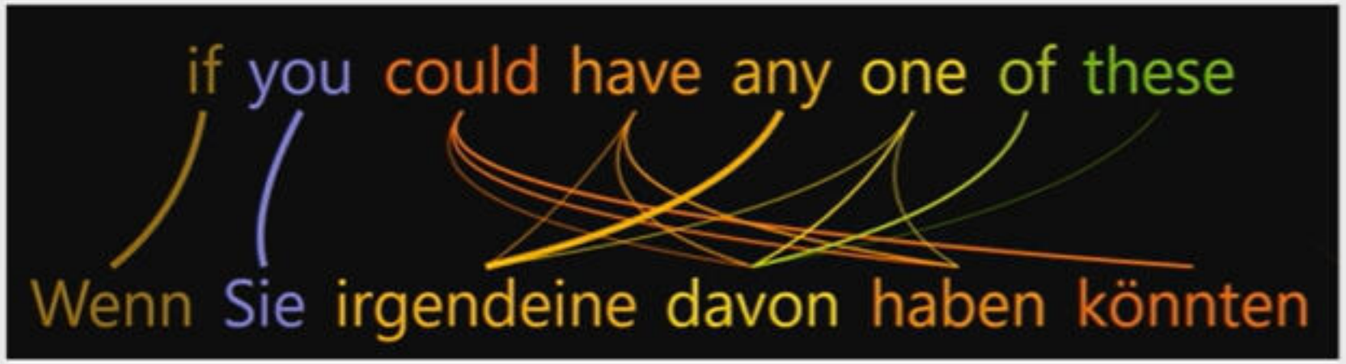
https://aws.amazon.com/blogs/ai/amazon-at-wmt-improving-machine-translation-with-user-feedback/



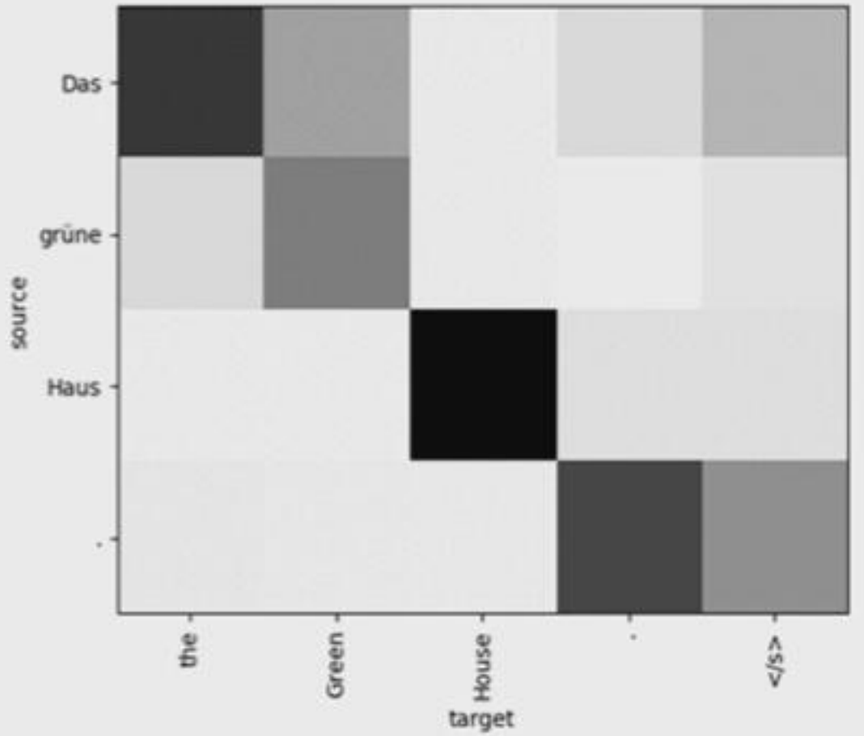
https://aws.amazon.com/blogs/ai/train-neural-machine-translation-models-with-sockeye/

Visualization of Importance

aws training and certification



https://aws.amazon.com/blogs/ai/amazon-at-wmt-improving-machine-translation-with-user-feedback/



https://aws.amazon.com/blogs/ai/train-neural-machine-translation-models-with-sockeye/

Attention Mechanism

aws training and certification

The attention computation happens at every decoder time step. It consists of the following stages:

1. The current target hidden state is compared with all source states to derive *attention weights*.
2. Based on the attention weights we compute a *context vector* as the weighted average of the source states.
3. Combine the context vector with the current target hidden state to yield the final *attention vector*.
4. The attention vector is fed as an input to the next time step (*input feeding*). The first three steps can be summarized by the following equations:

Attention Mechanism Contd.

aws training and certification

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

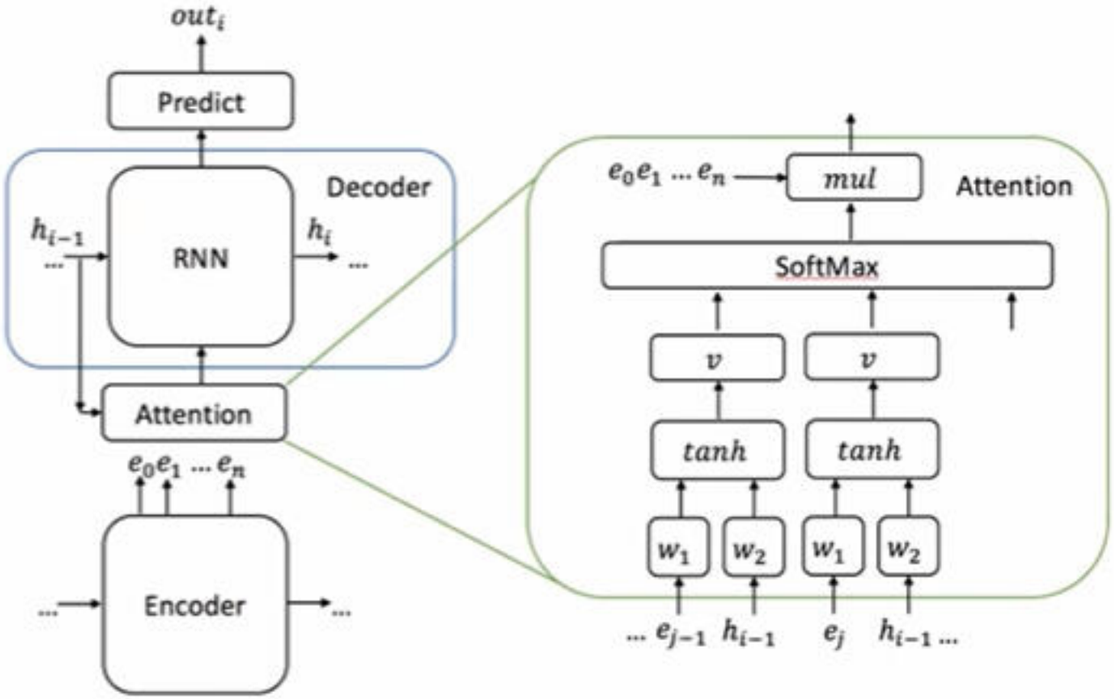
$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

α_{ts}: attention weights

c_t: context vector

α_t = attention vector



https://talbaumel.github.io/attention/

Calculating Attention Score

aws training and certification

Method	Pros	Cons	Formula
Dot Product	- Simple - No additional Parameters	- Input and output encoding in the same space	attn_score $\left(h_j^{(f)}, h_t^{(e)}\right) := h_j^{(f)T} h_t^{(e)}$
Bilinear Functions	- Input and output of different sized	- More expensive than $\otimes$	attn_score $\left(h_j^{(f)}, h_t^{(e)}\right) := h_j^{(f)T} W_a h_t^{(e)}$ W_a *is a linear transformation*
MLP	- More flexible than $\otimes$ - Fewer parameters than BLF		attn_score $\left(h_j^{(f)}, h_t^{(e)}\right) :=$ $w_{a2}^T \tanh\left(W_{a1}[h_t^{(e)}; h_j^{(f)}]\right)$ W_{a1} *and* w_{a2} *are weight matrix and vector of the first and second layers of the MLP respectively*

Translation Performance

aws training and certification

- **BLEU:** The BLEU score measures similarity to human translation. It measures how many words overlap in a *given translation* when compared to a *reference translation*. BLEU scores range from 0-100, the higher the score, the more the translation correlates to a human translation.

- **TER:** The TER score measures the amount of editing that a translator would have to perform to change a translation so it exactly matches a reference translation. By repeating this analysis on a large number of sample translations, it is possible to estimate the post-editing effort required for a project. TER scores also range from 0-100.

Translation Performance Contd.

aws training and certification

criterion	loss	BLEU	TER	NIST
MLE	N/A	30.48	60.85	8.26
MRT	−sBLEU	**36.71**	**53.48**	**8.90**
	sTER	30.14	53.83	6.02
	−sNIST	32.32	56.85	**8.90**

Sockeye Encode-Decoder Framework

aws training and certification

- Sockeye is a sequence-to-sequence framework with attention for Neural Machine Translation based on Apache MXNet Incubating. It implements the well-known encoder-decoder architecture with attention.
- Github repo: https://github.com/awslabs/sockeye
- Tutorials: https://github.com/awslabs/sockeye/tree/master/tutorials

Examples

aws training and certification

- `python3 -m sockeye.train -s train.source \ -t train.target \ -vs dev.source \ -vt dev.target \ --num-embed 32 \ --rnn-num-hidden 64 \ --attention-type dot \ --use-cpu \ --metrics perplexity accuracy \ --max-num-checkpoint-not-improved 3 \ -o seqcopy_model`
- `python -m sockeye.translate -m seqcopy_model`

Sockeye Features

aws training and certification

- Beam search inference
- Easy ensembling of multiple models
- Residual connections between RNN layers (Wu et al., 2016) [Deep LSTM with parallelism]
- Lexical biasing of output layer predictions (Arthur et al., 2016) [Low Frequency Words]
- Modeling coverage (Tu et al., 2016) [Keeping attention history to reduce over and under translation]
- Context gating (Tu et al., 2017) [Improving adequacy of translation by controlling rations of source and target context]

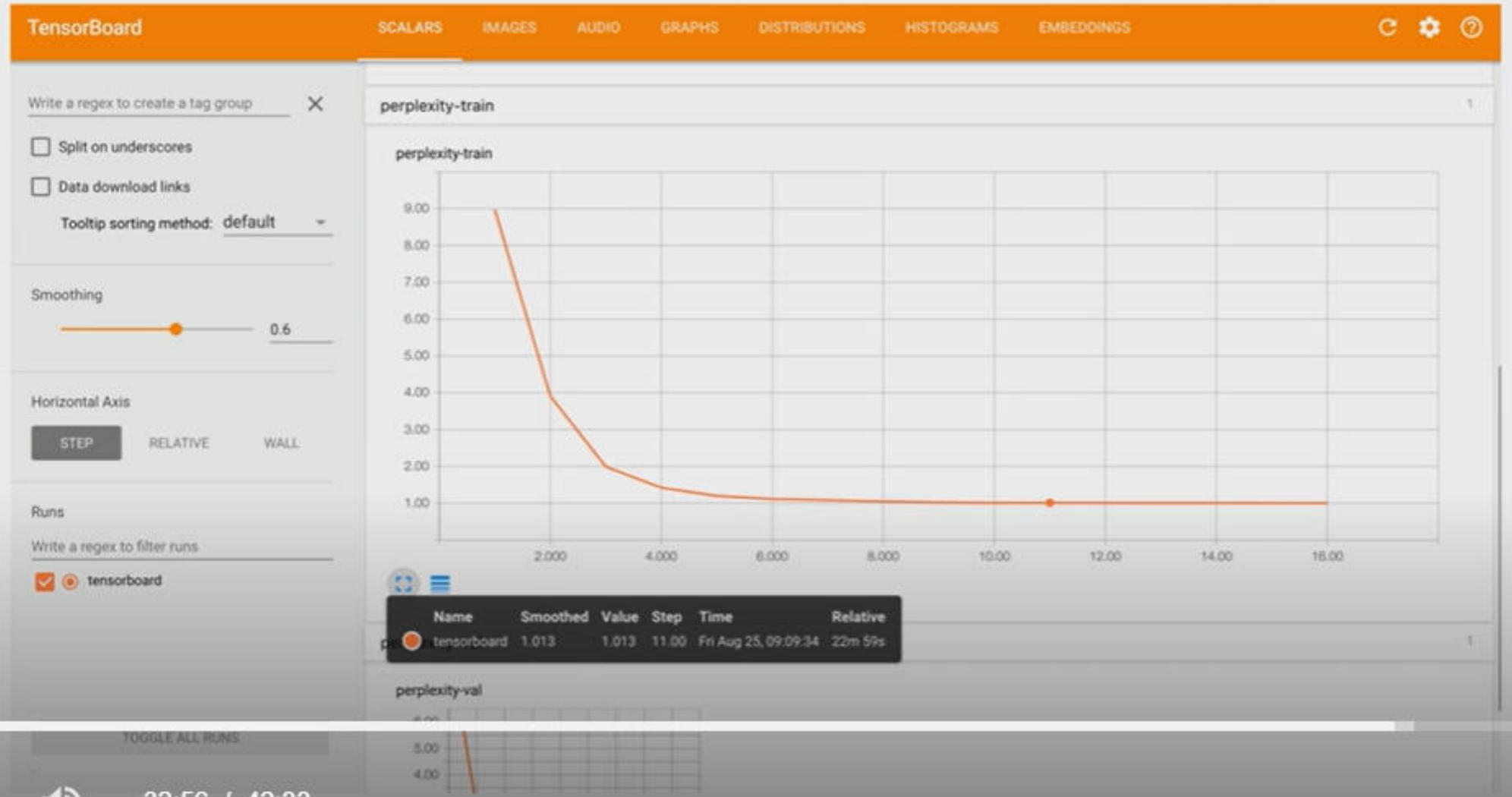
Sockeye Features Contd.

aws training and certification

- Cross-entropy label smoothing (e.g., Pereyra et al., 2017)
- Layer normalization (Ba et al, 2016) [improving training time]
- Multiple supported attention mechanisms [dot, mlp, bilinear, multihead-dot, encoder last state, location]
- Multiple model architectures (Encoder-Decoder Wu et al., 2016, Convolutional Gehring et al, 2017, Transformer Vaswani et al, 2017,)

Metrics and Visualization

aws training and certification

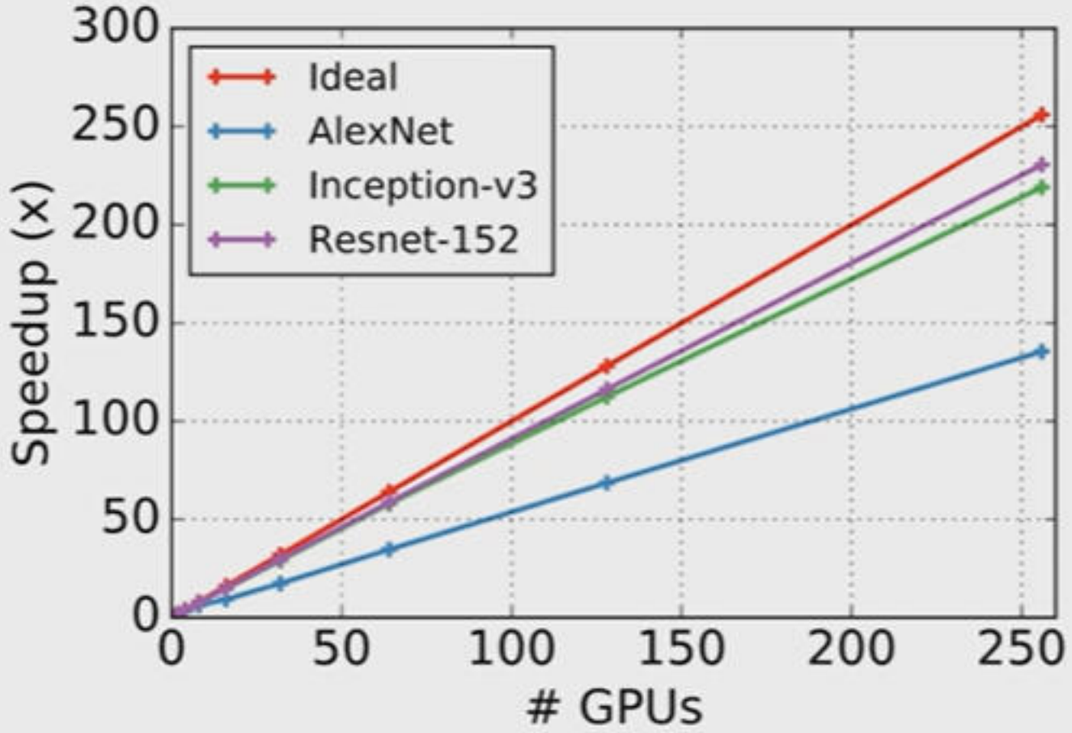


10 32:56 / 42:33

MXNet

aws training and certification

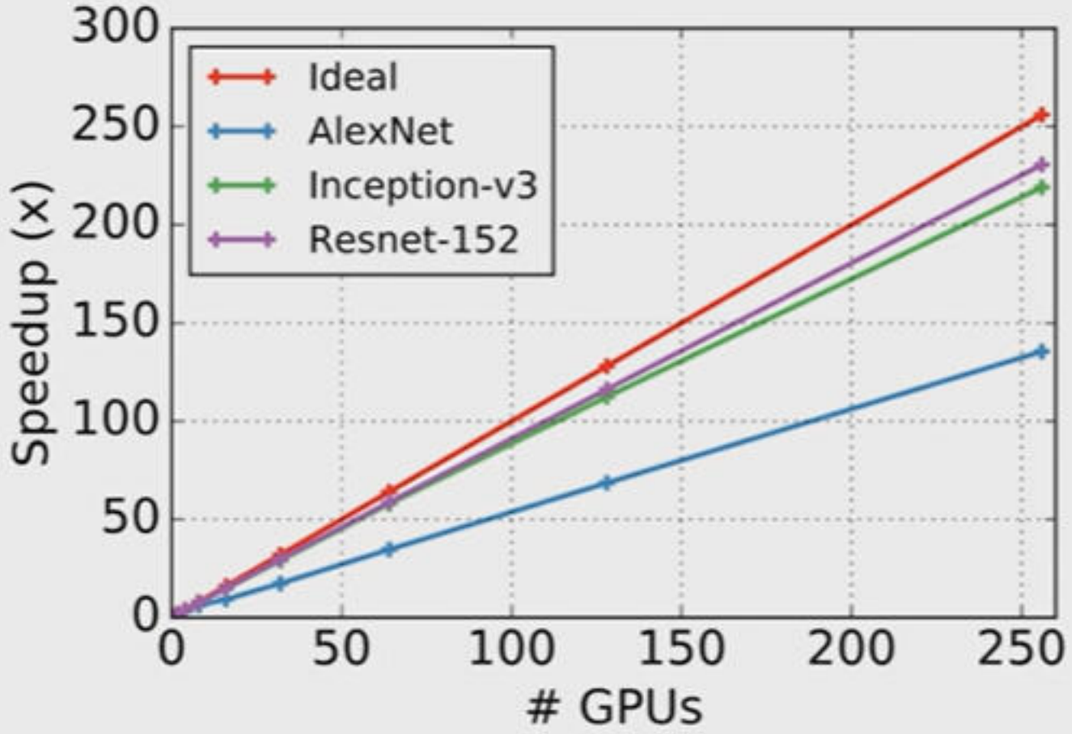
- Sockeye uses MXNet as an underlying deep learning framework.
- MXNet benchmark shows near linear performance across multiple machines.



MXNet

aws training and certification

- Sockeye uses MXNet as an underlying deep learning framework.
- MXNet benchmark shows near linear performance across multiple machines.



Multi-GPU Training

aws training and certification

Training can be carried out on multiple GPUs by either specifying multiple GPU device ids: --device-ids 0 1 2 3, or specifying the number GPUs required: --device-ids -n attempts to acquire n GPUs.

```
Every 2.0s: nvidia-smi

Wed Oct 11 22:28:25 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.66                 Driver Version: 375.66                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 0000:00:17.0     Off |                    0 |
| N/A   59C    P0    97W / 149W |    998MiB / 11439MiB |     84%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           On   | 0000:00:18.0     Off |                    0 |
| N/A   48C    P0   106W / 149W |    826MiB / 11439MiB |     80%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           On   | 0000:00:19.0     Off |                    0 |
| N/A   60C    P0    92W / 149W |    755MiB / 11439MiB |     83%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           On   | 0000:00:1A.0     Off |                    0 |
| N/A   52C    P0   114W / 149W |    577MiB / 11439MiB |     81%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla K80           On   | 0000:00:1B.0     Off |                    0 |
| N/A   60C    P0    92W / 149W |    555MiB / 11439MiB |     76%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla K80           On   | 0000:00:1C.0     Off |                    0 |
| N/A   48C    P0   107W / 149W |    546MiB / 11439MiB |     83%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla K80           On   | 0000:00:1D.0     Off |                    0 |
| N/A   63C    P0    92W / 149W |    547MiB / 11439MiB |     56%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla K80           On   | 0000:00:1E.0     Off |                    0 |
| N/A   51C    P0   103W / 149W |    536MiB / 11439MiB |     63%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     55844    C   python                                       994MiB |
|    1     55844    C   python                                       822MiB |
|    2     55844    C   python                                       751MiB |
|    3     55844    C   python                                       573MiB |
|    4     55844    C   python                                       551MiB |
|    5     55844    C   python                                       542MiB |
|    6     55844    C   python                                       543MiB |
|    7     55844    C   python                                       532MiB |
+-----------------------------------------------------------------------------+
```

10 33:45 / 42:33

Amazon SageMaker

aws training and certification

- A fully managed service that enables data scientists and developers to quickly and easily build production-ready machine-learning models.
- Amazon SageMaker includes several built-in state of the art algorithms that are fine-tuned to run on distributed environments. https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html
- Amongst built-in algorithms there is a Sequence2Sequence algorithm implemented using Sockeye.
- All you need to do it to tune model's hyperparameters and passing your data to the model.
- Another useful built-in algorithm is Amazon BlazingText and provides Word2Vec.

In [7]:
```
%%bash
# python3 create_vocab_proto.py -h
```

The cell below does the preprocessing. If you are using the complete dataset, the script might take around 10-15 min on an m4.xlarge notebook instance. Remove ".small" from the file names for training on full datasets.

In [8]:
```
%%time
%%bash
python3 create_vocab_proto.py \
        --train-source corpus.tc.en.small \
        --train-target corpus.tc.de.small \
        --val-source validation/newstest2014.tc.en \
        --val-target validation/newstest2014.tc.de
```

```
INFO:__main__:Building vocabulary from dataset: corpus.tc.en.small and corpus.tc.de.small
INFO:__main__:Final vocabulary: 10661 types (min frequency 1, top 50000 types)
INFO:__main__:Final vocabulary: 15980 types (min frequency 1, top 50000 types)
INFO:__main__:Source vocabulary size: 10661
INFO:__main__:Vocabulary saved to "vocab.src.json"
INFO:__main__:Target vocabulary size: 15980
INFO:__main__:Vocabulary saved to "vocab.trg.json"
INFO:__main__:Spawning 31 encoding worker(s) for encoding train datasets!
INFO:__main__:Processed 10000 lines for encoding to protobuf. 0 lines were ignored as they didn't have
                any content in either the source or the target file!
INFO:__main__:Completed writing the encoding queue!
INFO:__main__:Encoding finished! Writing records to "train.rec"
INFO:__main__:Processed input and saved to "train.rec"
INFO:__main__:Spawning 31 encoding worker(s) for encoding validation datasets!
INFO:__main__:Processed 3003 lines for encoding to protobuf. 0 lines were ignored as they didn't have
                any content in either the source or the target file!
INFO:__main__:Completed writing the encoding queue!
INFO:__main__:Encoding finished! Writing records to "val.rec"
INFO:__main__:Processed input and saved to "val.rec"
```

```
CPU times: user 0 ns, sys: 4 ms, total: 4 ms
Wall time: 1.8 s
```


The script will output 4 files, namely:

- train.rec : Contains source and target sentences for training in protobuf format

```
INFO:__main__:Spawning 31 encoding worker(s) for encoding validation datasets!
INFO:__main__:Processed 3003 lines for encoding to protobuf. 0 lines were ignored as they didn't have
              any content in either the source or the target file!
INFO:__main__:Completed writing the encoding queue!
INFO:__main__:Encoding finished! Writing records to "val.rec"
INFO:__main__:Processed input and saved to "val.rec"
```

```
CPU times: user 0 ns, sys: 4 ms, total: 4 ms
Wall time: 1.8 s
```

The script will output 4 files, namely:

- train.rec : Contains source and target sentences for training in protobuf format
- val.rec : Contains source and target sentences for validation in protobuf format
- vocab.src.json : Vocabulary mapping (string to int) for source language (English in this example)
- vocab.trg.json : Vocabulary mapping (string to int) for target language (German in this example)

Let's upload the pre-processed dataset and vocabularies to S3

In [11]:
```
def upload_to_s3(bucket, prefix, channel, file):
    s3 = boto3.resource('s3')
    data = open(file, "rb")
    key = prefix + "/" + channel + '/' + file
    s3.Bucket(bucket).put_object(Key=key, Body=data)

upload_to_s3(bucket, prefix, 'train', 'train.rec')
upload_to_s3(bucket, prefix, 'validation', 'val.rec')
upload_to_s3(bucket, prefix, 'vocab', 'vocab.src.json')
upload_to_s3(bucket, prefix, 'vocab', 'vocab.trg.json')
```

In [12]:
```
region_name = boto3.Session().region_name
```

In [13]:
```
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/seq2seq:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/seq2seq:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/seq2seq:latest'}
container = containers[region_name]
print('Using SageMaker Seq2Seq container: {} ({})'.format(container, region_name))
```

```
Using SageMaker Seq2Seq container: 685385470294.dkr.ecr.eu-west-1.amazonaws.com/seq2seq:latest (eu-west-1)
```

- train.rec : Contains source and target sentences for training in protobuf format
- val.rec : Contains source and target sentences for validation in protobuf format
- vocab.src.json : Vocabulary mapping (string to int) for source language (English in this example)
- vocab.trg.json : Vocabulary mapping (string to int) for target language (German in this example)

Let's upload the pre-processed dataset and vocabularies to S3

In [11]:
```
def upload_to_s3(bucket, prefix, channel, file):
    s3 = boto3.resource('s3')
    data = open(file, "rb")
    key = prefix + "/" + channel + '/' + file
    s3.Bucket(bucket).put_object(Key=key, Body=data)

upload_to_s3(bucket, prefix, 'train', 'train.rec')
upload_to_s3(bucket, prefix, 'validation', 'val.rec')
upload_to_s3(bucket, prefix, 'vocab', 'vocab.src.json')
upload_to_s3(bucket, prefix, 'vocab', 'vocab.trg.json')
```

In [12]:
```
region_name = boto3.Session().region_name
```

In [13]:
```
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/seq2seq:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/seq2seq:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/seq2seq:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/seq2seq:latest'}
container = containers[region_name]
print('Using SageMaker Seq2Seq container: {} ({})'.format(container, region_name))
```

```
Using SageMaker Seq2Seq container: 685385470294.dkr.ecr.eu-west-1.amazonaws.com/seq2seq:latest (eu-west-1)
```

Training the Machine Translation model

In [14]:
```
job_name = 'DEMO-seq2seq-en-de-' + strftime("%Y-%m-%d-%H", gmtime())
print("Training job", job_name)

create_training_params = \
{
    "AlgorithmSpecification": {
        "TrainingImage": container,
```

In [14]:

```
job_name = 'DEMO-seq2seq-en-de-' + strftime("%Y-%m-%d-%H", gmtime())
print("Training job", job_name)

create_training_params = \
{
    "AlgorithmSpecification": {
        "TrainingImage": container,
        "TrainingInputMode": "File"
    },
    "RoleArn": role,
    "OutputDataConfig": {
        "S3OutputPath": "s3://{}/{}/".format(bucket, prefix)
    },
    "ResourceConfig": {
        # Seq2Seq does not support multiple machines. Currently, it only supports single machine, multiple GPUs
        "InstanceCount": 1,
        "InstanceType": "ml.p3.8xlarge", # We suggest one of ["ml.p2.16xlarge", "ml.p2.8xlarge", "ml.p2.xlarge"]
        "VolumeSizeInGB": 50
    },
    "TrainingJobName": job_name,
    "HyperParameters": {
        # Please refer to the documentation for complete list of parameters
        "max_seq_len_source": "60",
        "max_seq_len_target": "60",
        "optimized_metric": "bleu",
        "batch_size": "64", # Please use a larger batch size (256 or 512) if using ml.p2.8xlarge or ml.p2.16xlarge
        "checkpoint_frequency_num_batches": "1000",
        "rnn_num_hidden": "512",
        "num_layers_encoder": "1",
        "num_layers_decoder": "1",
        "num_embed_source": "512",
        "num_embed_target": "512",
        "checkpoint_threshold": "3",
        "max_num_batches": "2100"
        # Training will stop after 2100 iterations/batches.
        # This is just for demo purposes. Remove the above parameter if you want a better model.
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 48 * 3600
    },
    "InputDataConfig": [
```

```
primary_container = {
    'Image': container,
    'ModelDataUrl': model_data
}

create_model_response = sage.create_model(
    ModelName = model_name,
    ExecutionRoleArn = role,
    PrimaryContainer = primary_container)

print(create_model_response['ModelArn'])
```

```
DEMO-seq2seq-en-de-2018-05-03-23
s3://cyrusmv-sagemaker-demos/sagemaker/DEMO-seq2seq/DEMO-seq2seq-en-de-2018-05-03-23/output/model.tar.gz
arn:aws:sagemaker:eu-west-1:[redacted]:model/demo-seq2seq-en-de-2018-05-03-23
CPU times: user 16 ms, sys: 0 ns, total: 16 ms
Wall time: 496 ms
```

Create endpoint configuration

Use the model to create an endpoint configuration. The endpoint configuration also contains information about the type and number of EC2 instances to use when hosting the model.

Since SageMaker Seq2Seq is based on Neural Nets, we could use an ml.p2.xlarge (GPU) instance, but for this example we will use a free tier eligible ml.m4.xlarge.

In [19]:
```
from time import gmtime, strftime

endpoint_config_name = 'DEMO-Seq2SeqEndpointConfig-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = sage.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InitialInstanceCount':1,
        'ModelName':model_name,
        'VariantName':'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

Create endpoint configuration

Use the model to create an endpoint configuration. The endpoint configuration also contains information about the type and number of EC2 instances to use when hosting the model.

Since SageMaker Seq2Seq is based on Neural Nets, we could use an ml.p2.xlarge (GPU) instance, but for this example we will use a free tier eligible ml.m4.xlarge.

In [19]:
```
from time import gmtime, strftime

endpoint_config_name = 'DEMO-Seq2SeqEndpointConfig-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = sage.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType':'ml.m4.xlarge',
        'InitialInstanceCount':1,
        'ModelName':model_name,
        'VariantName':'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

```
DEMO-Seq2SeqEndpointConfig-2018-05-03-23-46-48
Endpoint Config Arn: arn:aws:sagemaker:eu-west-1:[redacted]:endpoint-config/demo-seq2seqendpointconfig-2018-05-03-23-46-48
```

Create endpoint

Lastly, we create the endpoint that serves up model, through specifying the name and configuration defined above. The end result is an endpoint that can be validated and incorporated into production applications. This takes 10-15 minutes to complete.

In [20]:
```
%%time
import time

endpoint_name = 'DEMO-Seq2SeqEndpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = sage.create_endpoint(
```

```
        'InitialInstanceCount':1,
        'ModelName':model_name,
        'VariantName':'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

```
DEMO-Seq2SeqEndpointConfig-2018-05-03-23-46-48
Endpoint Config Arn: arn:aws:sagemaker:eu-west-1:[redacted]:endpoint-config/demo-seq2seqendpointconfig-2018-05-03-2
3-46-48
```

Create endpoint

Lastly, we create the endpoint that serves up model, through specifying the name and configuration defined above. The end result is an endpoint that can be validated and incorporated into production applications. This takes 10-15 minutes to complete.

In [20]:

```
%%time
import time

endpoint_name = 'DEMO-Seq2SeqEndpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = sage.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name)
print(create_endpoint_response['EndpointArn'])

resp = sage.describe_endpoint(EndpointName=endpoint_name)
status = resp['EndpointStatus']
print("Status: " + status)

# wait until the status has changed
sage.get_waiter('endpoint_in_service').wait(EndpointName=endpoint_name)

# print the status of the endpoint
endpoint_response = sage.describe_endpoint(EndpointName=endpoint_name)
status = endpoint_response['EndpointStatus']
print('Endpoint creation ended with EndpointStatus = {}'.format(status))

if status != 'InService':
    raise Exception('Endpoint creation failed.')
```

```
DEMO-Seq2SeqEndpoint-2018-05-03-23-46-52
```

Create endpoint

Lastly, we create the endpoint that serves up model, through specifying the name and configuration defined above. The end result is an endpoint that can be validated and incorporated into production applications. This takes 10-15 minutes to complete.

In [20]:
```
%%time
import time

endpoint_name = 'DEMO-Seq2SeqEndpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = sage.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name)
print(create_endpoint_response['EndpointArn'])

resp = sage.describe_endpoint(EndpointName=endpoint_name)
status = resp['EndpointStatus']
print("Status: " + status)

# wait until the status has changed
sage.get_waiter('endpoint_in_service').wait(EndpointName=endpoint_name)

# print the status of the endpoint
endpoint_response = sage.describe_endpoint(EndpointName=endpoint_name)
status = endpoint_response['EndpointStatus']
print('Endpoint creation ended with EndpointStatus = {}'.format(status))

if status != 'InService':
    raise Exception('Endpoint creation failed.')
```

```
DEMO-Seq2SeqEndpoint-2018-05-03-23-46-52
arn:aws:sagemaker:eu-west-1:[redacted]:endpoint/demo-seq2seqendpoint-2018-05-03-23-46-52
Status: Creating
Endpoint creation ended with EndpointStatus = InService
CPU times: user 56 ms, sys: 0 ns, total: 56 ms
Wall time: 6min 31s
```

If you see the message,

> Endpoint creation ended with EndpointStatus = InService

then congratulations! You now have a functioning inference endpoint. You can confirm the endpoint configuration and status by navigating to the "Endpoints" tab in the AWS SageMaker console.

We will finally create a runtime object from which we can invoke the endpoint.

In [21]:
```
runtime = boto3.client(service_name='runtime.sagemaker')
```

Perform Inference

Using JSON format for inference (Suggested for a single or small number of data instances)

Note that you don't have to convert string to text using the vocabulary mapping for inference using JSON mode

In [22]:
```
sentences = ["you are so good !",
             "can you drive a car ?",
             "i want to watch a movie ."
            ]

payload = {"instances" : []}
for sent in sentences:
    payload["instances"].append({"data" : sent})

response = runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType='application/json',
                                   Body=json.dumps(payload))

response = response["Body"].read().decode("utf-8")
response = json.loads(response)
print(response)
```

```
{'predictions': [{'target': 'COR / 08 / 08 / 149'}, {'target': 'wir werden .'}, {'target': 'Präsident Maystadt sollte
n sich : &amp;quot;'}]}
```

Retrieving the Attention Matrix

Perform Inference

Using JSON format for inference (Suggested for a single or small number of data instances)

Note that you don't have to convert string to text using the vocabulary mapping for inference using JSON mode

In [22]:
```
sentences = ["you are so good !",
             "can you drive a car ?",
             "i want to watch a movie ."
            ]

payload = {"instances" : []}
for sent in sentences:
    payload["instances"].append({"data" : sent})

response = runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType='application/json',
                                   Body=json.dumps(payload))

response = response["Body"].read().decode("utf-8")
response = json.loads(response)
print(response)
```

```
{'predictions': [{'target': 'COR / 08 / 08 / 149'}, {'target': 'wir werden .'}, {'target': 'Präsident Maystadt sollte
n sich : &amp;quot;'}]}
```

Retrieving the Attention Matrix

Passing `"attention_matrix":"true"` in `configuration` of the data instance will return the attention matrix.

In [23]:
```
sentence = 'can you drive a car ?'

payload = {"instances" : [{
```

```
{'predictions': [{'target': 'COR / 08 / 08 / 149'}, {'target': 'wir werden .'}, {'target': 'Präsident Maystadt sollte
n sich : &amp;quot;'}]}
```

Retrieving the Attention Matrix

Passing `"attention_matrix":"true"` in `configuration` of the data instance will return the attention matrix.

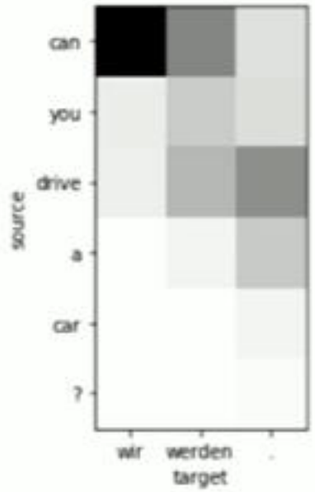
In [23]:
```
sentence = 'can you drive a car ?'

payload = {"instances" : [{
                           "data" : sentence,
                           "configuration" : {"attention_matrix":"true"}
                         }
                        ]}

response = runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType='application/json',
                                   Body=json.dumps(payload))

response = response["Body"].read().decode("utf-8")
response = json.loads(response)['predictions'][0]

source = sentence
target = response["target"]
attention_matrix = np.array(response["matrix"])

print("Source: %s \nTarget: %s" % (source, target))
```

```
Source: can you drive a car ?
Target: wir werden .
```

In [24]:
```
# Define a function for plotting the attentioan matrix
def plot_matrix(attention_matrix, target, source):
    source_tokens = source.split()
    target_tokens = target.split()
    assert attention_matrix.shape[0] == len(target_tokens)
    plt.imshow(attention_matrix.transpose(), interpolation="nearest", cmap="Greys")
    plt.xlabel("target")
    plt.ylabel("source")
```

```
    plt.xlabel("target")
    plt.ylabel("source")
    plt.gca().set_xticks([i for i in range(0, len(target_tokens))])
    plt.gca().set_yticks([i for i in range(0, len(source_tokens))])
    plt.gca().set_xticklabels(target_tokens)
    plt.gca().set_yticklabels(source_tokens)
    plt.tight_layout()
```

In [25]:
```
plot_matrix(attention_matrix, target, source)
```



Using Protobuf format for inference (Suggested for efficient bulk inference)

Reading the vocabulary mappings as this mode of inference accepts list of integers and returns list of integers.

In [26]:
```
import io
import tempfile
from record_pb2 import Record
from create_vocab_proto import vocab_from_json, reverse_vocab, write_recordio, list_to_record_bytes, read_next

source = vocab_from_json("vocab.src.json")
target = vocab_from_json("vocab.trg.json")
```

Documentation - This Guide
Search

- What Is Amazon SageMaker?
- How It Works
- Getting Started
- Using Built-in Algorithms
  - Common Information
  - Linear Learner
  - Factorization Machines
  - XGBoost Algorithm
  - Image Classification Algorithm
  - Sequence to Sequence (seq2seq)
    - How It Works
    - **Hyperparameters**
  - K-Means Algorithm
  - Principal Component Analysis (PCA)
  - Latent Dirichlet Allocation (LDA)
  - Neural Topic Model (NTM)
  - DeepAR Forecasting
  - BlazingText
  - Random Cut Forest
- Using Your Own Algorithms
- Automatically Scaling Amazon SageMaker Models
- Using TensorFlow
- Using Apache MXNet

Sequence2Sequence Hyperparameters

Parameter Name	Description
`max_seq_len_source`	Maximum length for the source sequence length. Sequences longer than this length are truncated to this length. Valid values: positive integer Default value: 100
`max_seq_len_target`	Maximum length for the target sequence length. Sequences longer than this length are truncated to this length. Valid values: positive integer Default value: 100
`encoder_type`	Encoder type. The *rnn* architecture is based on attention mechanism by Bahdanau et al. and *cnn* architecture is based on Gehring et al. Valid values: String. Either *rnn* or *cnn*. Default value: *rnn*
`decoder_type`	Decoder type. Valid values: String. Either *rnn* or *cnn*. Default value: *rnn*
`num_layers_encoder`	Number of layers for Encoder *rnn* or *cnn*. Valid values: positive integer Default value: 1
`num_layers_decoder`	Number of layers for Decoder *rnn* or *cnn*. Valid values: positive integer Default value: 1
`rnn_num_hidden`	The number of *rnn* hidden units for encoder and decoder. This must be a multiple of 2 because the algorithm uses bi-directional LSTM by default. Valid values: positive integer

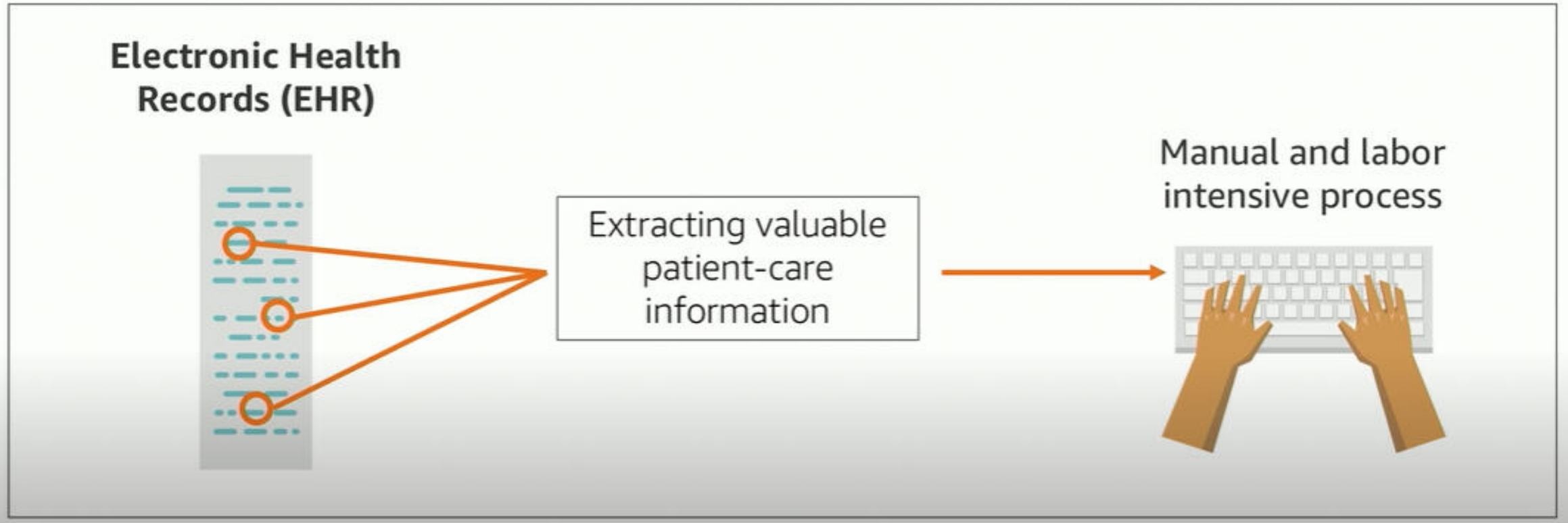
	Valid values: positive integer Default value: 5
`cnn_num_hidden`	Number of *cnn* hidden units for encoder and decoder. Valid values: positive integer Default value: 512
`cnn_activation_type`	The *cnn* activation type to be used. Valid values: String. One of *glu*, *relu*, *softrelu*, *sigmoid*, or *tanh*. Default value: *glu*
`cnn_hidden_dropout`	Dropout probability for dropout between convolutional layers. Valid values: Float. Range in [0,1]. Default value: 0
`num_embed_source`	Embedding size for source tokens. Valid values: positive integer Default value: 512
`num_embed_target`	Embedding size for target tokens. Valid values: positive integer Default value: 512
`embed_dropout_source`	Dropout probability for source side embeddings. Valid values: Float. Range in [0,1]. Default value: 0
`embed_dropout_target`	Dropout probability for target side embeddings. Valid values: Float. Range in [0,1]. Default value: 0
`rnn_attention_type`	Attention model for encoders. *mlp* refers to concat and bilinear refers to general from the Luong et al. paper.

References

- Neubig, G. (2017). *Neural Machine Translation and Sequence-to-Sequence Models: A Tutorial. Carnegie Mellon University: Language Technologies Institute.*
- Cliffton, A., Denkowski, M., Domhan, T., Hieber, F., Post, M., Sokolov, A., & Vilar, D. (2017). Sockeye Documentation. http://sockeye.readthedocs.io/en/latest/index.html
- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C.,…Zheng, X. (2015) Vector Representations of Words. *Tutorials.*
- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C.,…Zheng, X. (2015) Vector Representations of Words. *Github.*
- Weisstein, E. (2018). Wolfram MathWorld. *Affine Transformation.*
- Klein, G., Kim, Y., Deng, Y., Senellart, J. & Rush, A. (2018). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints.*
- Baumel, T. (2018). Sequence to Sequence Attention Models in DyNet. *Github.*

What problems exist that Comprehend Medical can solve?

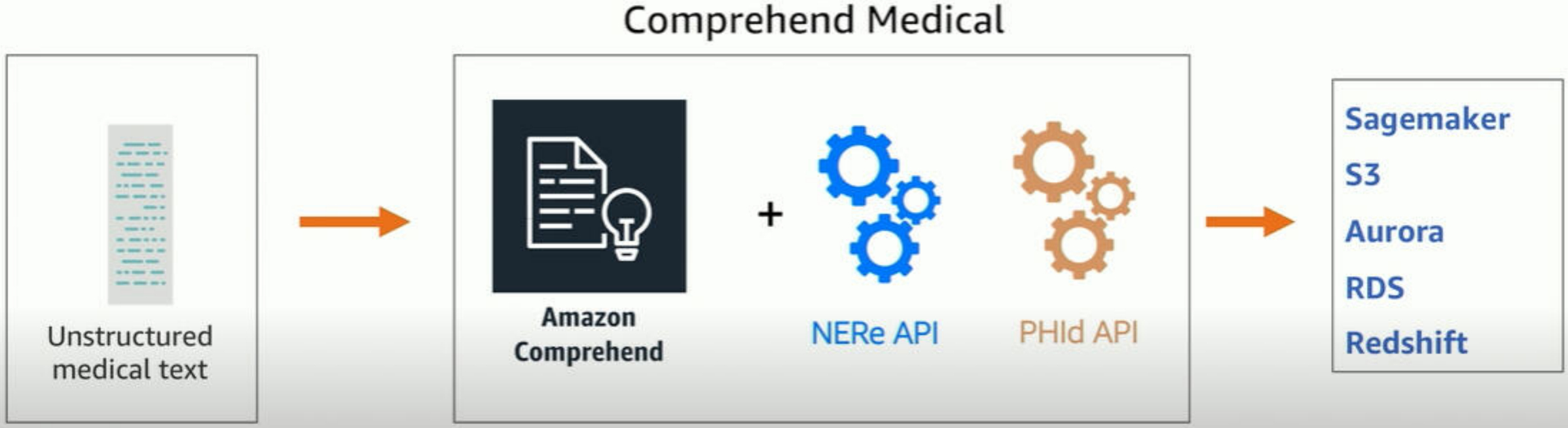
aws training and certification



01:08 / 07:26

How does Comprehend Medical Work?

aws training and certification



01:32 / 07:26

Comprehend

https://us-west-2.console.aws.amazon.com/comprehend/v2/home?region=us-west-2#try-comprehend-medical

aws Services Resource Groups Welcome to AWS N. Virginia Support

Amazon Comprehend

- Real-time analysis
- Job management
- Customization
  - Custom classification
- Comprehend Medical Beta
  - Real-time analysis

Comprehend Medical > Real-time analysis

Real-time analysis Beta

See how Comprehend Medical recognizes entities related to the healthcare domain. To analyze your text, type or paste it in the text box.

Input text

GENERAL : Well-developed , well-nourished male in no acute distress .
HEENT : Oropharynx clear . Sclerae are anicteric .
CARDIOVASCULAR : Regular rate and rhythm . No murmurs , rubs , or gallops .
CHEST : Lungs CTA bilaterally .
ABDOMEN : Benign .
EXTREMITIES : No clubbing , cyanosis , or edema .

1309 of 20000 characters used.

Clear text Analyze

Insights

Analyzed text

LABAHN HOSPITAL NOTE
• Address (LABAHN HOSPITAL)

HISTORY OF PRESENT ILLNESS Mr. Rickie Cresta is a 26 - year - old male with refractory ITP .
• Name (Rickie Cresta) • Age (26) • Dx name (refractory ITP) — Diagnosis

INTERVAL HISTORY Mr. Burl Bickler was last seen on July 11th 2016 , with Dr. Menzel Eplett . Mr. Rubin Corrado has
• Name (Burl Bickler) • Date (July 11th 2016) • Name (Menzel Eplett) • Name (Rubin Corrado)

since started on romiplostim on May 13th 2007 . Despite the gradual dose increases in romiplostim ,
• Generic name (romiplostim) • Date (May 13th 2007) • Generic name (romiplostim)

Test value

his platelet count has varied significantly , ranging anywhere from 5 to almost 60,000 . Recently Mr.

10 03:26 / 07:26

Comprehend

https://us-west-2.console.aws.amazon.com/comprehend/v2/home?region=us-west-2#try-comprehend-medical

aws Services Resource Groups Welcome to AWS N. Virginia Support

Amazon Comprehend

- Real-time analysis
- Job management
- Customization
  - Custom classification
- Comprehend Medical Beta
  - Real-time analysis

1309 of 20000 characters used.

Clear text Analyze

Insights

Analyzed text

PHYSICAL EXAMINATION
- Test name (PHYSICAL EXAMINATION)

Test value / Test value / Test value

VITAL SIGNS : **Temperature** is **36** . **pulse** of **80** . **blood pressure** **112 / 60** .
- VITAL SIGNS: Test name (VITAL SIGNS)
- Temperature: Test name (Temperature)
- 36: Test value (36)
- pulse: Test name (pulse)
- 80: Test value (80)
- blood pressure: Test name (blood pressure)
- 112 / 60: Test value (112 / 60)

Sign / Sign / Negation, Sign

GENERAL : **Well-developed** , **well-nourished** male in no **acute** **distress** .
- GENERAL: System organ site (GENERAL)
- Well-developed: Dx name (Well-developed)
- well-nourished: Dx name (well-nourished)
- acute: Acuity (acute)
- distress: Dx name (distress)

Sign / Sign

HEENT : **Oropharynx clear** . **Sclerae are anicteric** .
- HEENT: System organ site (HEENT)
- Oropharynx clear: System organ site (Oropharynx)
- Oropharynx clear: Dx name (Oropharynx clear)
- Sclerae are anicteric: System organ site (Sclerae)
- Sclerae are anicteric: Dx name (Sclerae are anicteric)

Sign / Negation, Sign / Negation, Sign / Negation, Sign

CARDIOVASCULAR : **Regular rate and rhythm** . No **murmurs** , **rubs** , or **gallops** .
- CARDIOVASCULAR: System organ site (CARDIOVASCULAR)
- Regular rate and rhythm: Dx name (Regular rate and rhythm)
- murmurs: Dx name (murmurs)
- rubs: Dx name (rubs)
- gallops: Dx name (gallops)

Sign

CHEST : **Lungs CTA** **bilaterally** .
- CHEST: System organ site (CHEST)
- Lungs CTA: System organ site (Lungs)
- Lungs CTA: Dx name (Lungs CTA)
- bilaterally: Direction (bilaterally)

Filter entities Category : All 1 2 3 4 5 6

Entity Type Category Traits

10 05:12 / 07:26

aws

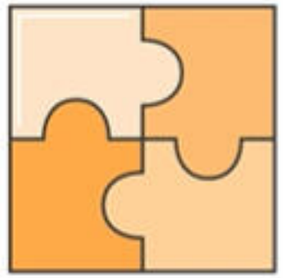
We launch

Features of Amazon Translate

aws training and certification

Secure Communication

Authentication and Access Control

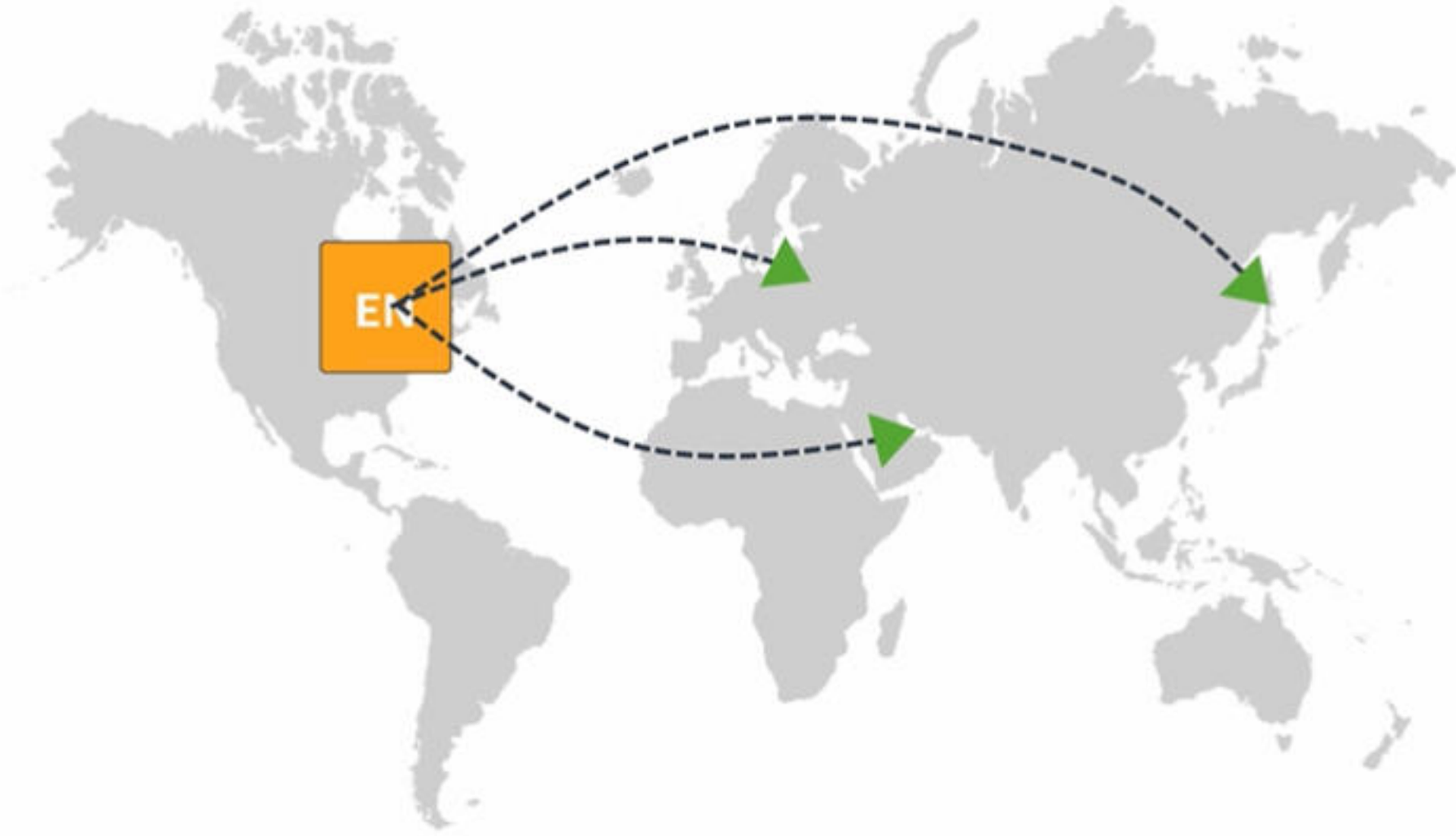


Integration with AWS Services

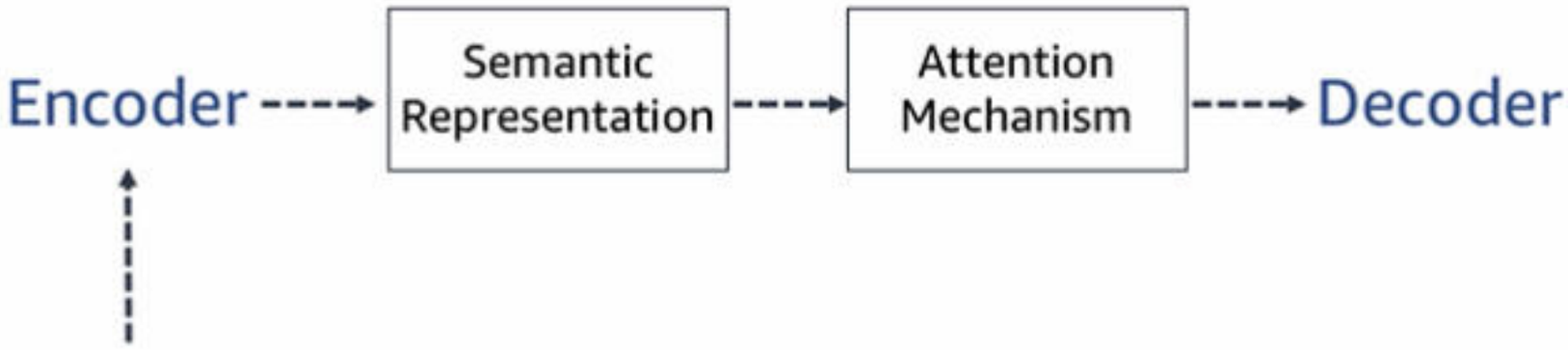
01:19 / 09:04

Applications for Any Language

aws training and certification



How It Works

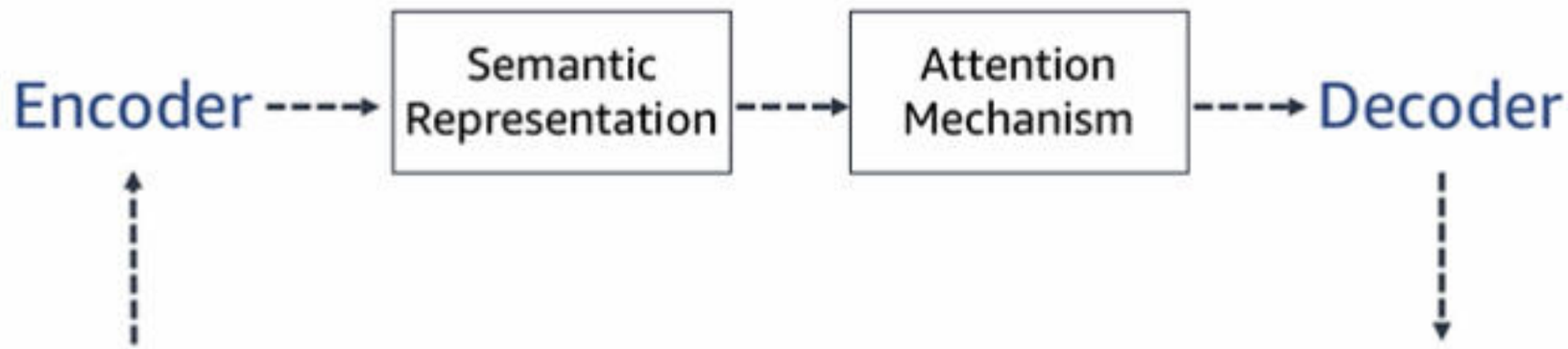


"Good morning, everyone."

Source: English

Target: Spanish

How It Works



"Good morning, everyone."

"Buenos días a todos."

Source: English

Target: Spanish

Getting Started

aws training and certification

1

Set up
AWS account

2

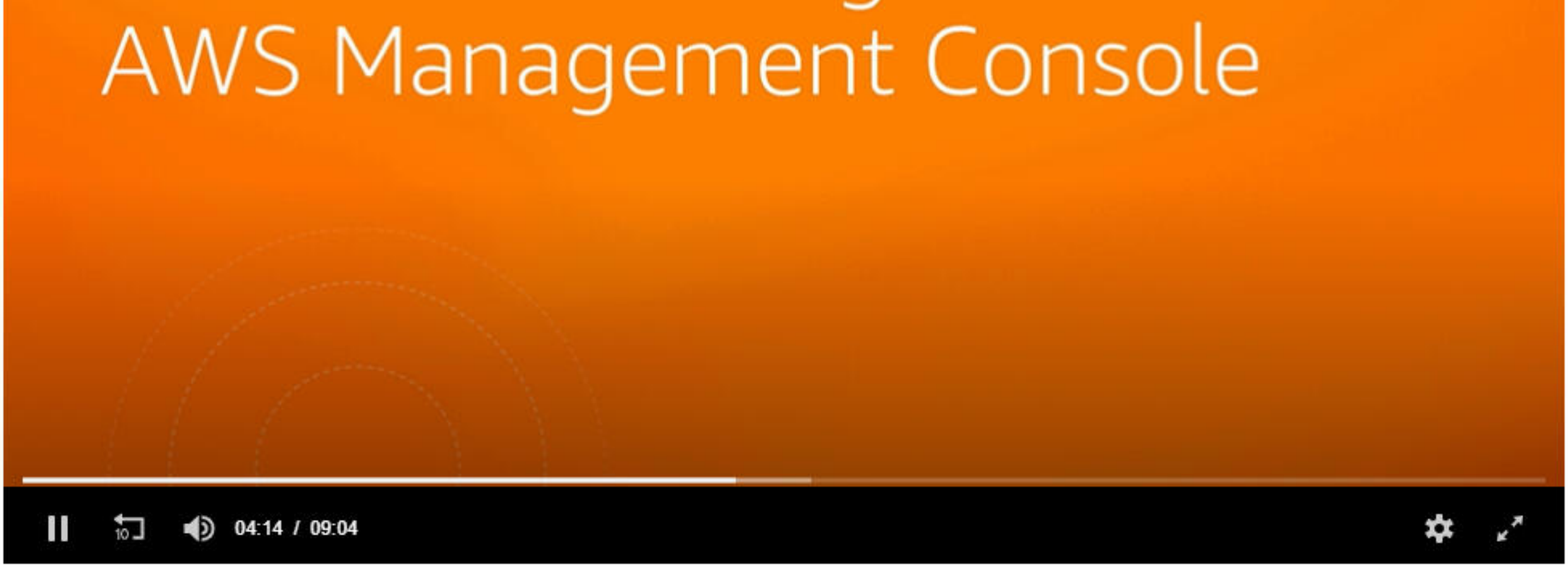
Create
IAM user

3

Connect: Console,
AWS CLI, AWS SDKs

aws training and certification

Demo: Connecting from the AWS Management Console



Connecting from the AWS CLI

aws training and certification

Short text: Input directly into command line

```
aws translate translate-text \
       --endpoint-url endpoint \
       --region region \
       --source-language-code "en" \
       --target-language-code "es" \
       --text "Hello, World"
```

Long Text: Use JSON file

```
aws translate translate-text \
       --endpoint-url endpoint \
       --region region \
       --cli-input-json file://translate.json > translated.json
```

Using AWS SDKs

aws training and certification

```
import boto3

translate = boto3.client(service_name='translate', region_name='region',
            endpoint_url='endpoint', use_ssl=True)

result = translate.translate_text(Text="Hello, World",
    SourceLanguageCode="en", TargetLanguageCode="de")
print('TranslatedText: ' + result.get('TranslatedText'))
print('SourceLanguageCode: ' + result.get('SourceLanguageCode'))
print('TargetLanguageCode: ' + result.get('TargetLanguageCode'))
```

Use Cases

Amazon Translate

Translating web-authored content in real time and on demand

Batch translating pre-existing content for analysis and insights

Back To Products

1966 SHELBY COBRA 427 S/C

Classic Cars

This model of the 1966 Shelby Cobra 427 S/C includes many authentic details and operating parts. The new car was designed in cooperation with Ford in Detroit. A new chassis was built using 4 in (102 mm) main chassis tubes (up from 3 in (76 mm)) and coil spring suspension all around. The new car also had wide fenders and a larger radiator opening. It was powered by the "side oiler" Ford 427 engine (7.0 L) rated at 425 bhp (317 kW), which provided a top speed of 164 mph (262 km/h) in the standard model and 485 bhp (362 kW) with a top speed of 185 mph (298 km/h) in the competition model.

CURRENT PRICE: $74000

ADD TO CART

REVIEWS

30/11/2017 17:30:03 **Carlos Escapa** wrote:
A mi tambien me gusta. Tendria que ir a la peluqueria todos los dias.
Translate

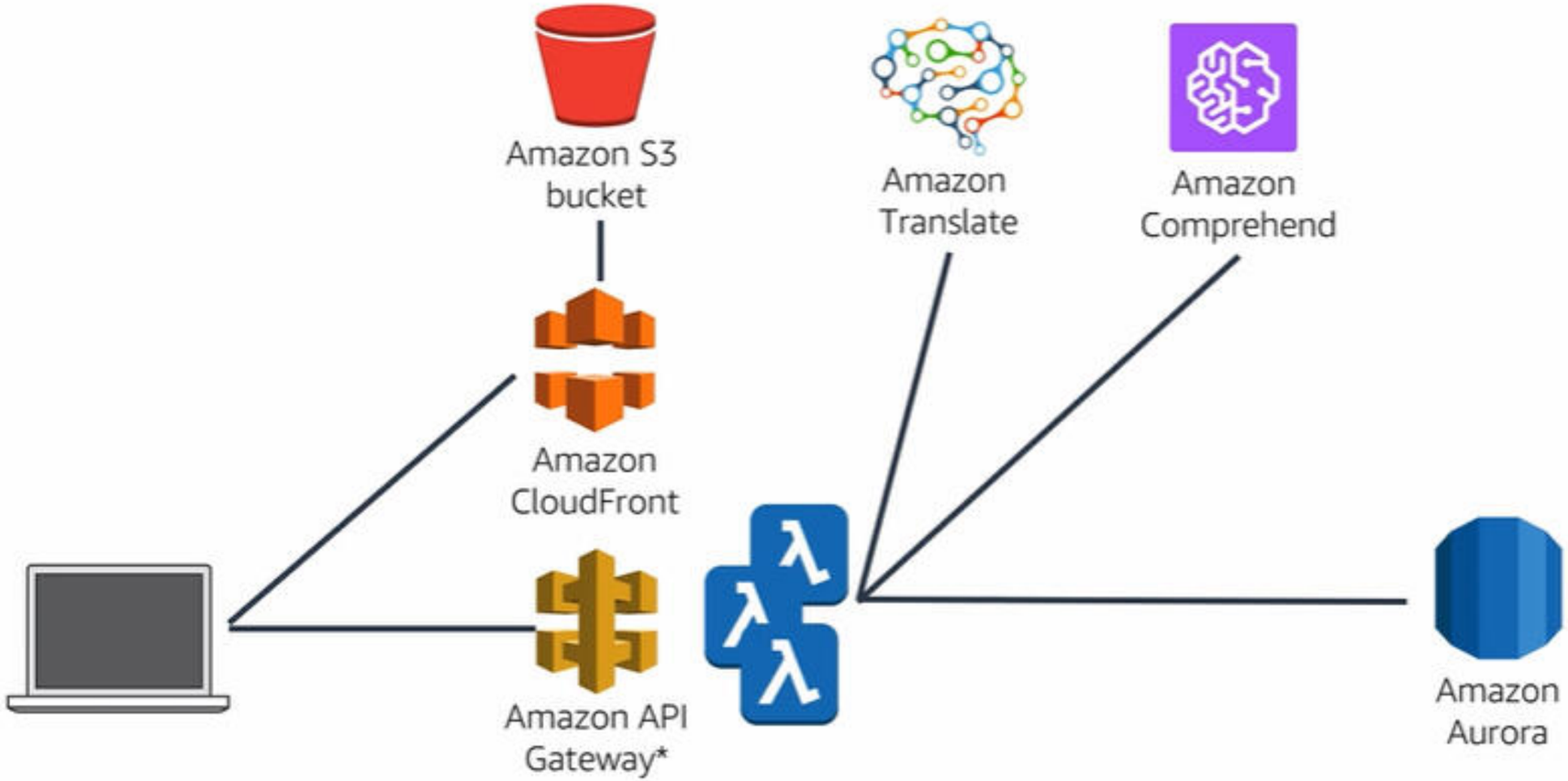
01/03/2018 06:55:32 **Diego Natali** wrote:
I love this car a lot. I believe that James Bond had one. I wish I had one but I am not as rich as Bill Gates!!!
Translate

23/03/2018 12:00:19 **Giuseppe Porcelli** wrote:
When I bought it, I didn't like its size. So I hate it!
Translate

23/03/2018 12:01:16 **Kris Skrinak** wrote:
Slower than my Ducati, faster than my one wheel. Love it!.
Translate

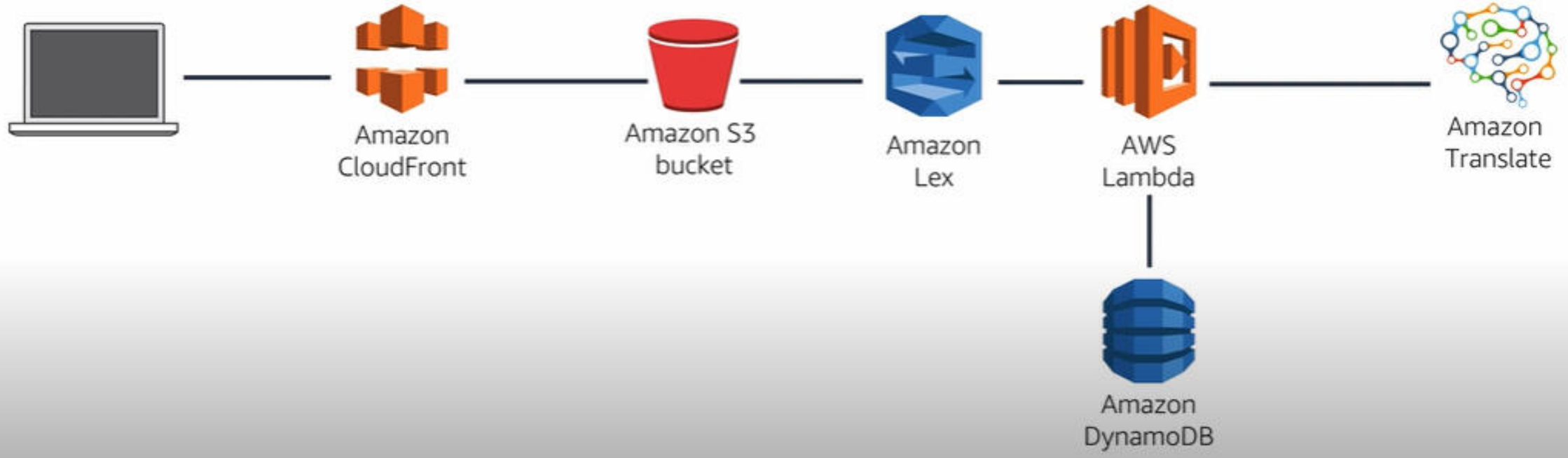
Please login to add or update your review

Example: On-Demand Translation



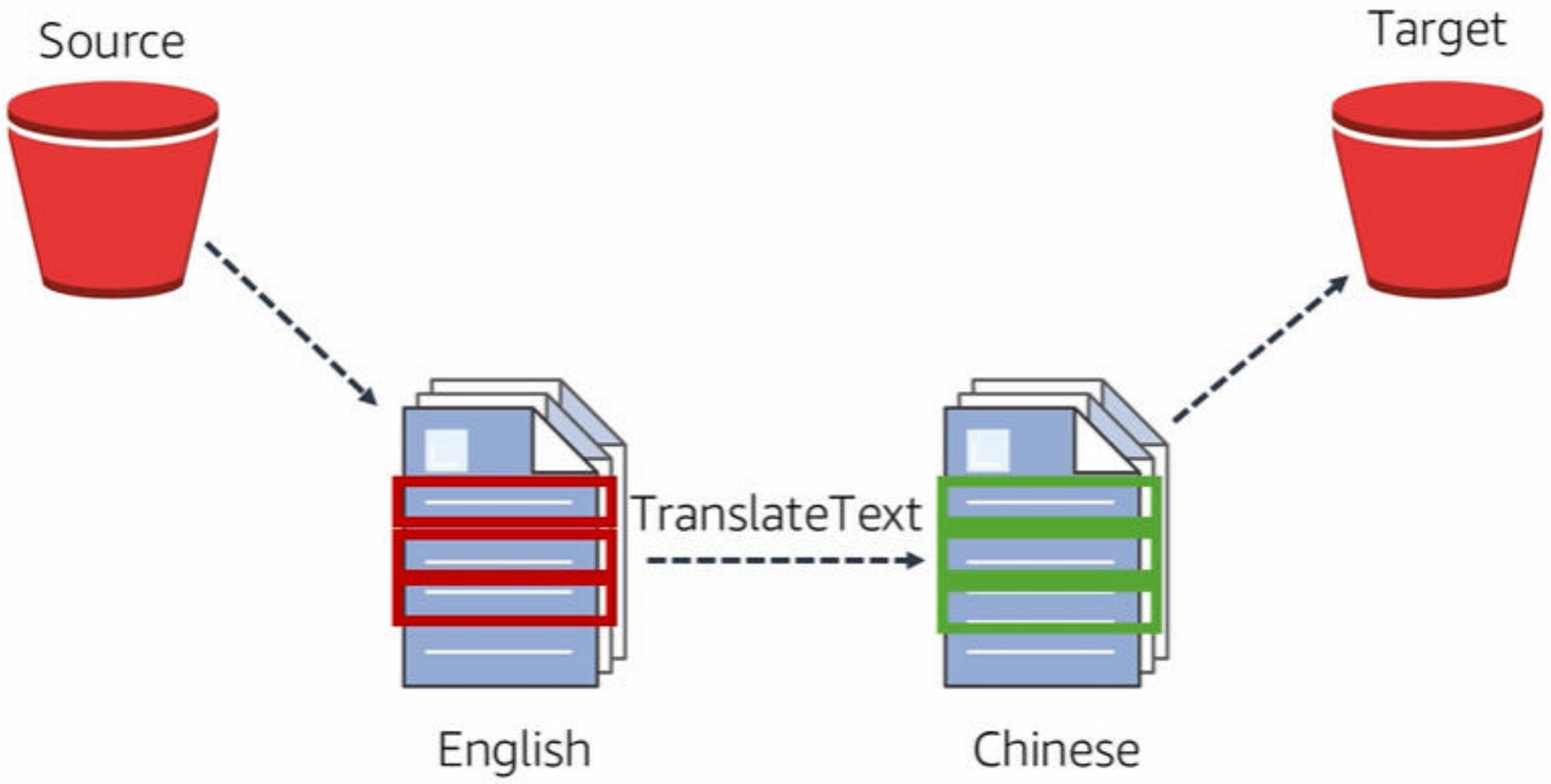
Example: Chatbot Translation

aws training and certification



10 07:18 / 09:04

Example: Batch Translations



Summary

aws training and certification

- Neural machine translation engine
- Real-time, on-demand, and batch translations
- Integrates with AWS services
- Connect via the AWS Management Console, AWS CLI, and AWS SDKs

Thanks for watching!

aws training and certification

Why Amazon Transcribe?

aws training and certification

Readable, helpful transcriptions

```
Four score and seven years ago our fathers brought
forth on this continent a new nation conceived in
liberty and dedicated to the proposition that all
men are created equal. Now we are engaged in a
great civil war testing whether that nation or any
nation so conceived and so dedicated can long
endure.
```

[{"start_time":"0.150","end_time":"0.580","alternatives":
[{"confidence":"0.9984","content":"Four"}],

Why Amazon Transcribe?

aws training and certification

Source-specific accuracy

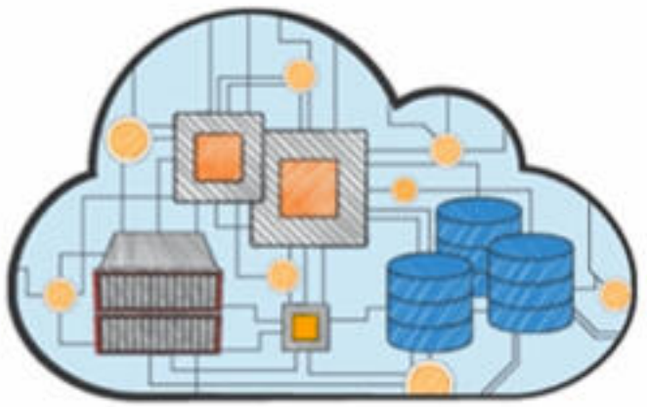
- Custom vocabulary
- Language selection
- Multiple speaker recognition

Why Amazon Transcribe?

aws training and certification

API Interface

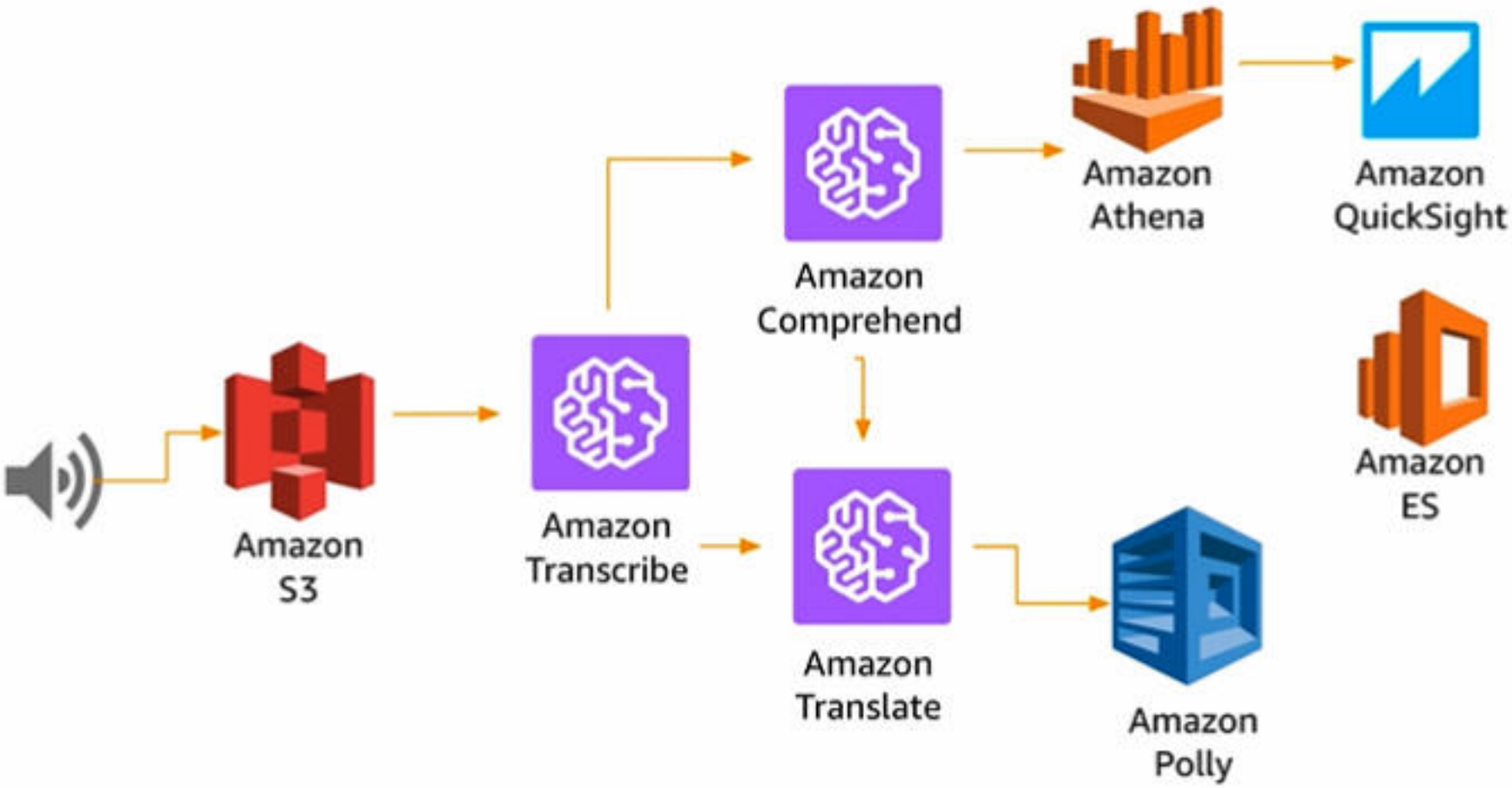
Authentication and Access Control



Integration with AWS Services

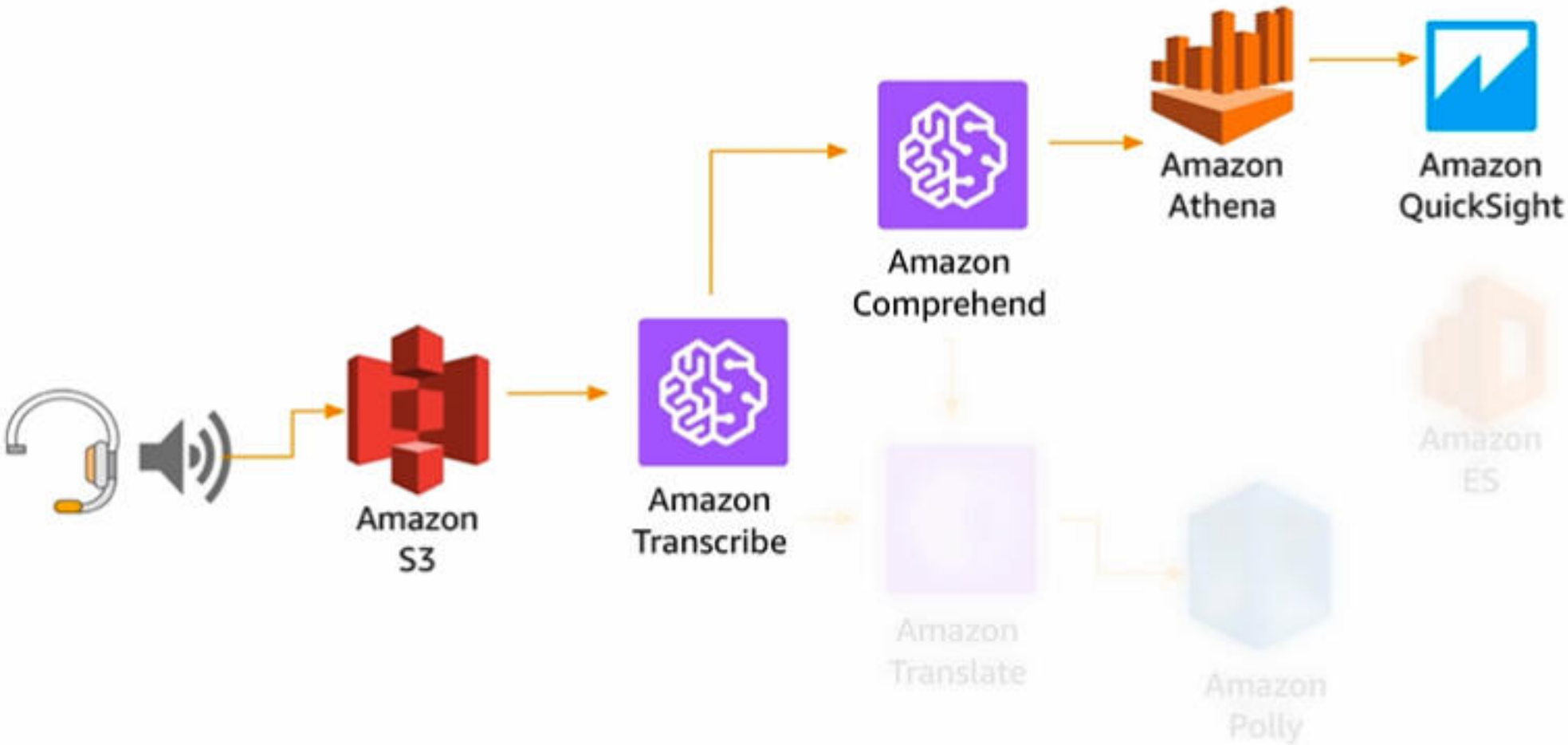
Integrating with AWS Services

aws training and certification



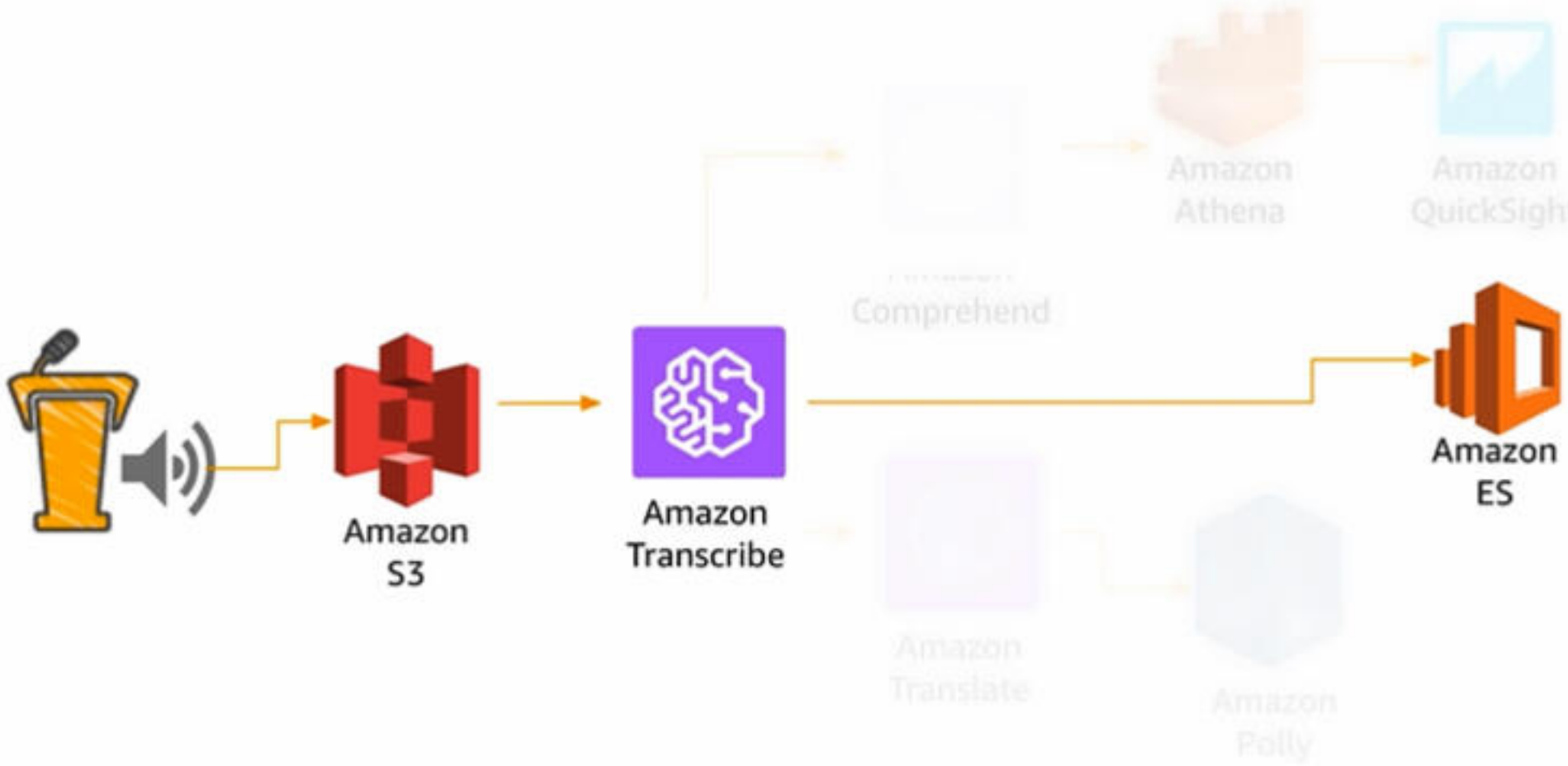
Use Case Call Center Analysis

aws training and certification



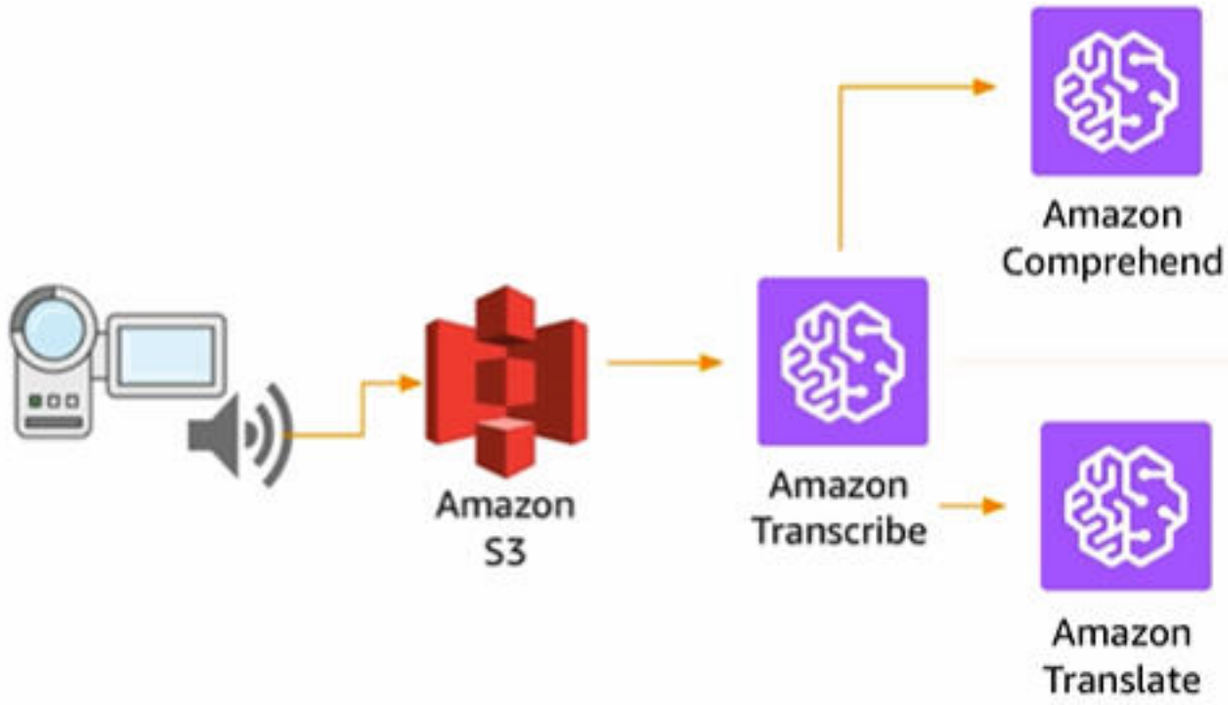
Use Case: Meeting Transcription

aws training and certification



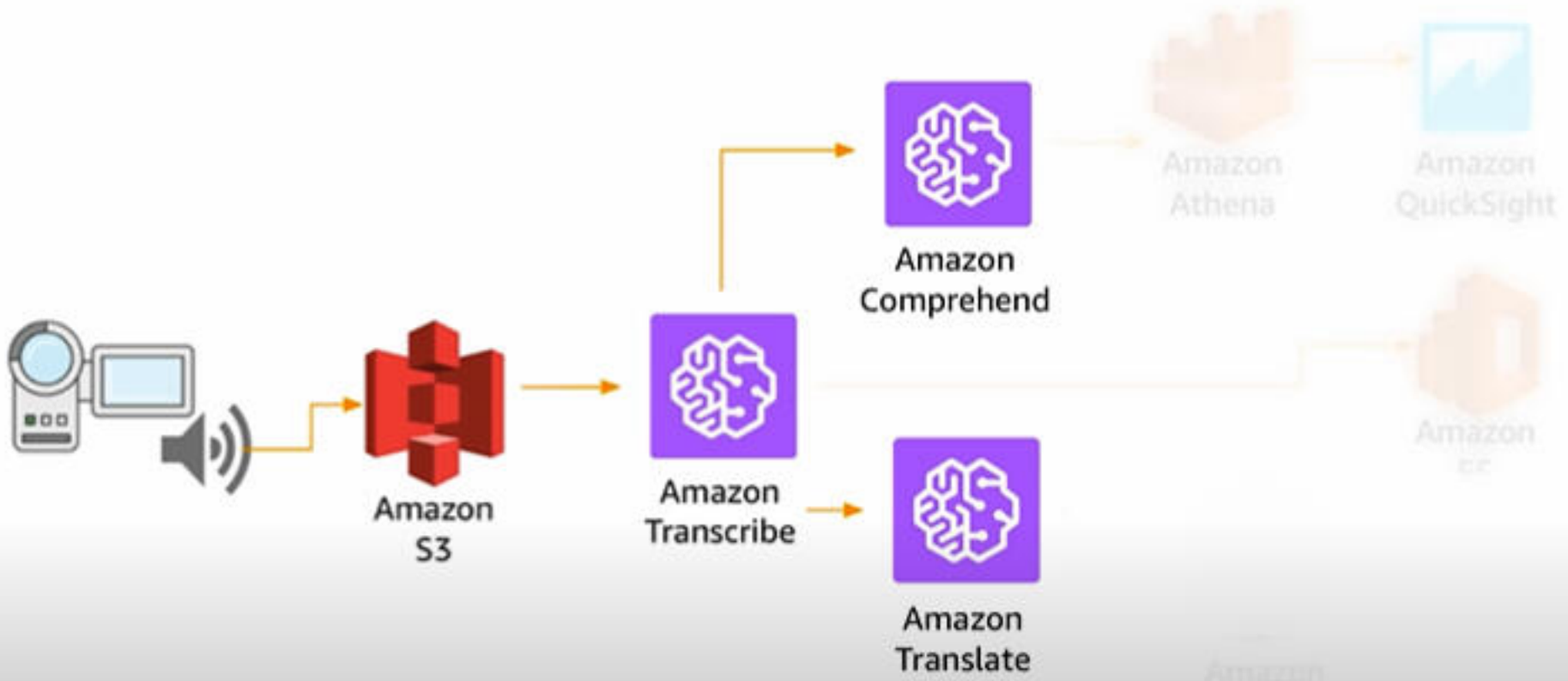
Use Case: Closed Captioned Video

aws training and certification



Use Case: Closed Captioned Video

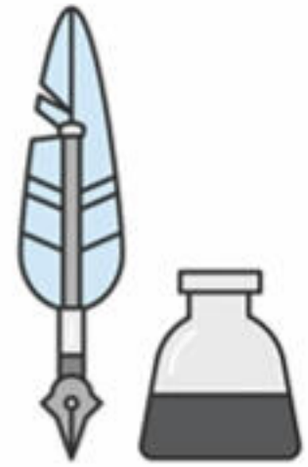
aws training and certification



Getting Started

aws training and certification

To try Amazon Transcribe, you'll need:



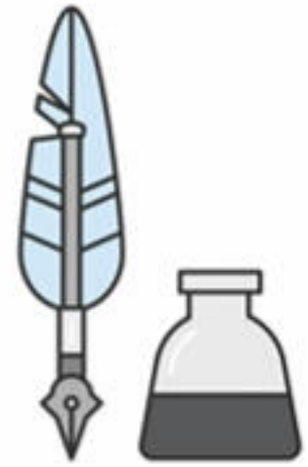
1. WAV, FLAC, MP3 or MP4 audio file(s) on Amazon S3
2. AWS account with an IAM user that has full access to the Transcribe API calls
3. Familiarity with the Command Line Interface (CLI) and a Text Editor

Visit https://aws.amazon.com/documentation/transcribe/ for the Developer Guide.

Getting Started

aws training and certification

To try Amazon Transcribe, you'll need:



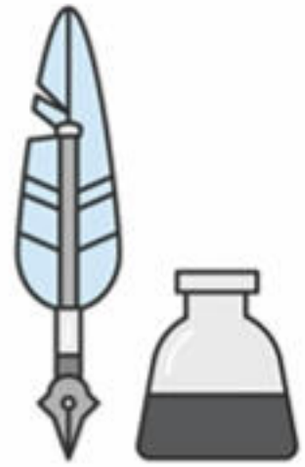
1. WAV, FLAC, MP3 or MP4 audio file(s) on Amazon S3
2. AWS account with an IAM user that has full access to the Transcribe API calls
3. Familiarity with the Command Line Interface (CLI) and a Text Editor

Visit https://aws.amazon.com/documentation/transcribe/ for the Developer Guide.

Getting Started

aws training and certification

To try Amazon Transcribe, you'll need:



1. WAV, FLAC, MP3 or MP4 audio file(s) on Amazon S3
2. AWS account with an IAM user that has full access to the Transcribe API calls
3. Familiarity with the Command Line Interface (CLI) and a Text Editor

Visit https://aws.amazon.com/documentation/transcribe/ for the Developer Guide.

Demonstration

aws training and certification

- Create a transcription job
- Verify the job's completion
- Review the results

MACHINE LEARNING

Amazon Transcribe

Automatic Speech Recognition powered by deep learning

Amazon Transcribe provides high-quality and affordable speech-to-text transcription for a wide range of use cases.

Launch the demo

We know you're curious. Jump in and test our Automatic Speech Recognition API.

Try Amazon Transcribe

Features

Amazon S3 integration

Amazon Transcribe is integrated with Amazon Simple Storage Service (S3), enabling you to easily transcribe audio files stored in S3 buckets.

Timestamp generation

Amazon Transcribe returns a timestamp for each word, enabling you to search for a particular word in the transcription and locate it in the original stored recording.

Support for telephony audio

Amazon Transcribe supports telephony-quality audio (8 kHz) with high accuracy.

Easy-to-read transcriptions

Amazon Transcribe adds punctuation and formatting automatically so the output is more reader-friendly and can be used without further editing.

Sample use cases

Voice analytics

Discover customer insights and improve contact center performance.

Media & Entertainment

Automate closed captioning and subtitling workflows for greater accessibility.

Advertising

Search & Compliance

Pricing (US)

Preview

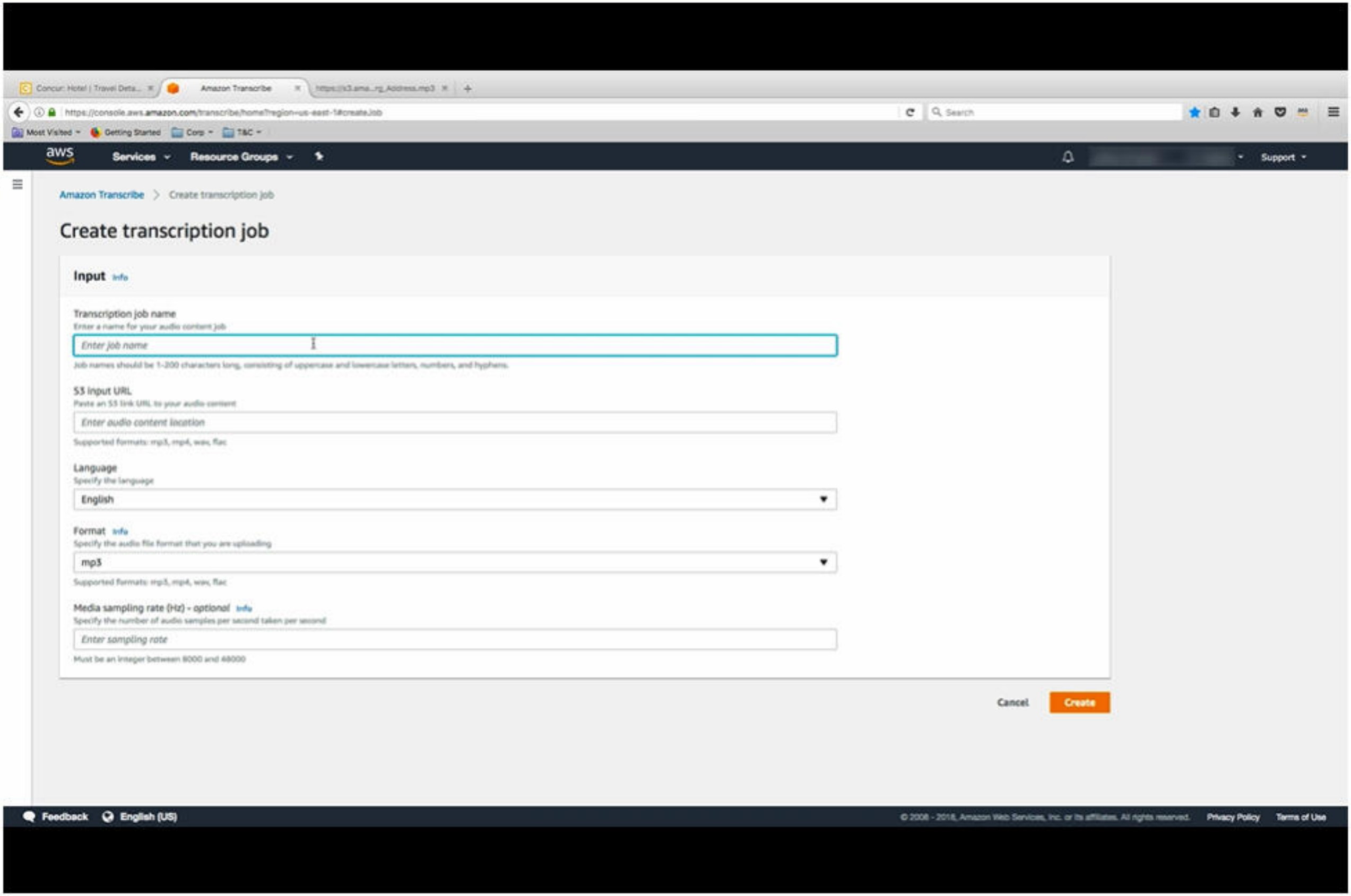
You can test Amazon Transcribe free of charge during Preview. Your feedback is highly appreciated, and helps build the best service for your needs.

Preview	Free

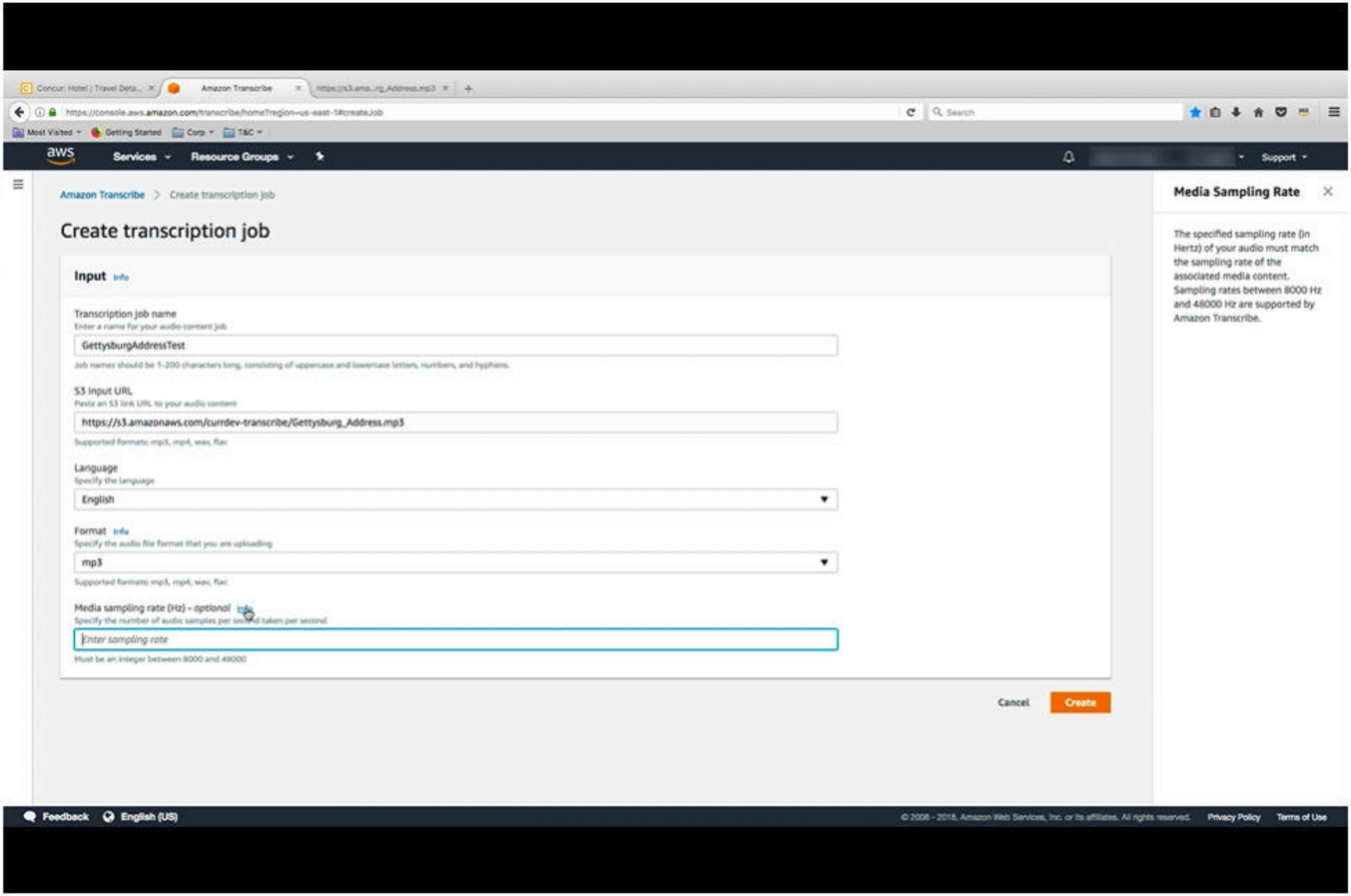
After preview

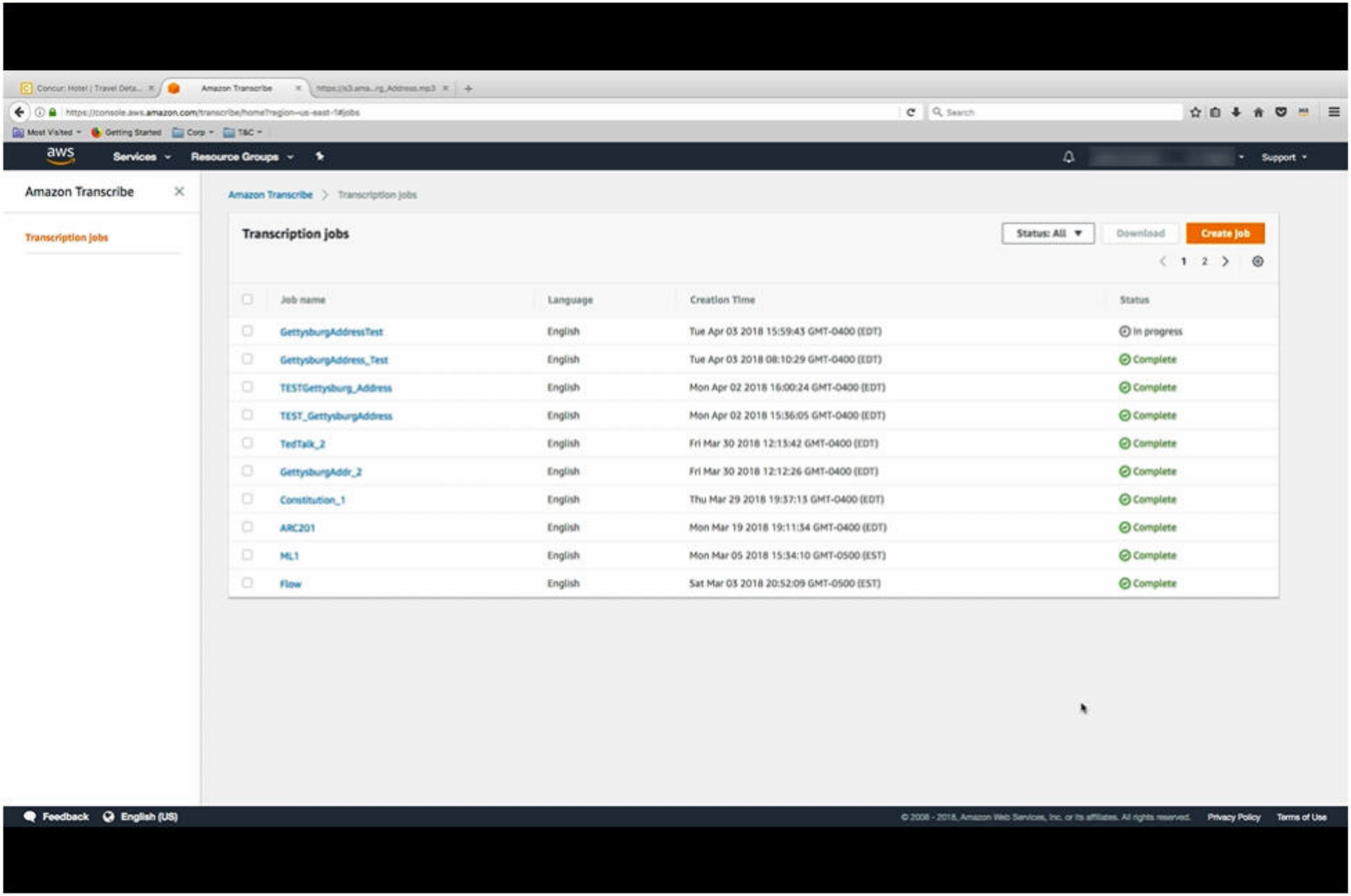
The first 60 minutes of audio in each monthly cycle will be free for the first year upon signup. Additional usage will be billed at $0.0004/second on a pay-as-you-go basis. Every request is measured in seconds with a minimum of 15 seconds per request. For additional pricing details, see the pricing page on our website.

1 second of audio	$0.0004

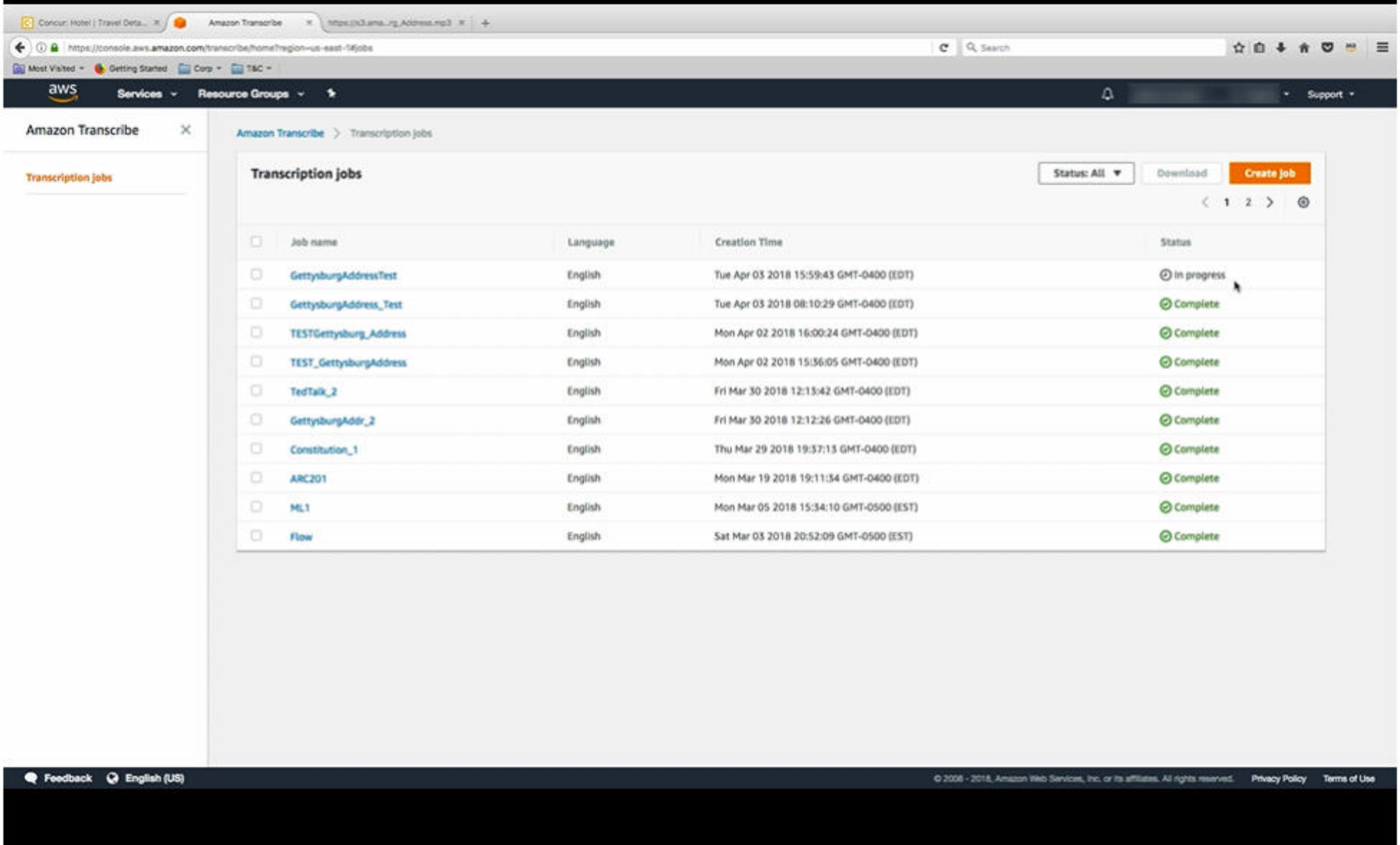


content.aws.training/video/sitscb/en/v1/1.0.0/jwplayer.html?endpoint=https%3a%2f%2flrs.aws.training%2fTCAPI%2f&auth=Basic%20OjczZTlhOWY0L...





content.aws.training/video/sitscb/en/v1/1.0.0/jwplayer.html?endpoint=https%3a%2f%2flrs.aws.training%2fTCAPI%2f&auth=Basic%20OjczZTlhOWY0L...



content.aws.training/video/sitscb/en/v1/1.0.0/jwplayer.html?endpoint=https%3a%2f%2flrs.aws.training%2fTCAPI%2f&auth=Basic%20OjczZTlhOW...

Warning | Amazon Transcribe | https://s3.ama...rg_Address.mp3

https://console.aws.amazon.com/transcribe/home?region=us-east-1#job-details/GettysburgAddressTest

Most Visited | Getting Started | Corp | T&C

aws | Services | Resource Groups | Support

Amazon Transcribe > Transcription jobs > GettysburgAddressTest

GettysburgAddressTest

Details

Download transcription

Job name
GettysburgAddressTest

Status
Complete

Expiration date
The transcription will be available for another 89 days

Input URL
https://s3.amazonaws.com/currdev-transcribe/Gettysburg_Address.mp3

Output URL
https://s3.amazonaws.com/ /GettysburgAddressTest/asrOutput.json

Code Samples

View API documentation

Audio conversion

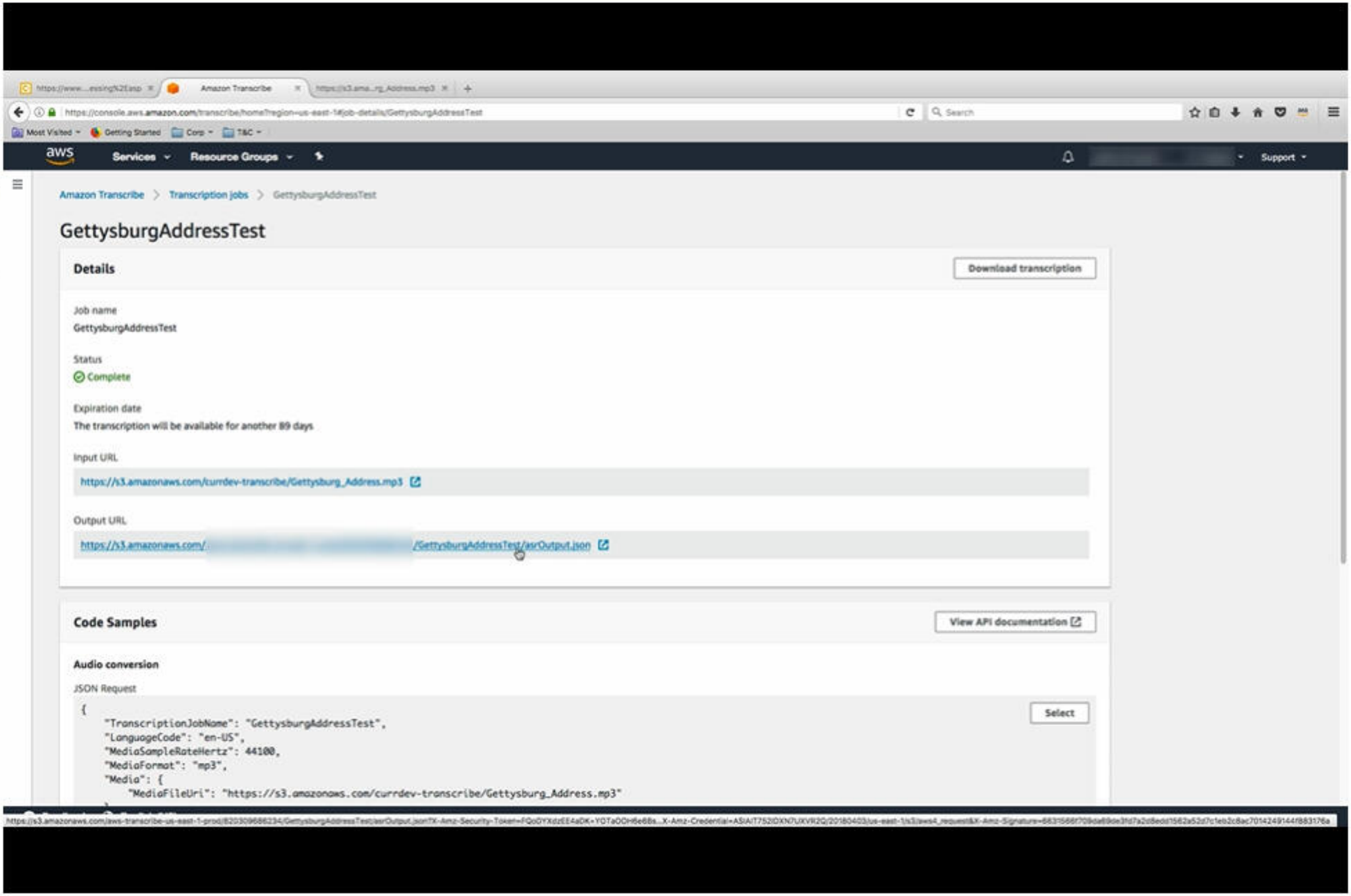
JSON Request

Select

```
{
    "TranscriptionJobName": "GettysburgAddressTest",
    "LanguageCode": "en-US",
    "MediaSampleRateHertz": 44100,
    "MediaFormat": "mp3",
    "Media": {
        "MediaFileUri": "https://s3.amazonaws.com/currdev-transcribe/Gettysburg_Address.mp3"
```

Feedback | English (US) | Privacy Policy | Terms of Use

08:43 / 09:41



content.aws.training/video/sitscb/en/v1/1.0.0/jwplayer.html?endpoint=https%3a%2f%2flrs.aws.training%2fTCAPI%2f&auth=Basic%20OjczZTlhOW...

[redacted]

https://console.aws.amazon.com/transcribe/homeRegion=us-east-1#job-details/GettysburgAddressTest

Most Visited · Getting Started · Corp · T&C

aws Services · Resource Groups · Support

Amazon Transcribe > Transcription jobs > GettysburgAddressTest

GettysburgAddressTest

Details

Download transcription

Job name
GettysburgAddressTest

Status
Complete

Expiration date
The transcription will be available for another 89 days

Input URL
https://s3.amazonaws.com/currdev-transcribe/Gettysburg_Address.mp3

Output URL
https://s3.amazonaws.com/ /GettysburgAddressTest/asrOutput.json

Code Samples

View API documentation

Audio conversion

JSON Request

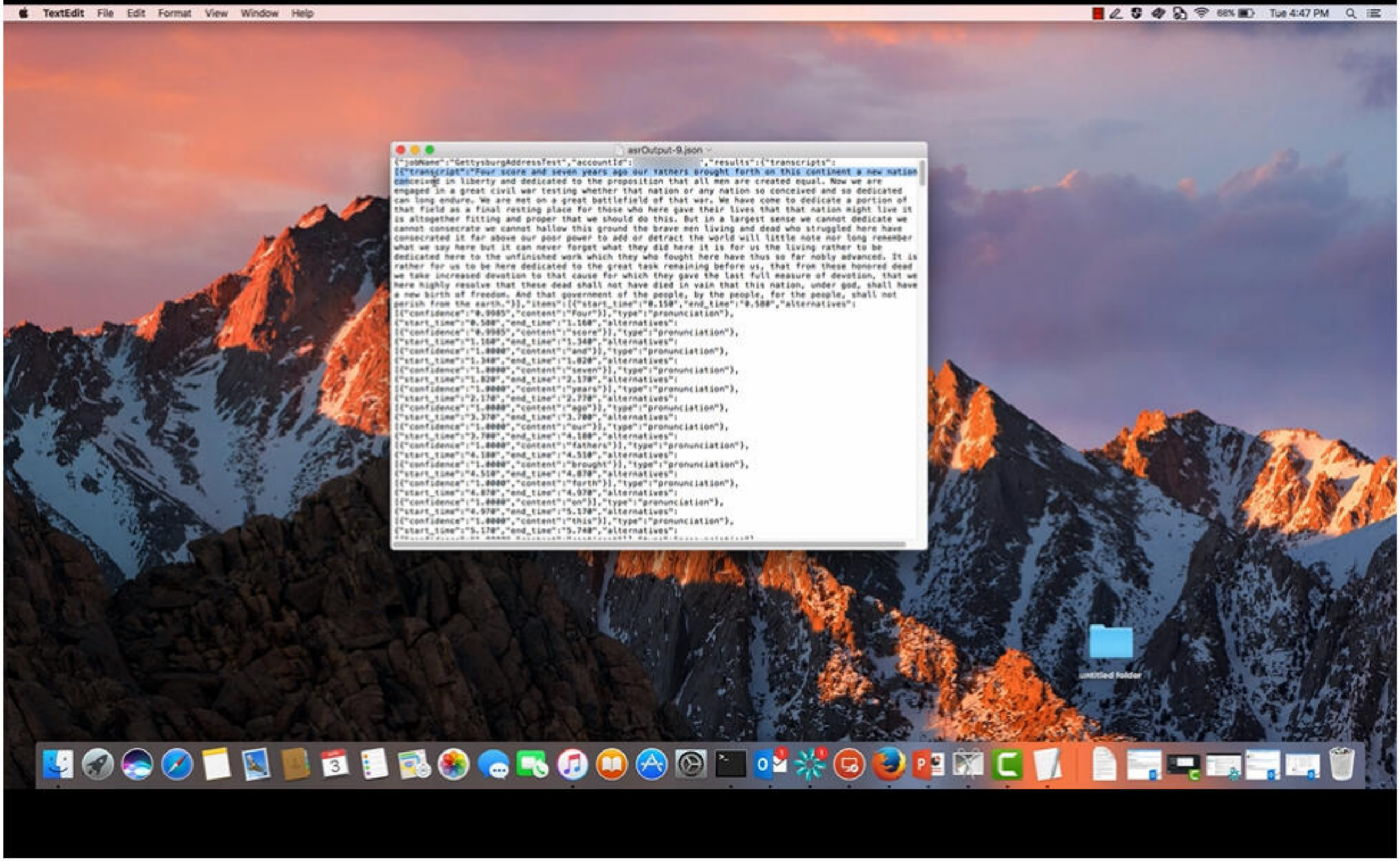
Select

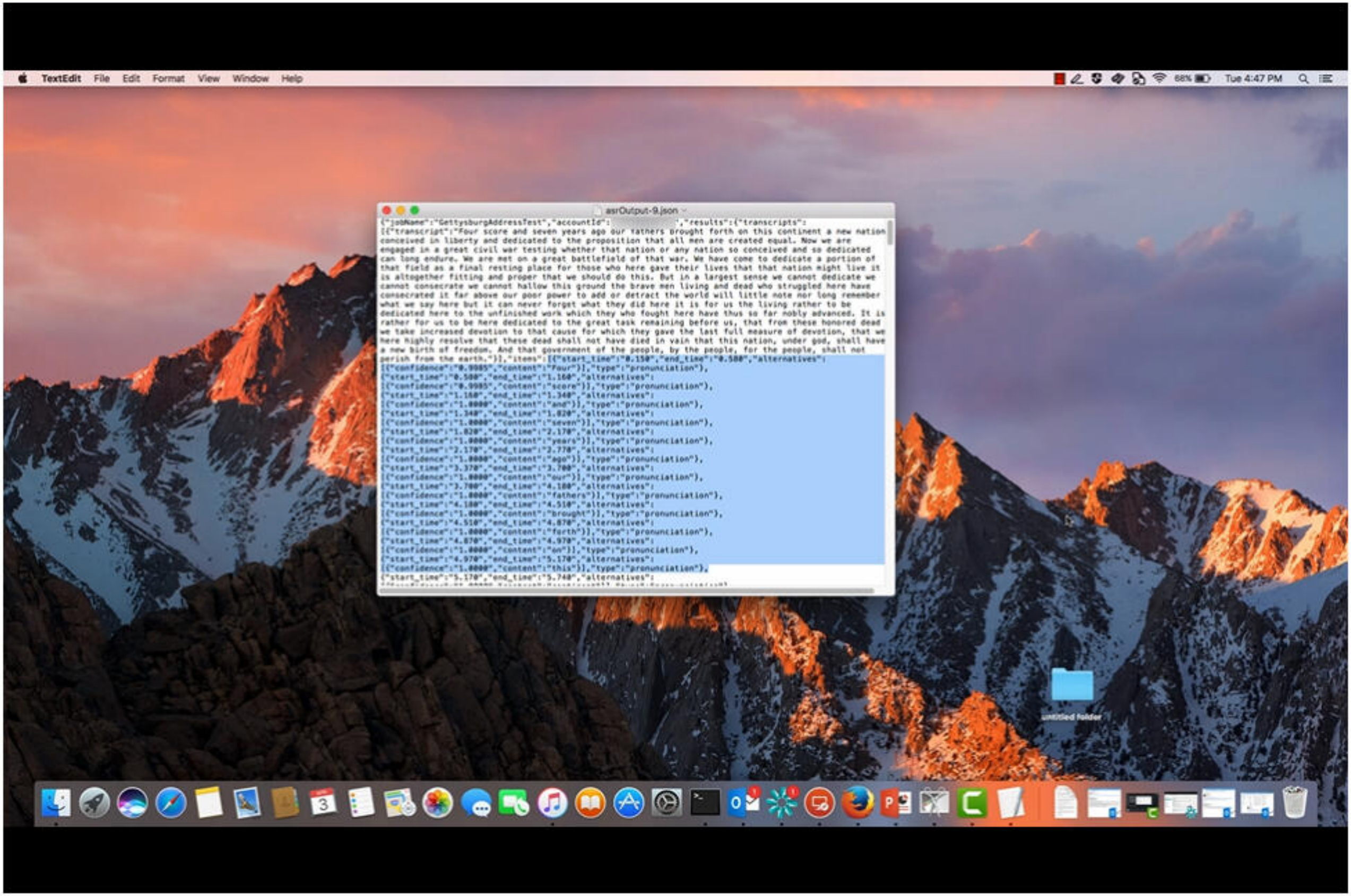
```
{
    "TranscriptionJobName": "GettysburgAddressTest",
    "LanguageCode": "en-US",
    "MediaSampleRateHertz": 44100,
    "MediaFormat": "mp3",
    "Media": {
        "MediaFileUri": "https://s3.amazonaws.com/currdev-transcribe/Gettysburg_Address.mp3"
    }
```

https://s3.amazonaws.com/aws-transcribe-us-east-1-prod/820309686234/GettysburgAddressTest/asrOutput.json?X-Amz-Security-Token=FQoDYXdzEE4aDK+YDTaOCH6e6Bs...X-Amz-Credential=ASIAIT752IDXN7UXVR2Q/20180403/us-east-1/s3/aws4_request&X-Amz-Signature=6831568f709da89de3fd7a2cd8edd1582a52d7c1eb2c8ac7014249144f883176a

[redacted]

content.aws.training/video/sitscb/en/v1/1.0.0/jwplayer.html?endpoint=https%3a%2f%2flrs.aws.training%2fTCAPI%2f&auth=Basic%20OjczZTlhOW...





Thanks for watching!

aws training and certification

Machine Intelligence

aws training and certification

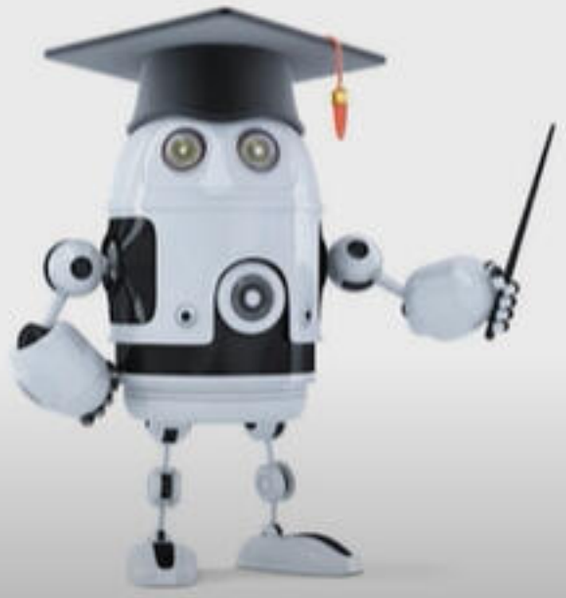
Knowledge Acquisition

Inference

Types of Machine Learning

aws training and certification

Supervised



Unsupervised

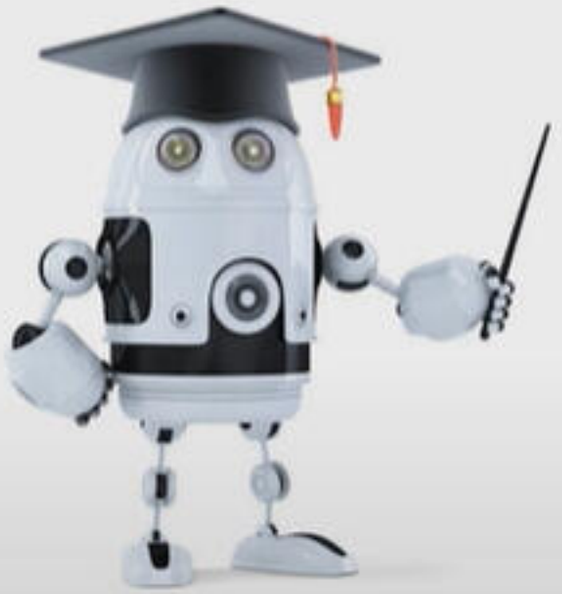
things I can

eat | play with

Types of Machine Learning

aws training and certification

Supervised

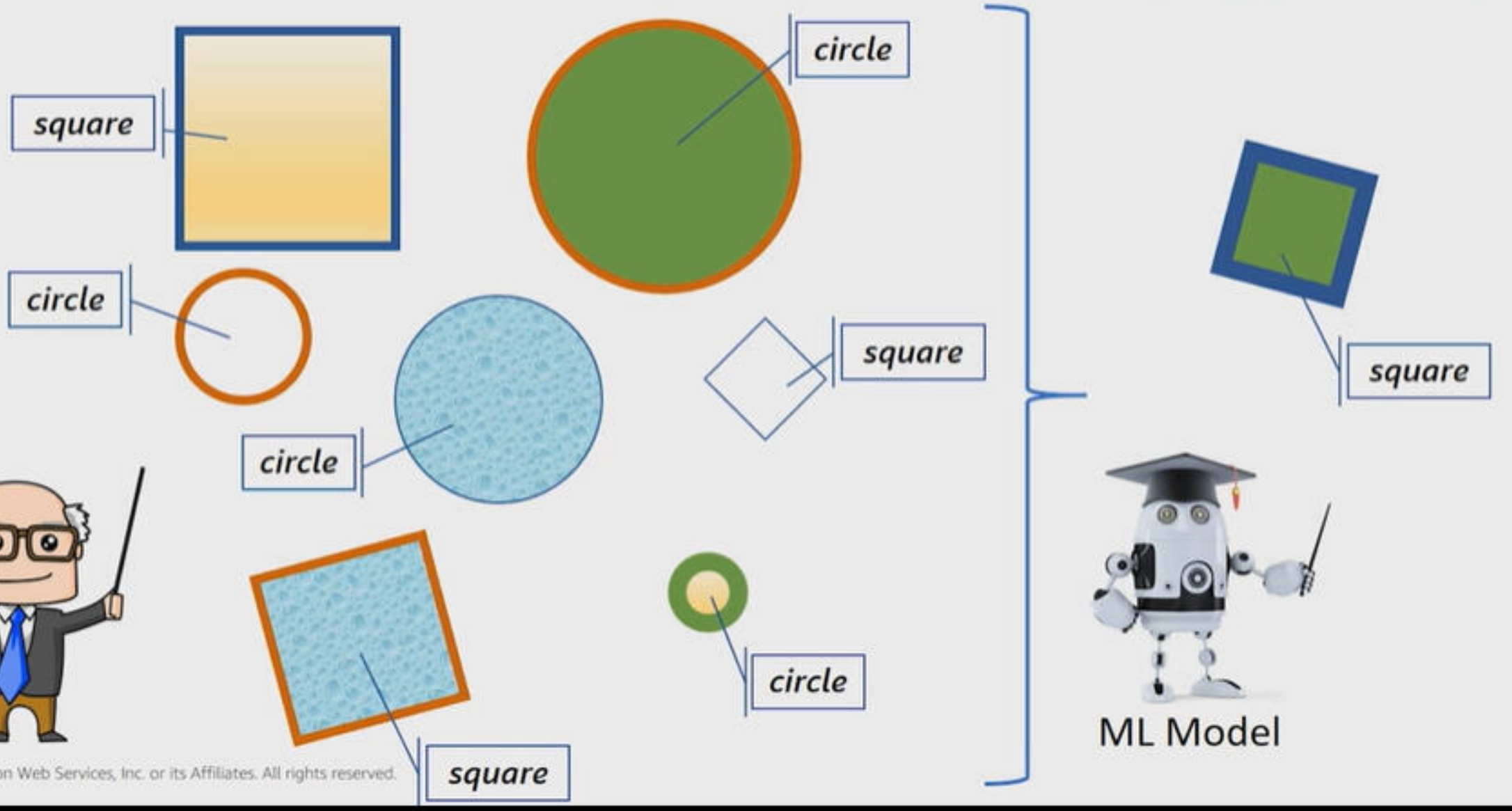


Unsupervised

Reinforcement

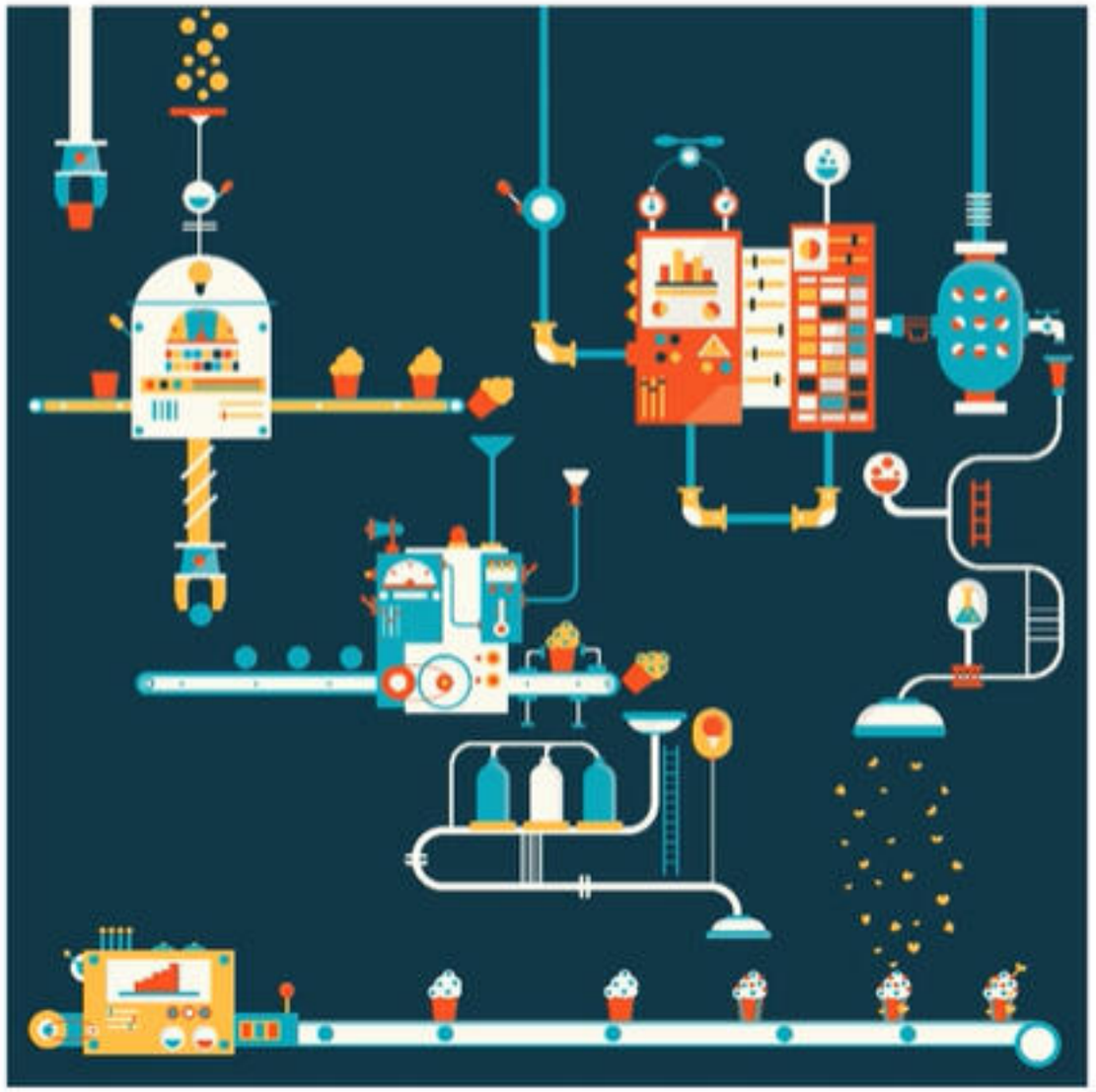
Supervised Learning: Example

aws training and certification



Supervised Learning

aws training and certification



Supervised Learning: Observations

aws training and certification

Input →  → Outcome

Supervised Learning: Training

aws training and certification

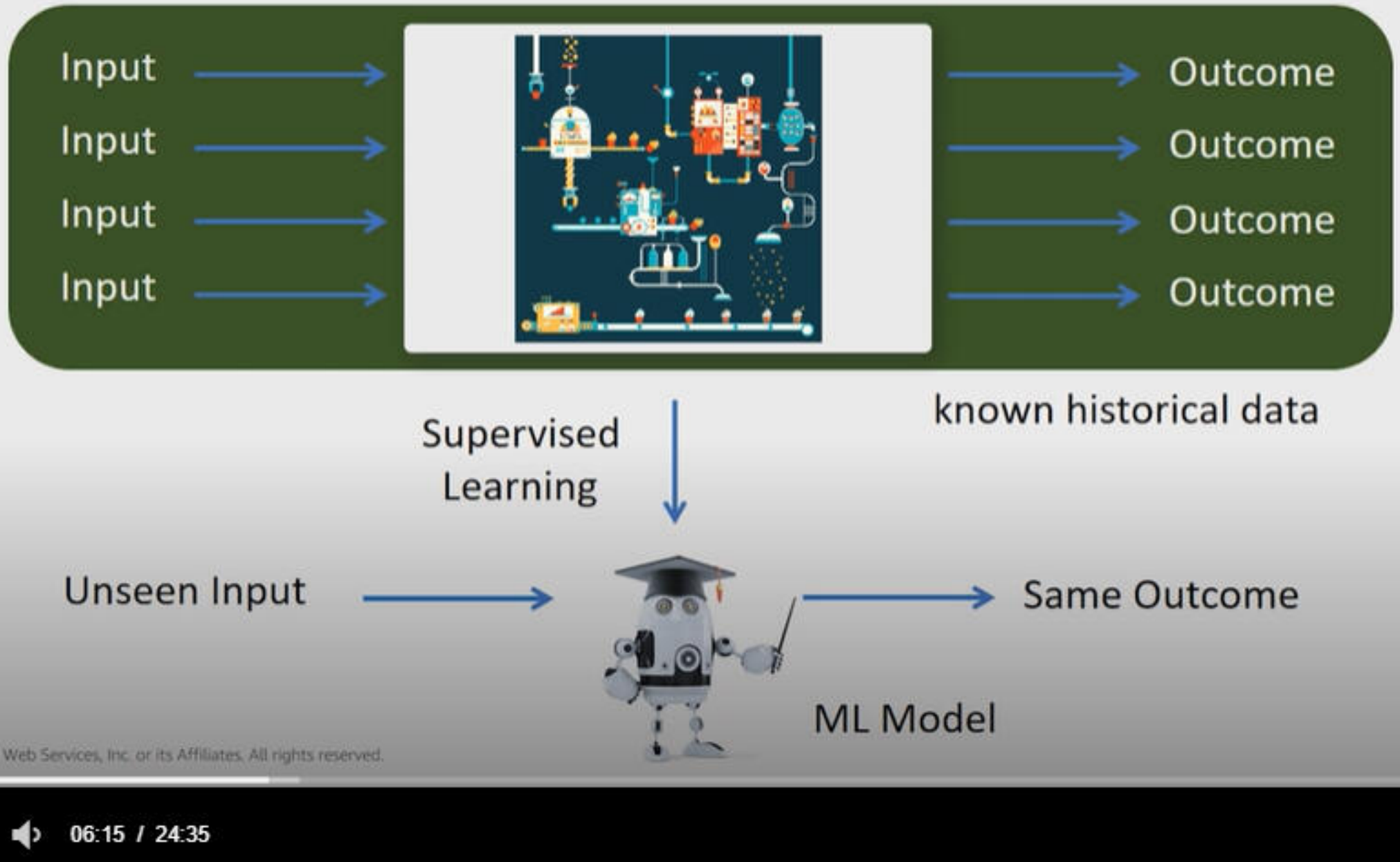
known historical data

Supervised Learning

ML Model

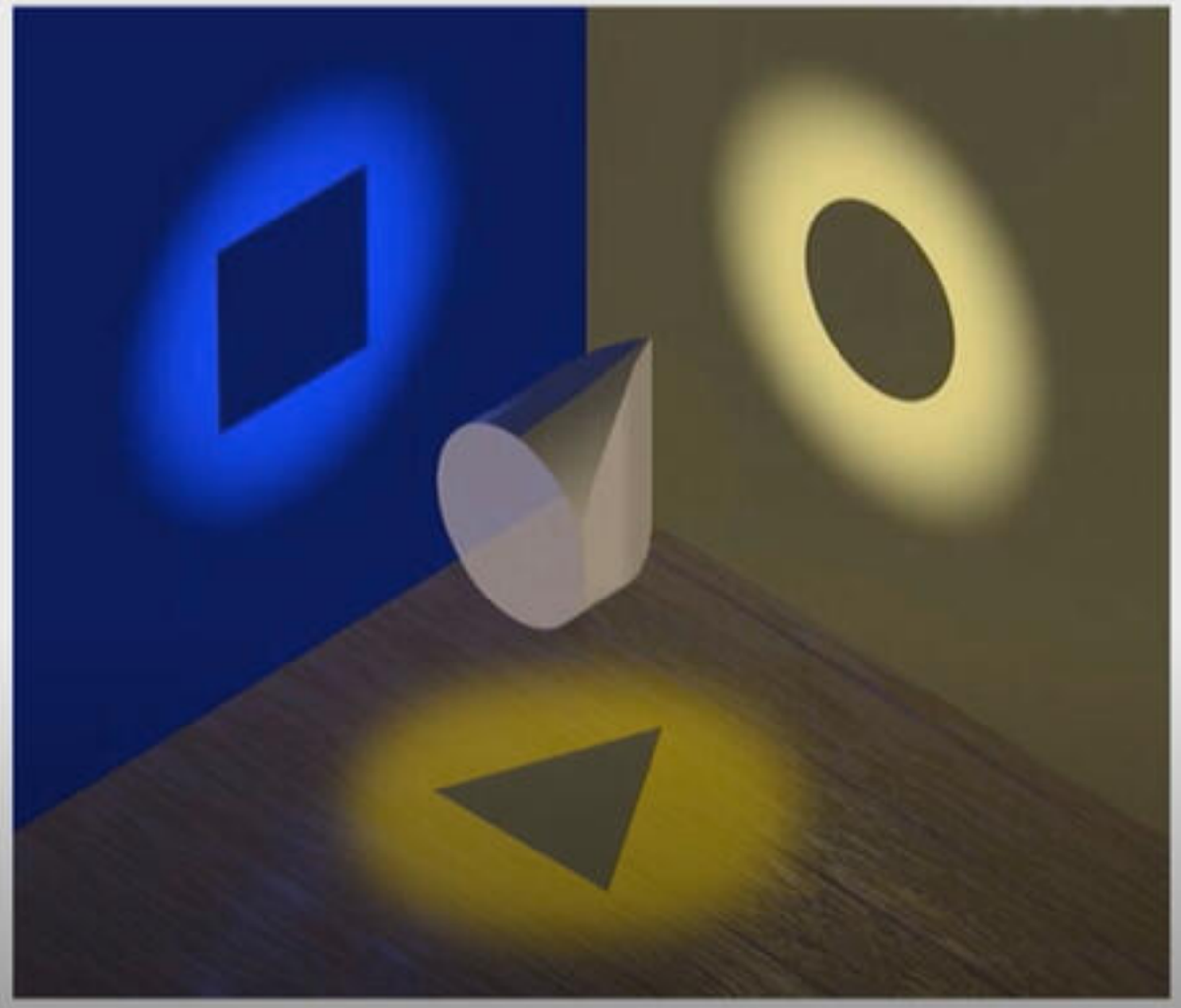
Supervised Learning: Trained Model

aws training and certification



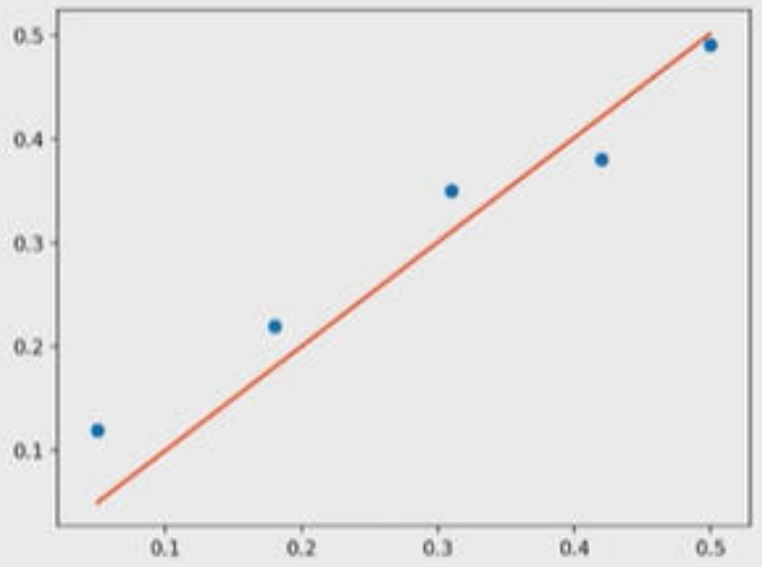
Mutually Exclusive Classes

aws training and certification



Predicting Numbers: Regression

aws training and certification



Supervised Learning Examples

aws training and certification

Supervised Learning Examples

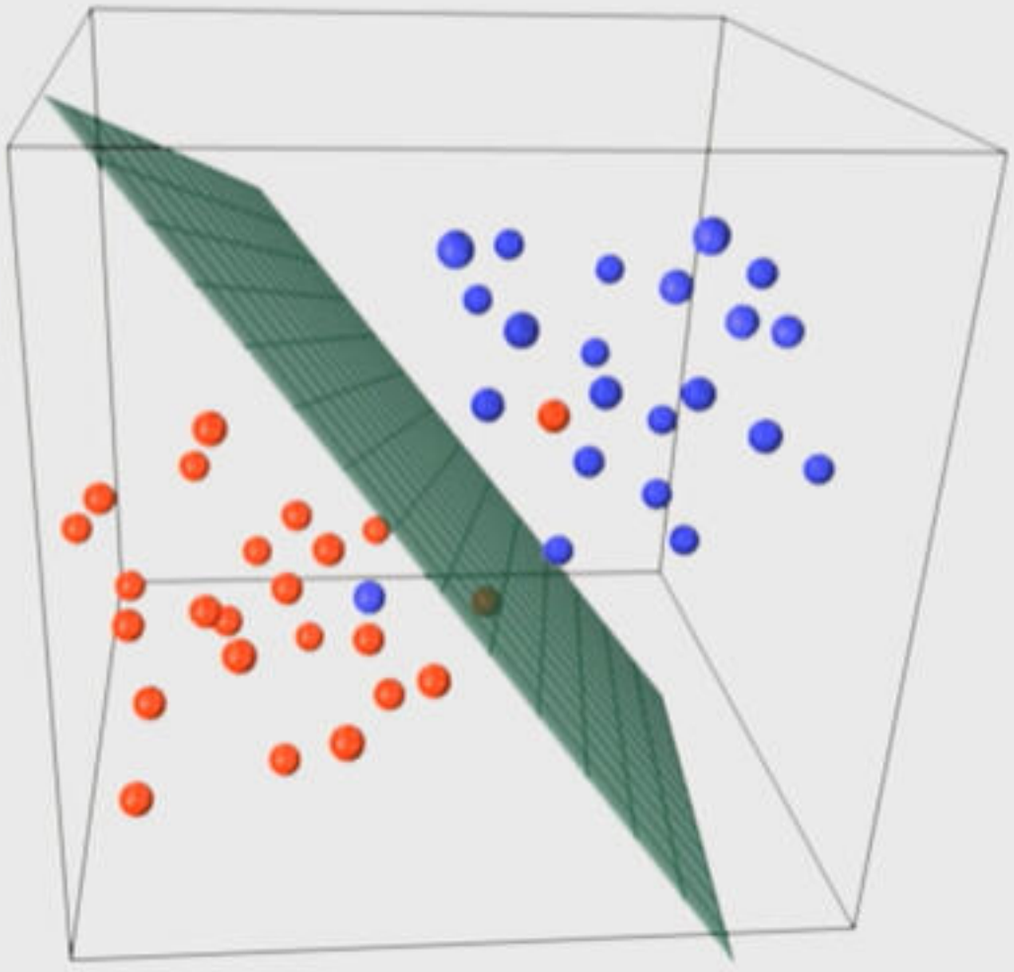
aws training and certification

aws training and certification

Supervised Learning Algorithms

Linear Supervised Algorithms

aws training and certification



Linear Supervised Algorithms

aws training and certification

- In Amazon SageMaker: Linear Learner

  - Linear + Logistic Regression

Linear Supervised Algorithms

aws training and certification

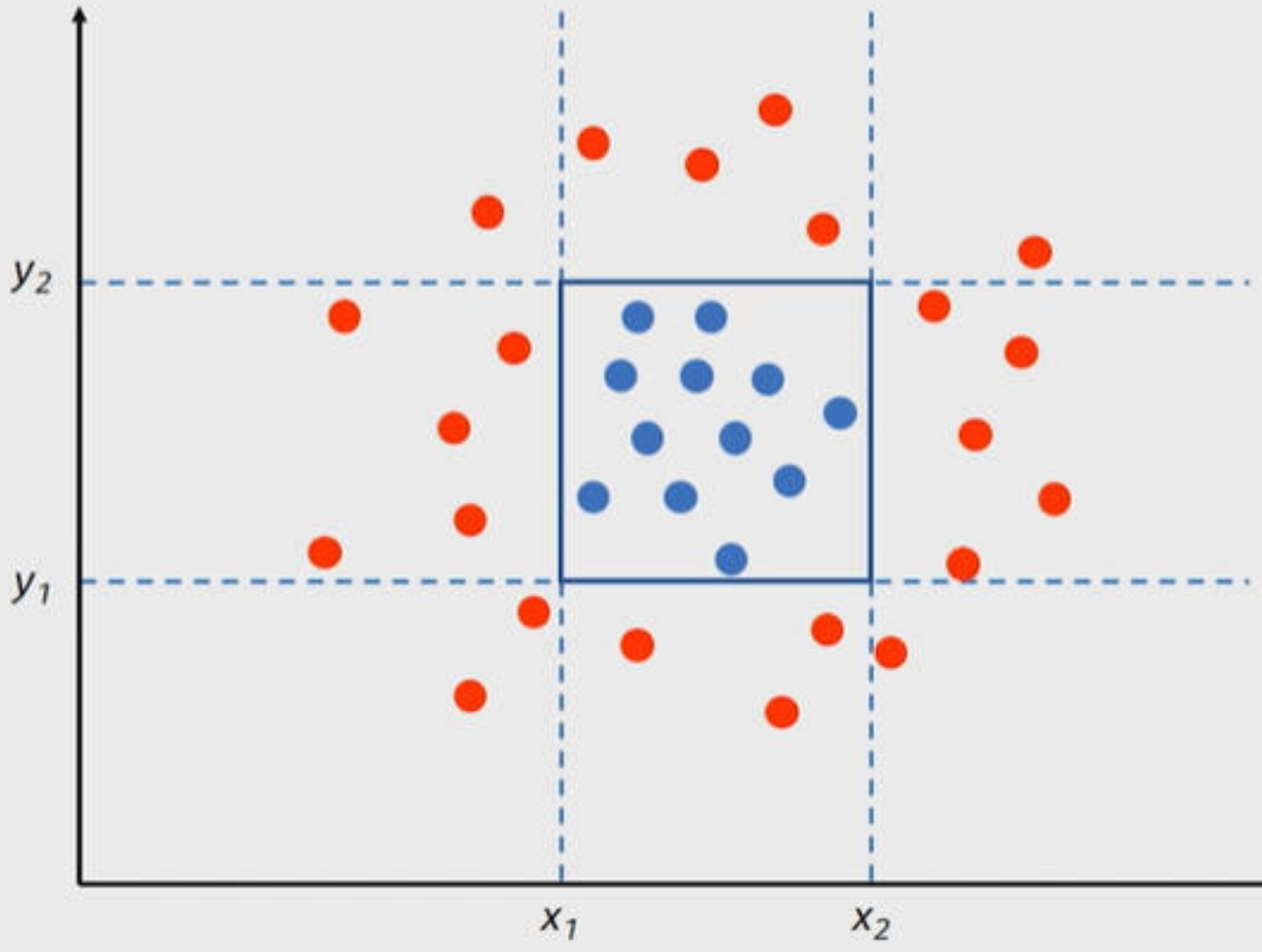
- In Amazon SageMaker: Linear Learner

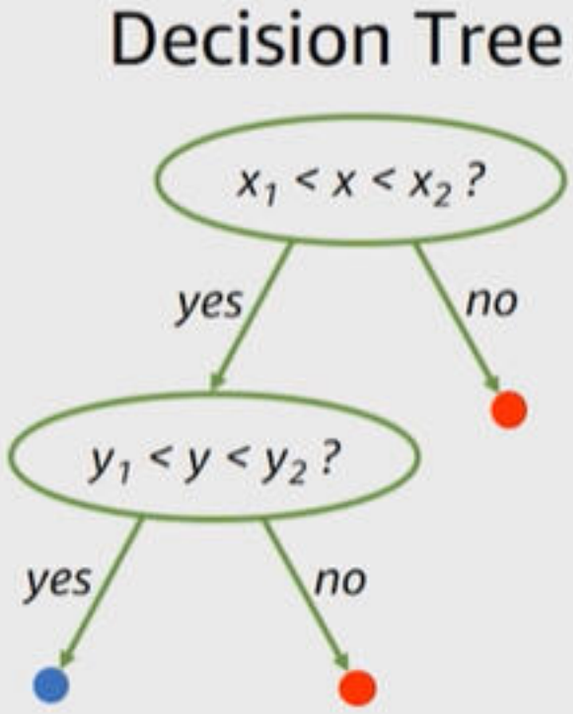
  - Linear + Logistic Regression

- Other common algorithms:
  - Support Vector Machine (SVM)
  - Perceptron

Non-Linear Supervised Algorithms

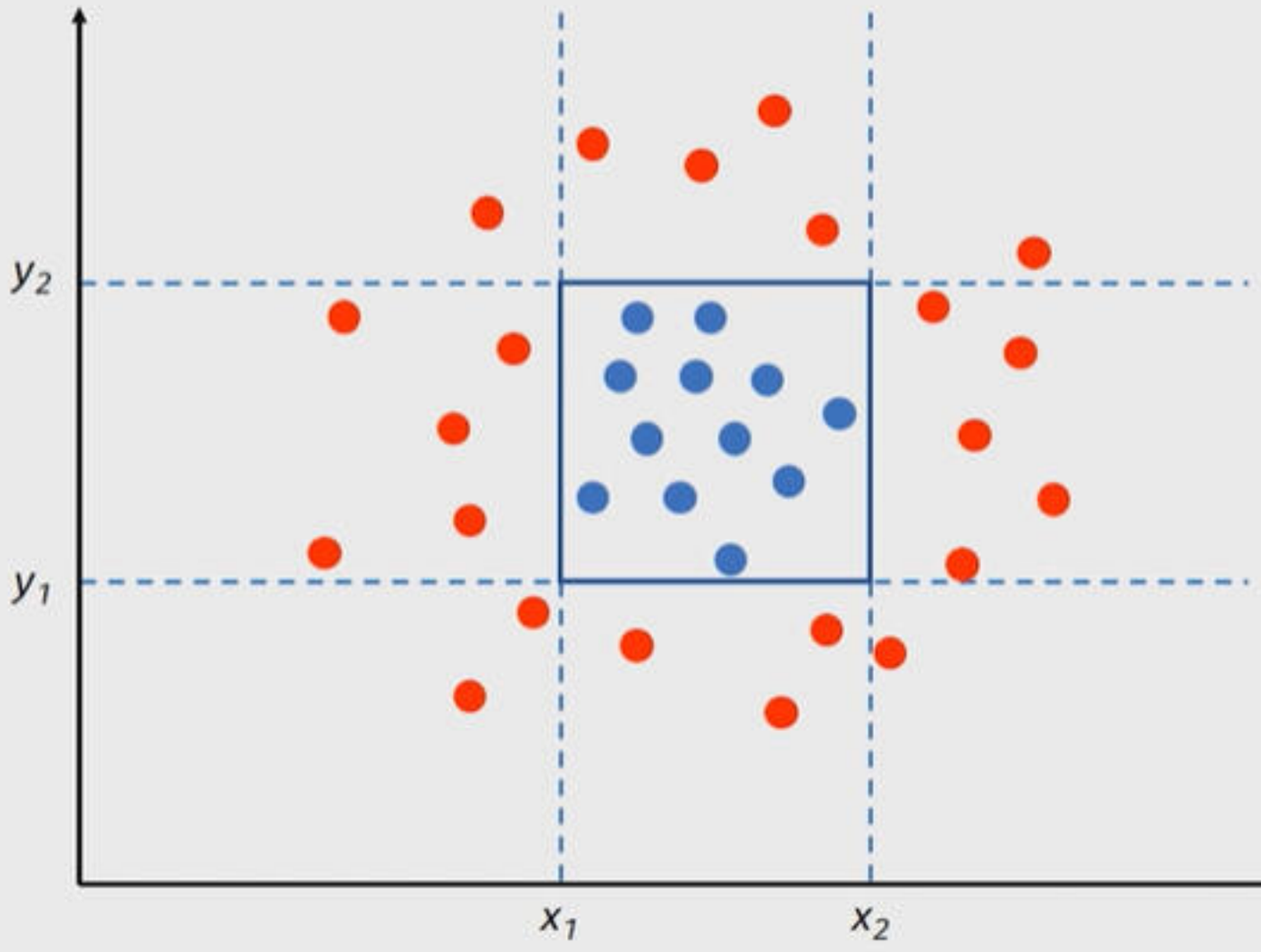
aws training and certification

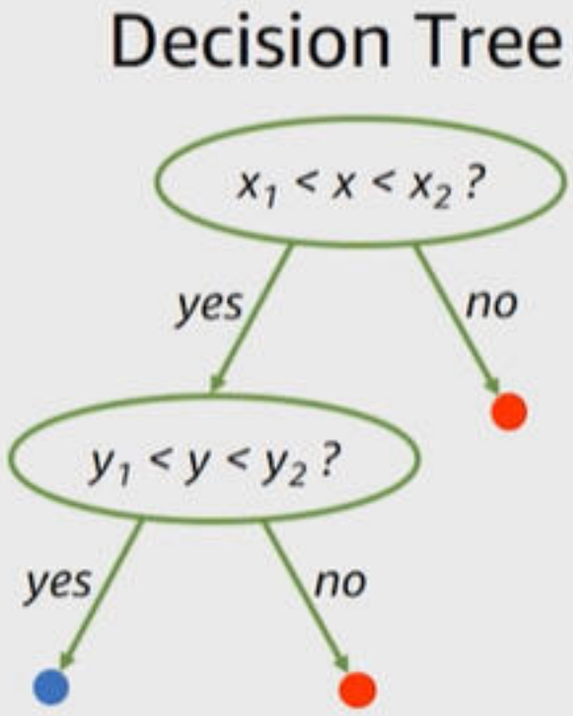




Non-Linear Supervised Algorithms

aws training and certification





Non-Linear Supervised Algorithms

aws training and certification

- XGBoost - Gradient Boosted Trees
- Factorization Machines
  - Good for high dimensional sparse datasets
  - e.g. click prediction & item recommendation

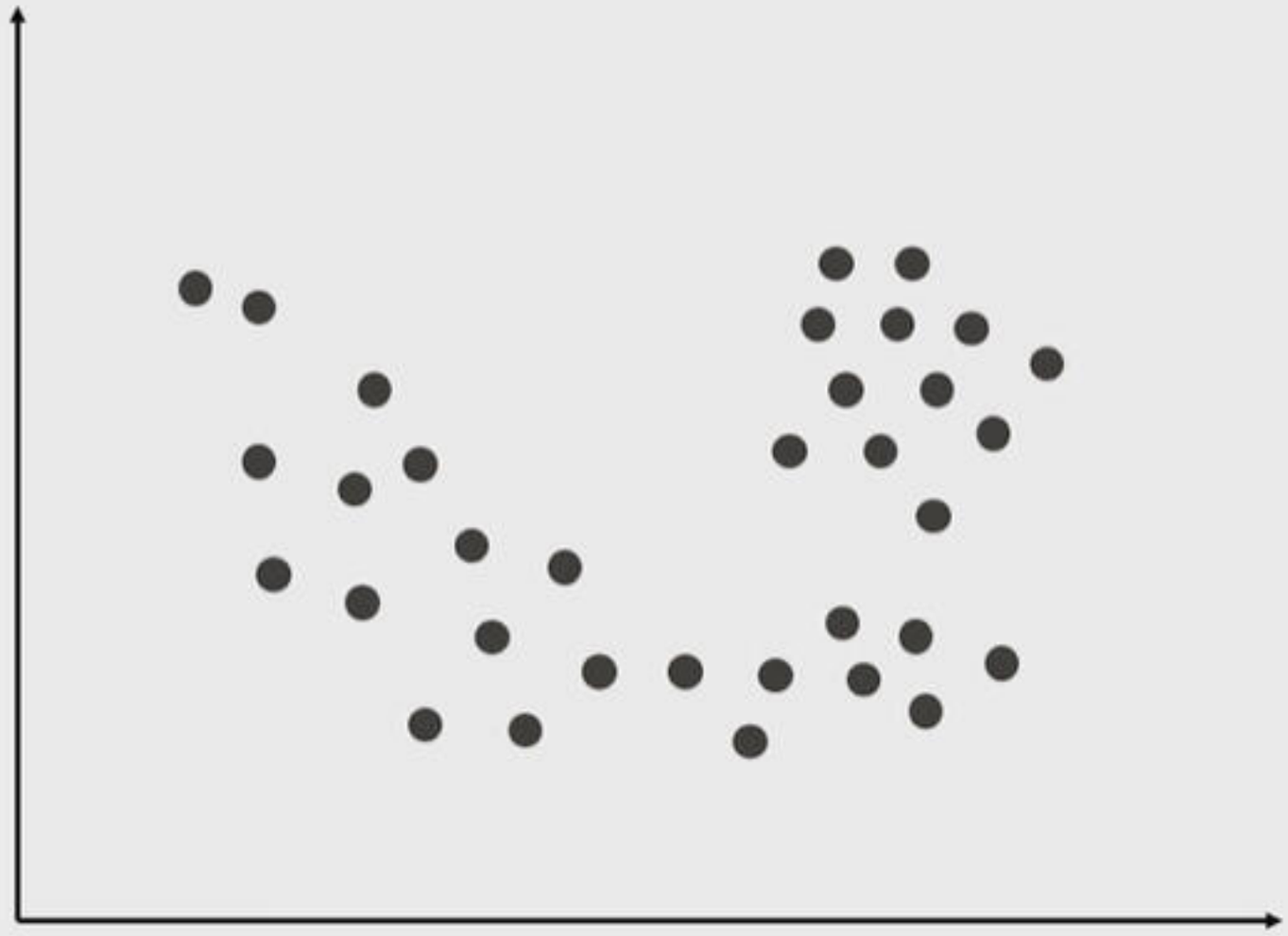
Non-Linear Supervised Algorithms

aws training and certification

- XGBoost - Gradient Boosted Trees
- Factorization Machines
  - Good for high dimensional sparse datasets
  - e.g. click prediction & item recommendation
- Other popular approaches
  - Polynomial
  - Neural Networks

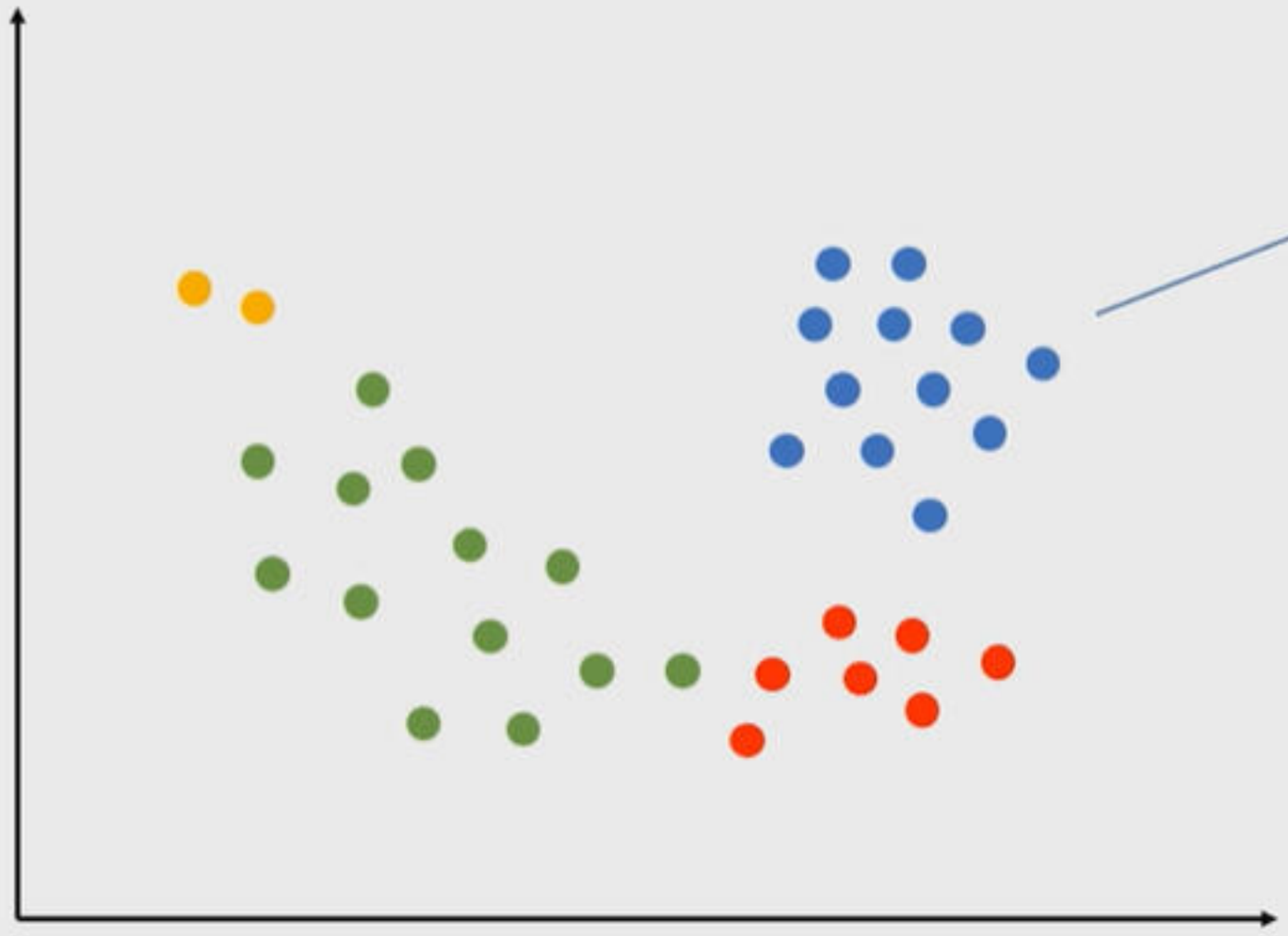
Unsupervised Learning: Clustering

aws training and certification



Unsupervised Learning: Clustering

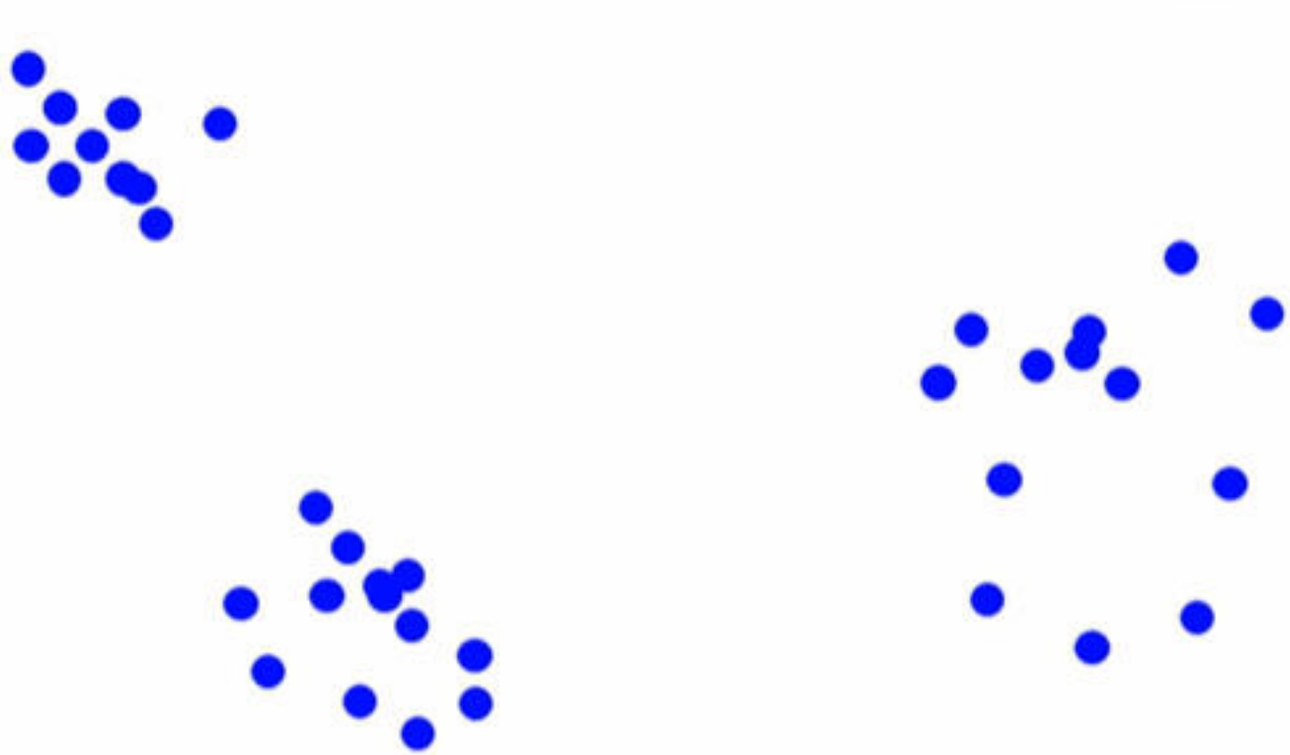
aws training and certification



But what do they represent???

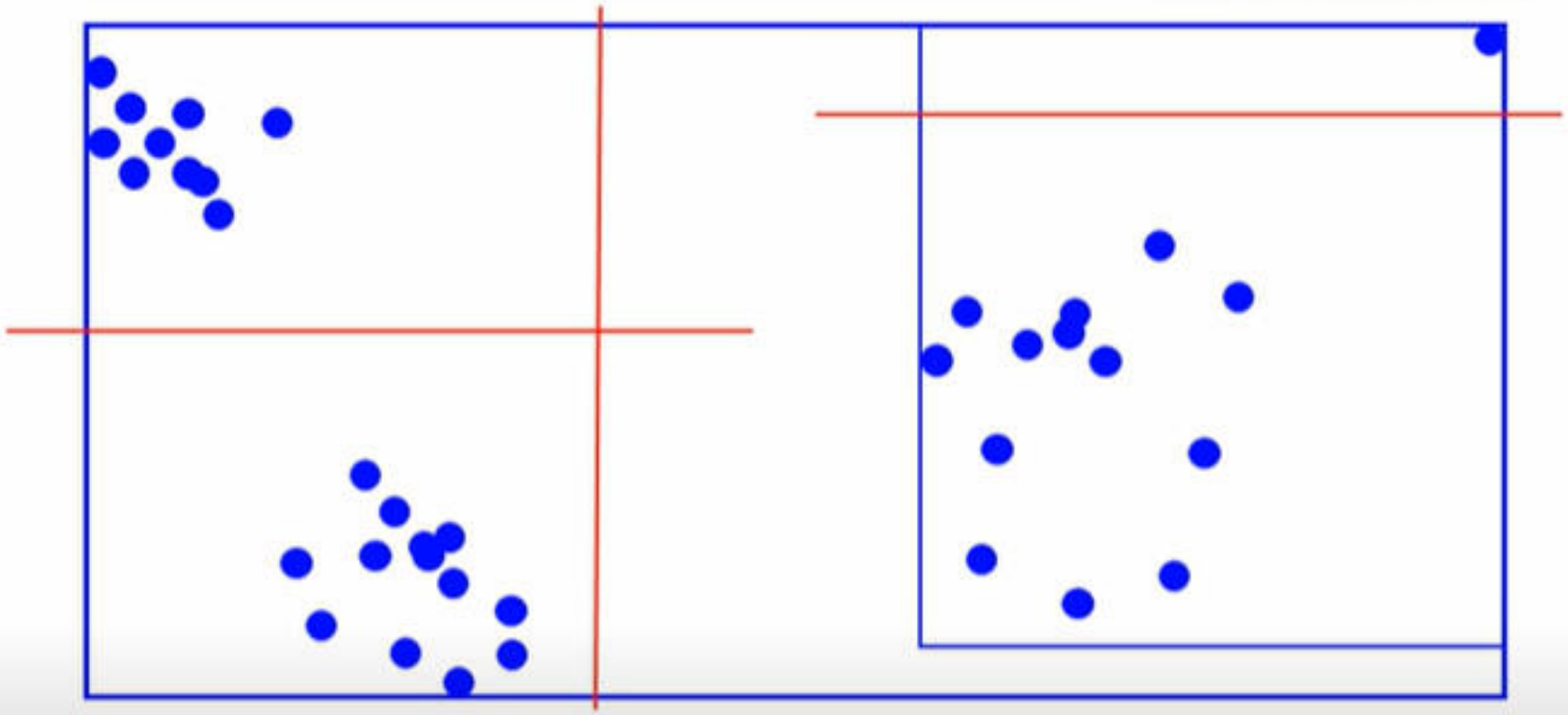
Anomalies: Random Cut Forest

aws training and certification



Anomalies: Random Cut Forest

aws training and certification

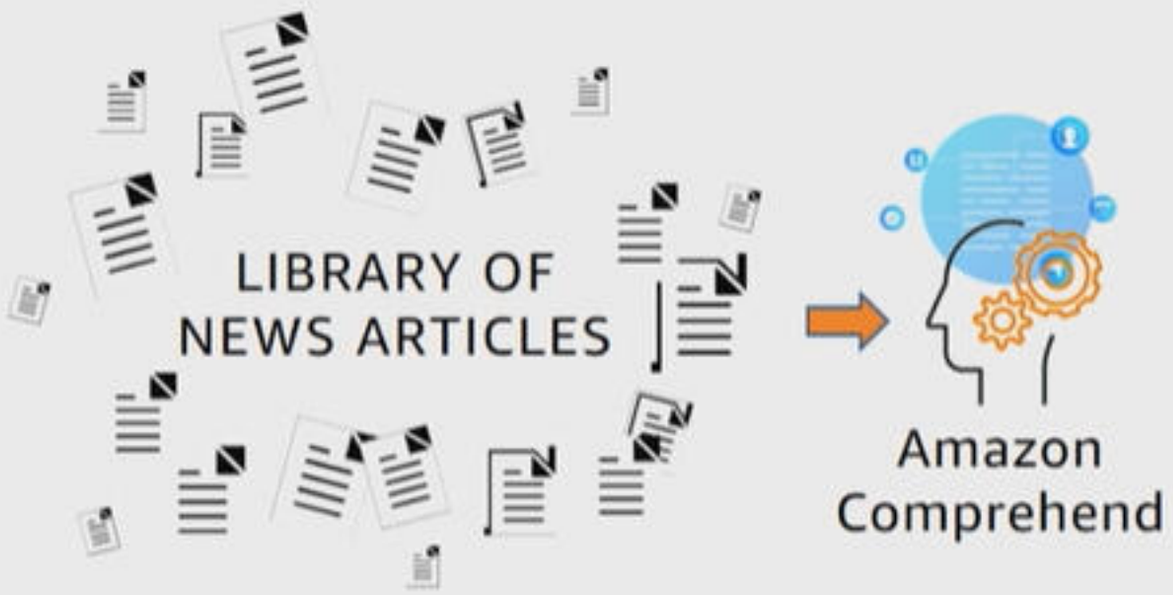


Range-biased Cut

Recurse: The cutting stops when each point is isolated.

Unsupervised ML: Topic Modeling

aws training and certification



Unsupervised ML: Topic Modeling

aws training and certification

Topic	Term	Weight
000	team	0.118533
000	game	0.106072
000	player	0.031625
000	season	0.023633

Unsupervised Algorithms

aws training and certification

- K-Means Clustering
  - Improved Web-scale k-means clustering
- Principal Component Analysis (PCA)
  - Reduces dimensionality within dataset
  - Precursor to Supervised Learning
- Latent Dirichlet Allocation (LDA)
  - Topic Modelling
- Anomaly Detection

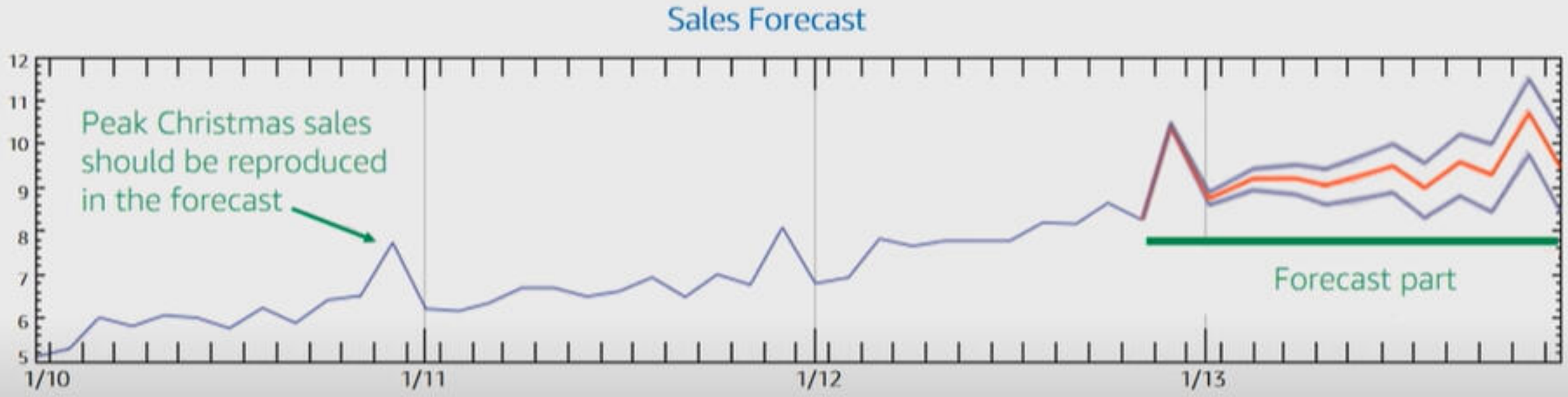
Unsupervised Algorithms

aws training and certification

- K-Means Clustering
  - Improved Web-scale k-means clustering
- Principal Component Analysis (PCA)
  - Reduces dimensionality within dataset
  - Precursor to Supervised Learning
- Latent Dirichlet Allocation (LDA)
  - Topic Modelling
- Anomaly Detection

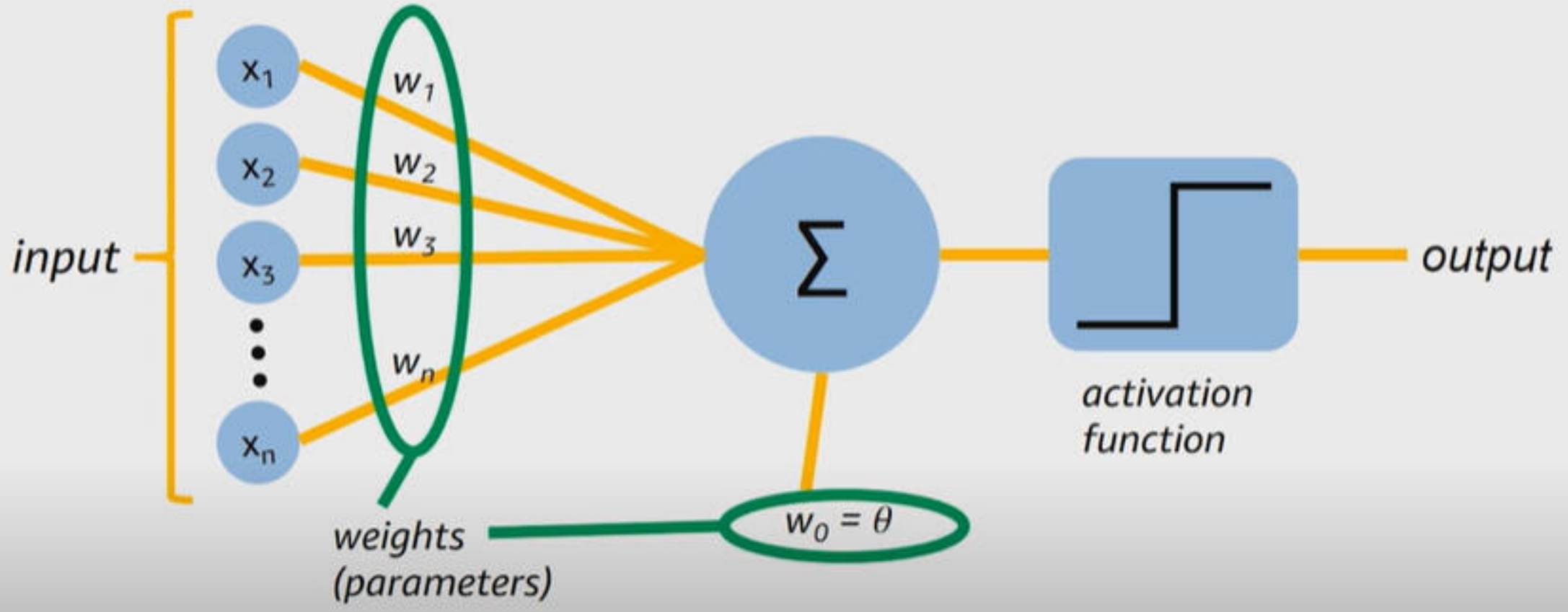
Time Series Forecasting

aws training and certification



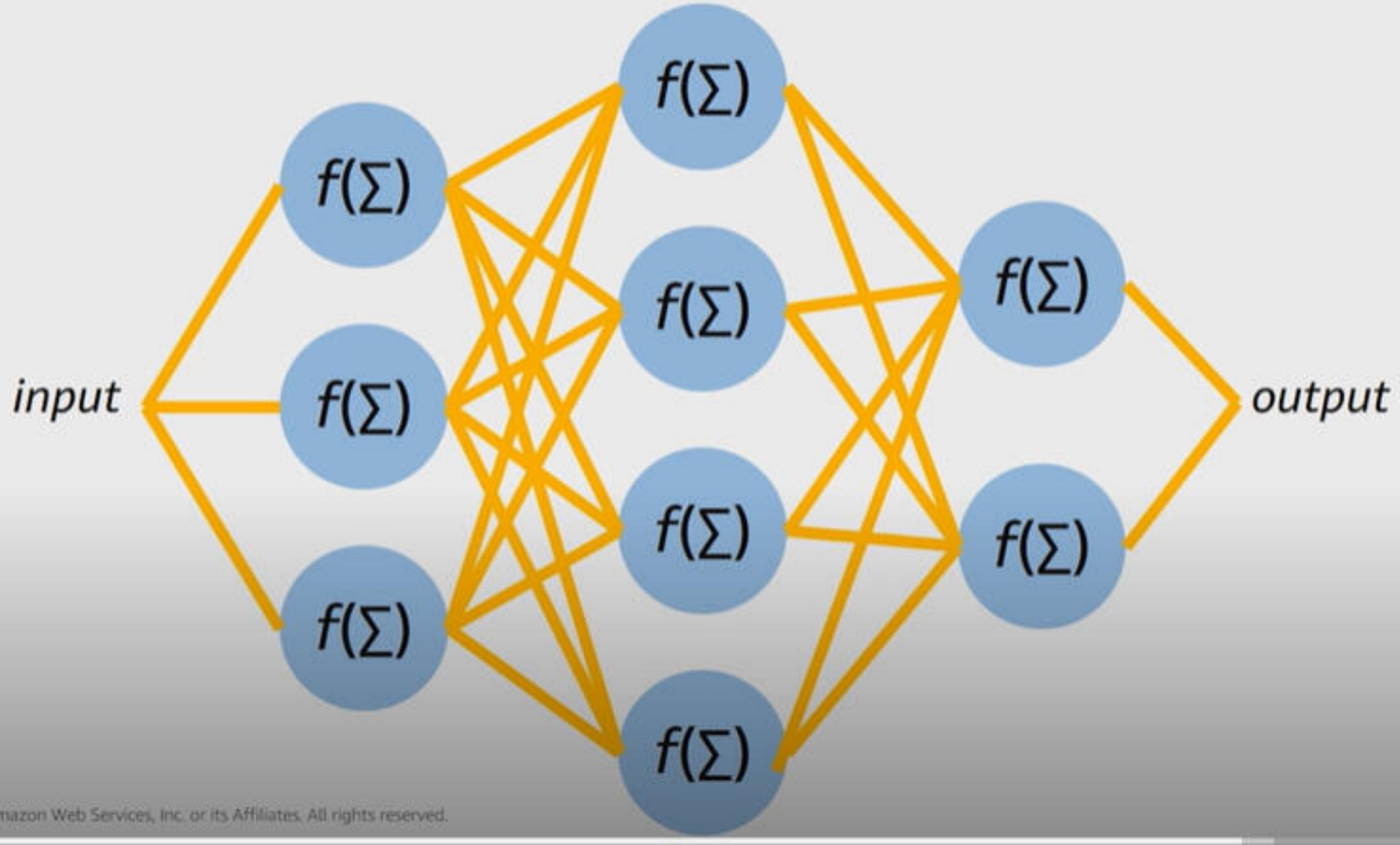
Deep Learning: Neurons

aws training and certification



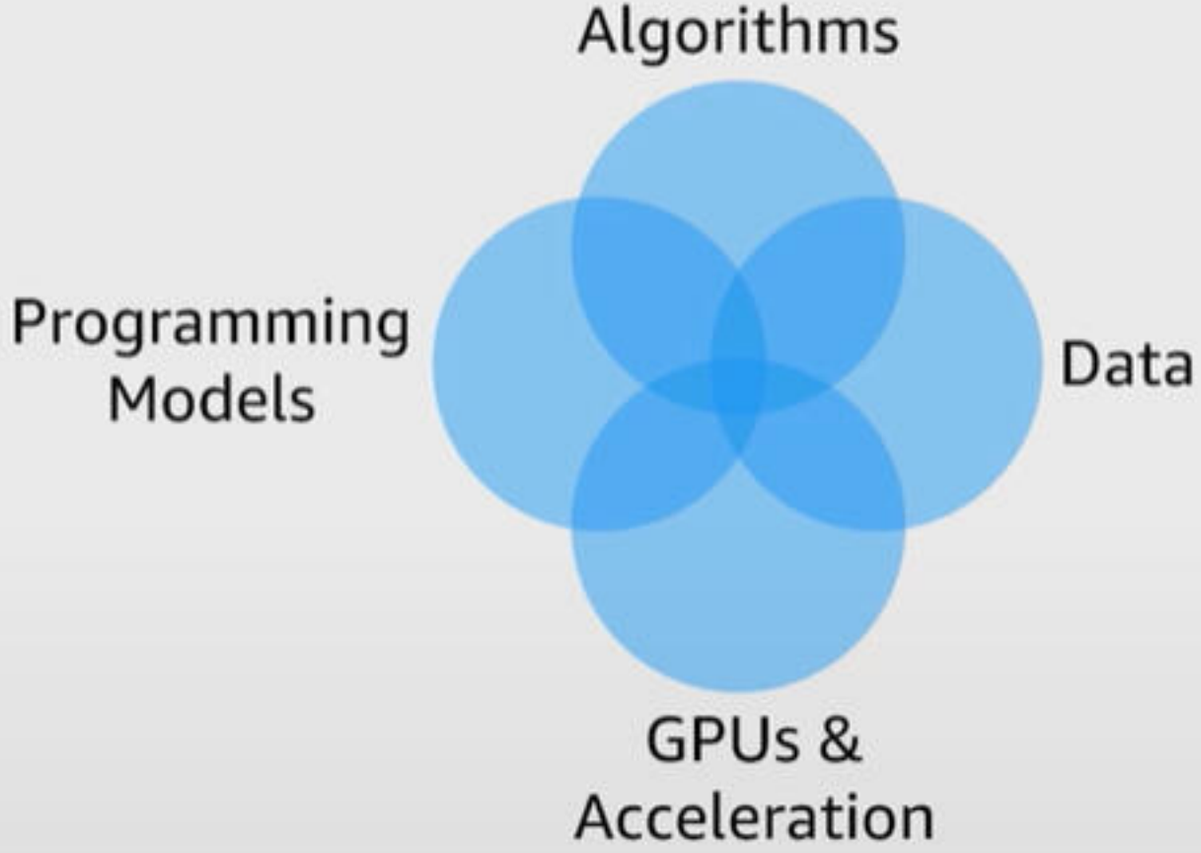
Deep Learning: Neural Networks

aws training and certification



The Advent of Deep Learning

aws training and certification



The Advent of Deep Learning

aws training and certification

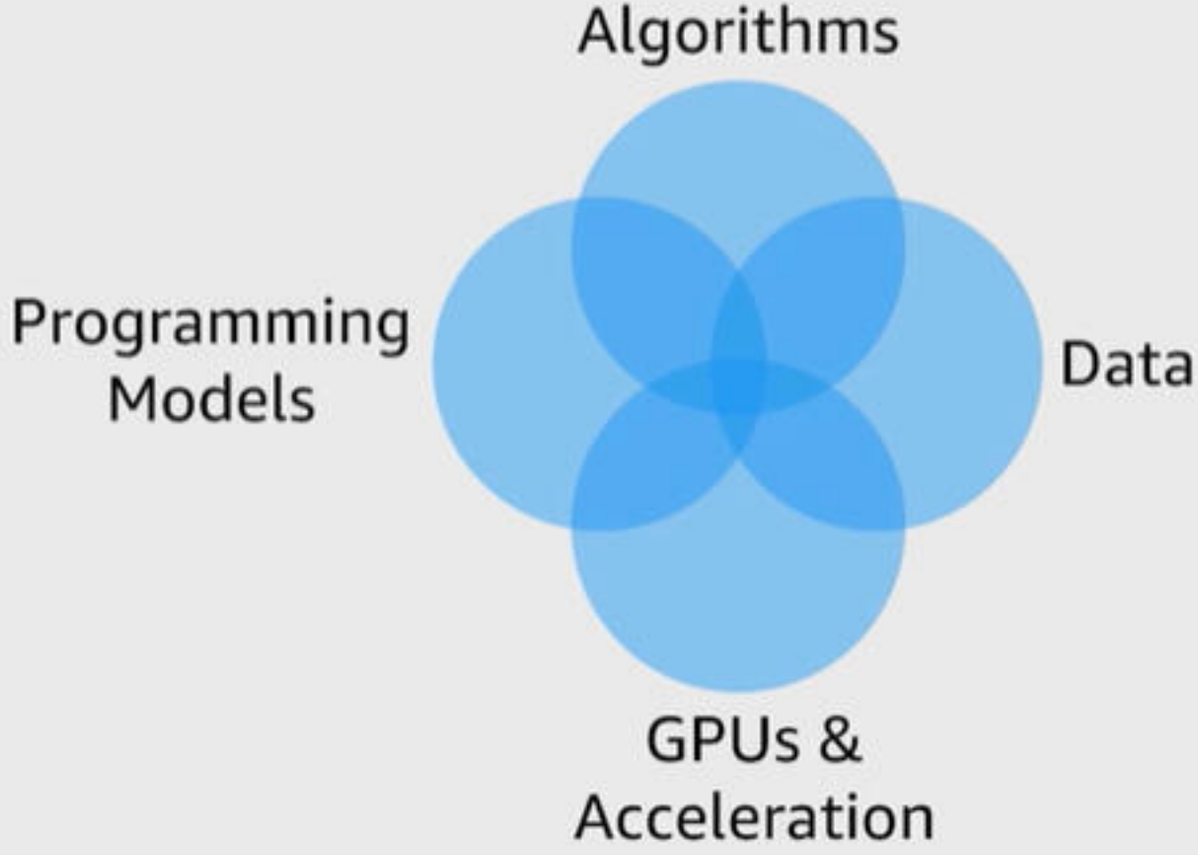
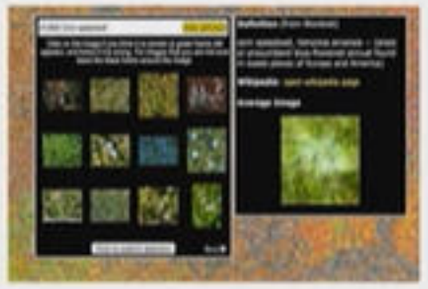


image understanding



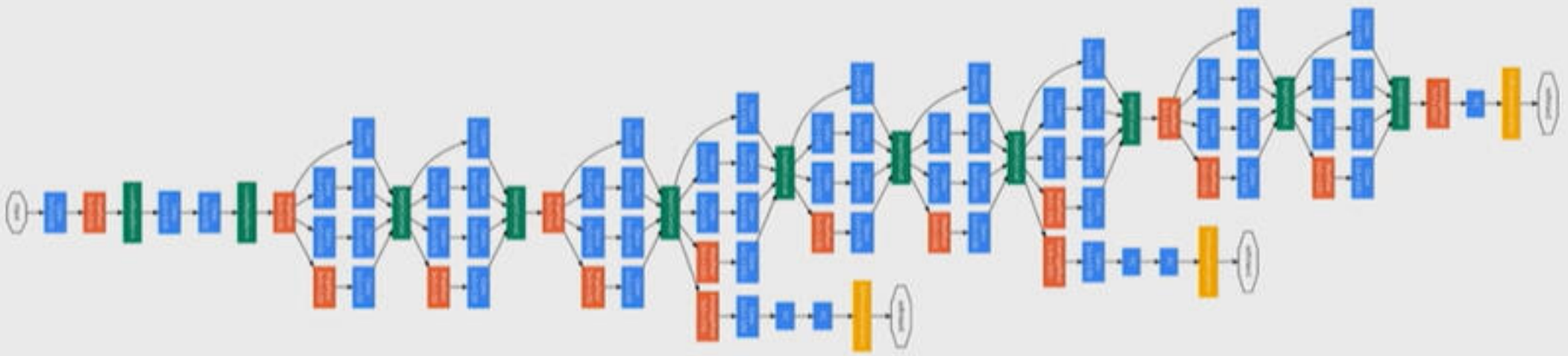
natural language processing

speech recognition

autonomy

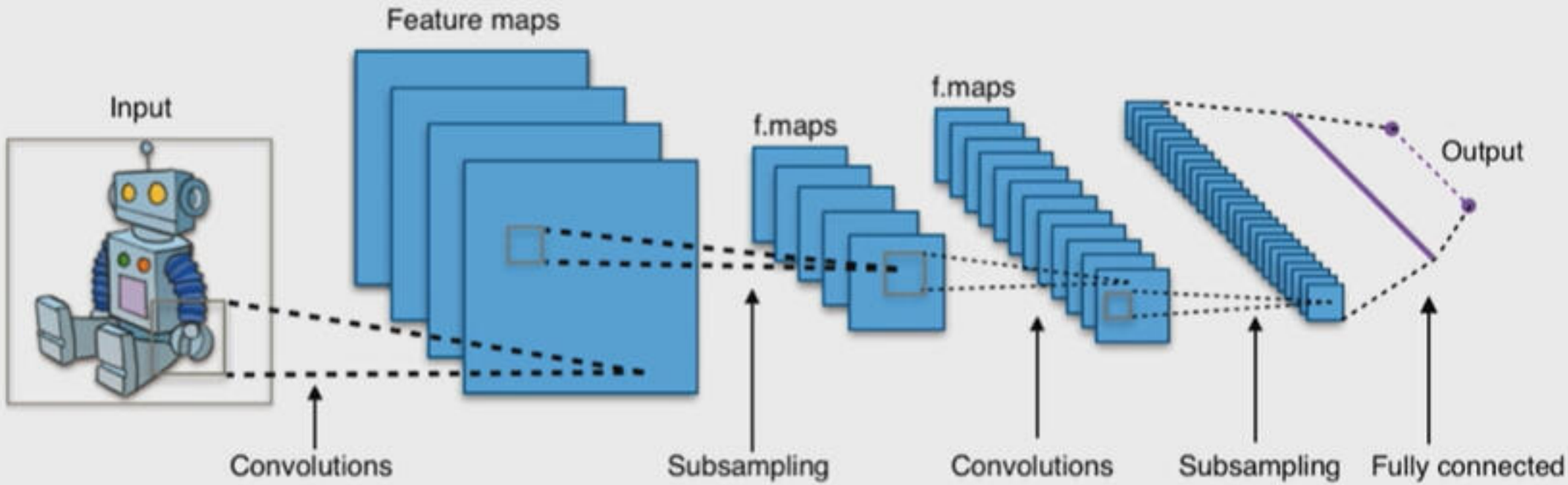
Deep Neural Networks

aws training and certification



Convolutional Neural Networks (CNNs)

aws training and certification



Convolutional Neural Networks (CNNs)

aws training and certification

Object recognition, image classification

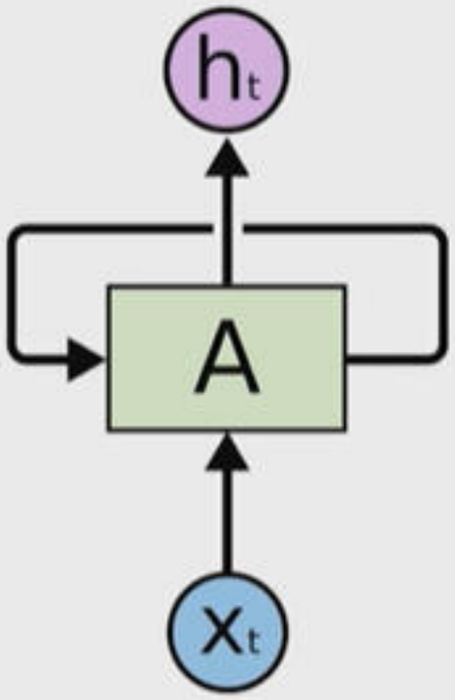
Semantic segmentation

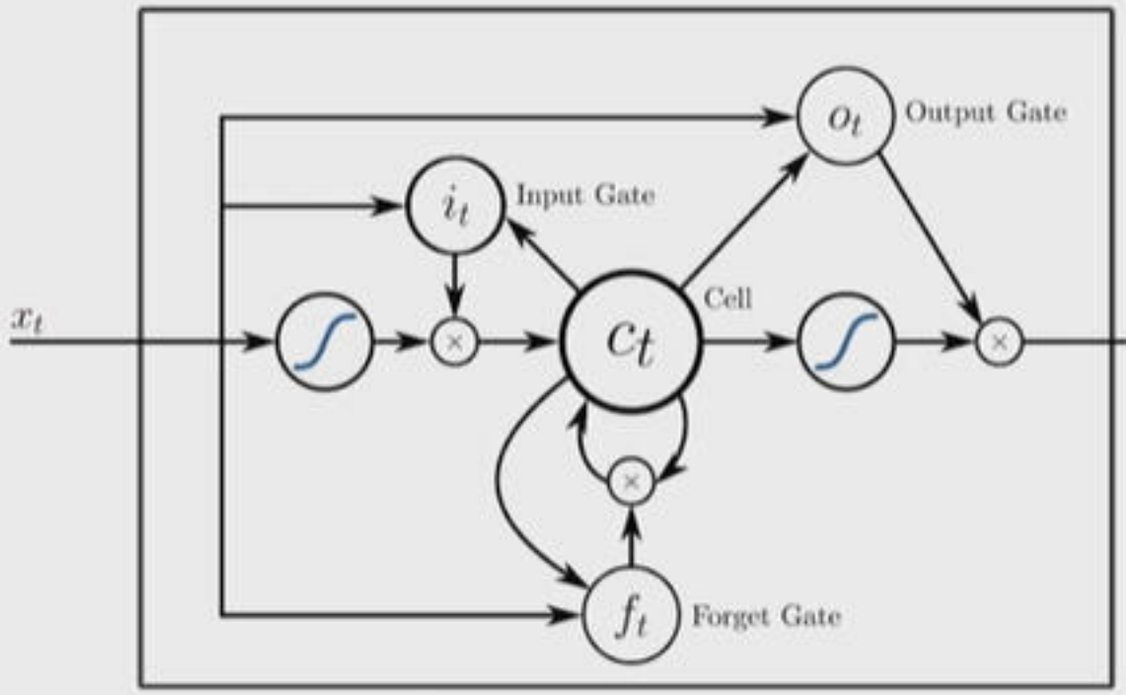
Artistic style transfer

Meow Generator

Recurrent Neural Networks (RNNs)

aws training and certification



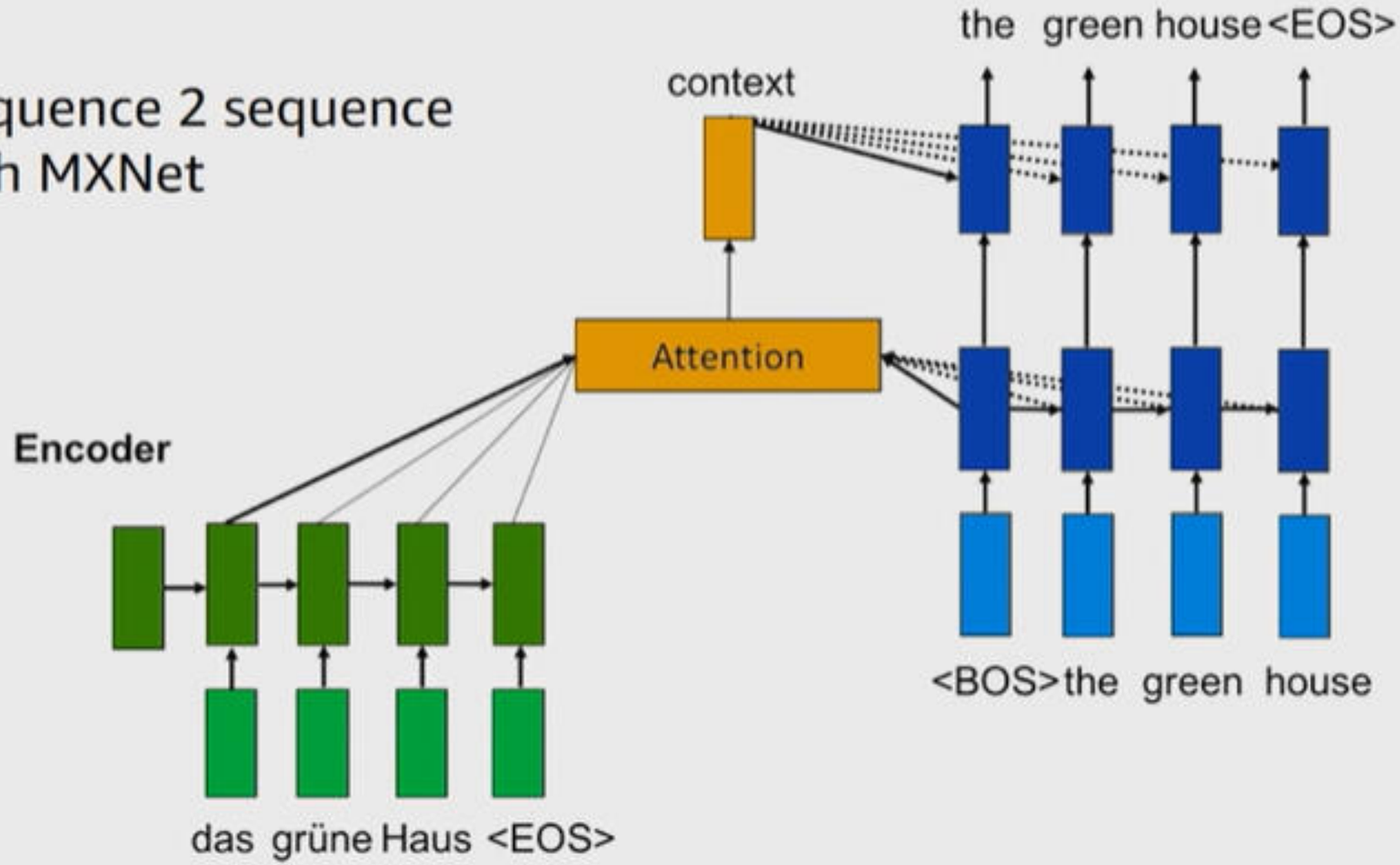


Long Short-Term Memory
or LSTM network

Recurrent Neural Networks (RNNs)

aws training and certification

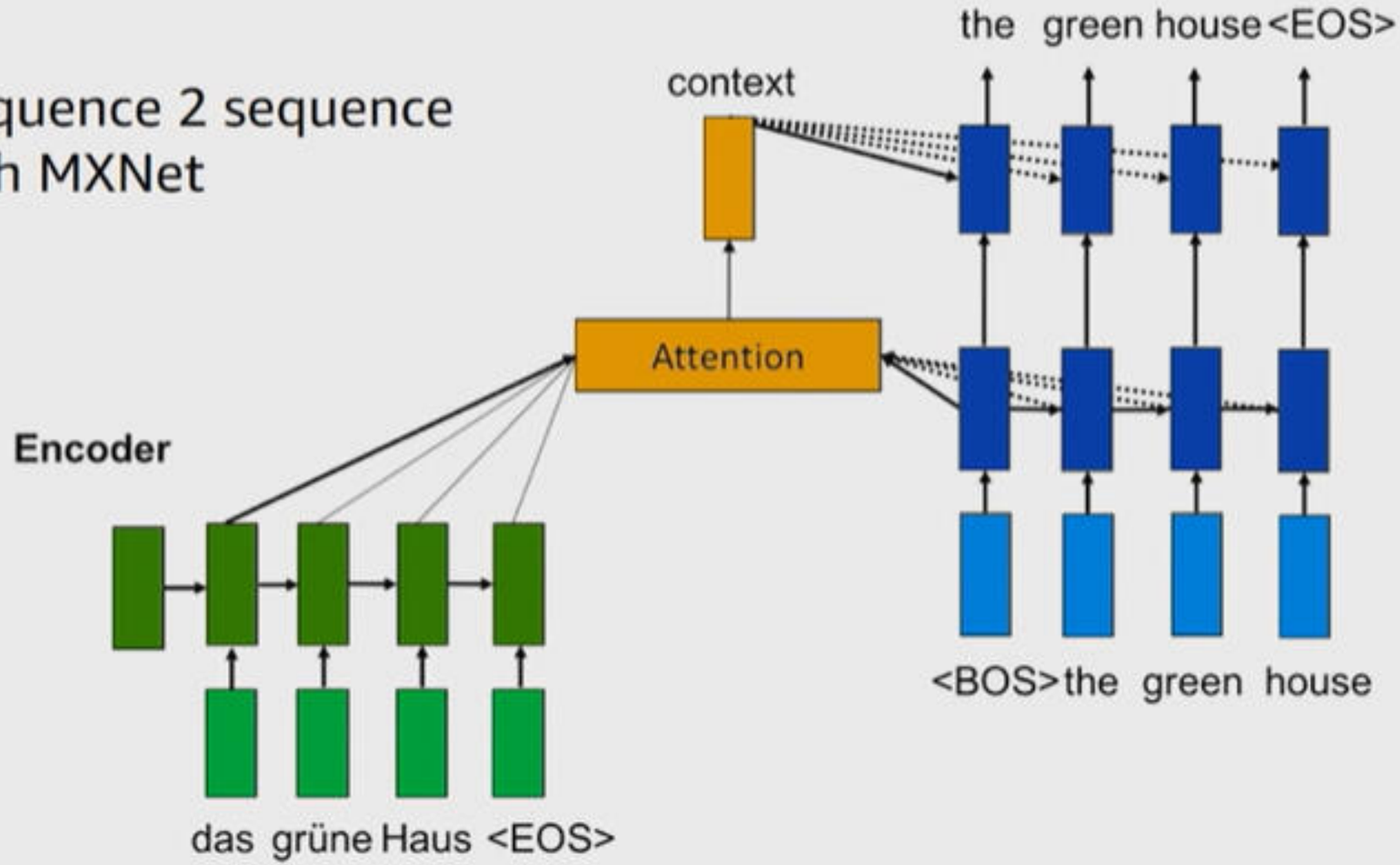
Sockeye – sequence 2 sequence modeling with MXNet



Recurrent Neural Networks (RNNs)

aws training and certification

Sockeye – sequence 2 sequence modeling with MXNet



Different Network Topologies

aws training and certification

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

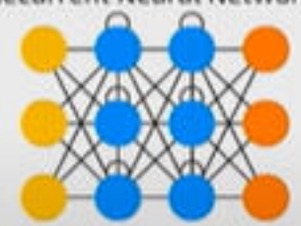
Perceptron (P)

Feed Forward (FF)

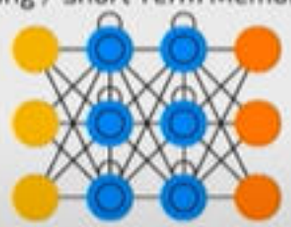
Radial Basis Network (RBF)

Deep Feed Forward (DFF)

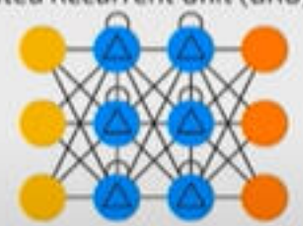
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



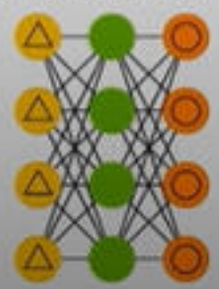
Gated Recurrent Unit (GRU)



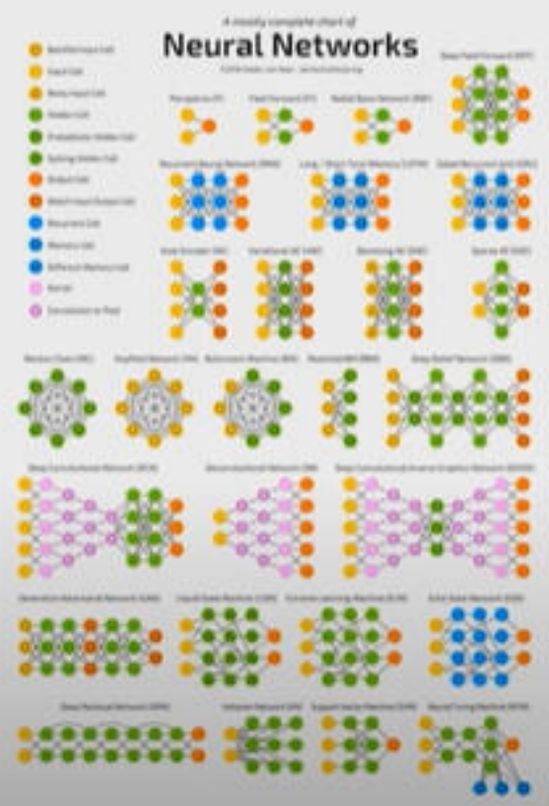
Auto Encoder (AE)

Variational AE (VAE)

Denoising AE (DAE)



Sparse AE (SAE)



Deep Learning Algorithms

aws training and certification

Image Classification

- Convolutional Neural Network (ResNet)

Sequence to Sequence (seq2seq)

- RNN for text summarization, translation, TTS

Neural Topic Modelling (NTM)

DeepAR Forecasting

- Time Series Prediction

SageMaker