TELEPATHY LABS

Telepathy Labs Stella™ Transcribe User Guide

Version 1.1.0

Abstract

This document describes the Telepathy Labs *Stella™ Transcribe* solution and tools. The solution includes the baseline **T**elepathy **L**abs **T**ranscription **S**ervice, hereafter referred to as **TLTS**, needed to convert digital audio speech signals into text, and the **T**elepathy **L**abs **A**SR **C**ustomization **S**ervice, hereafter referred to as **TLACS**, needed to improve the performance of **TLTS** by creating new language models. The document also includes installation instructions and external references section.

Contents

1 Telepathy Labs Stella™ Transcribe: Synopsis

1.1 Overview

The *Stella™ Transcribe* solution allows a user to determine a textual representation derived from spoken material contained in media such as audio and video files. This process is generally referred to as "Transcribing the speech signal" i.e., producing a text-like representation from speech signal in a variety of formats.

The spoken material transcription process makes use of powerful AI technologies which are referred to later in this document as ASR - Automatic Speech Recognition, and are implemented through a set of components in a service called TLTS (**T**elepathy **L**abs **T**ranscription **S**ervice) described in detail in section 3.

In addition to TLTS, the *Stella™ Transcribe* solution comes with an important add-on, a customization service designed to improve the accuracy of the transcription. This customization service is also made available as a set of components in a service called TLACS (**T**elepathy **L**abs **A**SR **C**ustomization **S**ervice) described in detail in section 5.

1.2 Quick Setup

The *Stella™ Transcribe* solution packaging is suitable for on-premise installation, therefore not requiring any external services to perform at its best.

The solution is deployed as a set of docker images coordinated into a stack of components defined in a suitable docker-compose file. To to perform a quick setup, we provide a specific section in this documentation that allows you to pull docker images from a defined docker repository, install and run them to create a baseline TLTS instance, with a default Language Model, a baseline TLACS instance allowing you to create custom language models, and to connect the two services in a local instance, see section 2.

2 Stella™ Transcribe: Installation Instructions

These are the instructions to create an instance of TLTS on a Host Machine together with TLACS.

2.1 Requirements

- Python 3.5.x (or later)
- Docker version 19.03.4, build 9013bf583a (or later)

2.2 Deploy on Amazon EC2 Instance

2.2.1 Prerequisites

- (Optional) Create a new AWS EC2 Instance.
  - Minimum Required Specifications for testing purpose
    - AMI: Amazon Linux 2
    - Instance Type: t2.large, 2CPU, 8GB Memory
    - Storage: >100GB
  - Minimum Required Specifications for production, the ASR latency will depend on the instance type.
    - AMI: Amazon Linux 2
    - Instance Type: m4.10xlarge, 40CPU, 160GB Memory
    - Storage: >100GB
  - Ensure that the Instance is under a Security Group that has the following TCP ports open
    - 22 for SSH
    - 443 for HTTPS
    - 8082 to interact with TLTS Service
    - 8083 to download transcribed output of TLTS Service
    - 3000 to interact with TLACS
    - 3001 to visualize processed output of TLACS
    - 2049 for NFS mount of Amazon EFS, The Source could be a Security Group ID
  - Copy the SSH private key to your local machine. This will be used to log into the EC2 instance.
- Use the SSH private key to log into the AWS EC2 Instance.

```
$ ssh -i ssh_key.pem ec2-user@"Public DNS (IPv4) of EC2 Instance"
```

- Install and Start Docker service on the Amazon EC2 Instance.

```
$ sudo yum update -y
$ sudo amazon-linux-extras install docker
$ sudo usermod -a -G docker ec2-user  # add ec2-user to docker group
$ sudo service docker start
$ exit # Log out and log back in again
       # to pick up the new docker group permissions.
```

  - For more information please refer to AWS Documentation: "Docker Basics for Amazon ECS"

- Install python3, python3-pip, and set up a virtualenv.

```
# Install python3, python3-pip and its dependencies, default is python3.7
$ sudo yum -y install python3 python3-pip
$ pip3 install --user virtualenv
```

2.2.2 Mount Amazon EFS in Amazon EC2 instance

- Obtain details related to Amazon EFS that will be mounted on the Amazon EC2 machine.
  - Log into your AWS Management Console. Open EFS → tl-kaden-test-efs. (Your EFS File System Name)
  - Note down the File system ID e.g., fs-xxxxxxxx. This will be used to mount the EFS.
  - Ensure that the EFS and your EC2 instances are in the same Security Group. Also, ensure that EFS and EC2 instances are in the same availability zone.
- Use the SSH private key to log into AWS EC2 Instance.

```
$ ssh -i ssh_key.pem ec2-user@"Public DNS (IPv4) of EC2 Instance"
```

- Install EFS mount utils.

```
$ sudo yum install -y amazon-efs-utils
```

- Mount EFS to a local directory.

```
$ mkdir /home/ec2-user/efs_data
$ sudo mount -t efs -o tls fs-xxxxxxxx:/ /home/ec2-user/efs_data
```

- (Optional) To Unmount EFS.

```
$ sudo umount /home/ec2_user/efs_data
```

- The audio files to transcribe could be stored as follows

```
/home
└── ec2-user
    └── efs-data
        └── GroupTherapies
            ├── Session1
            │   ├── member_1.wav
            │   ├── member_2.wav
            │   ├── member_3.wav
            │   ├── member_4.wav
            │   ├── member_5.wav
            │   └── member_6.wav
            └── Session2
                ├── member_1.wav
                ├── member_2.wav
                └── member_3.wav
```

2.2.3 Setup TLTS and TLACS

- Use the SSH private key to log into the machine.

```
$ ssh -i ssh_key.pem ec2-user@"Public DNS (IPv4) of EC2 Instance"
```

- Extract the files delivered to you to your current working directory i.e., ec2-user The directory structure on the Amazon EC2 instance is shown below

```
/home
└── ec2-user
    ├── efs-data
    ├── ssl
    ├── docker-compose-kaden-fs.yml
    ├── env.txt
    ├── env_common.txt
    ├── env_mongo.txt
    └── data-storage
```

- Create a Python Virtual Environment and Activate it.

```
$ virtualenv -p python3 /home/ec2-user/venv
$ source /home/ec2-user/venv/bin/activate
```

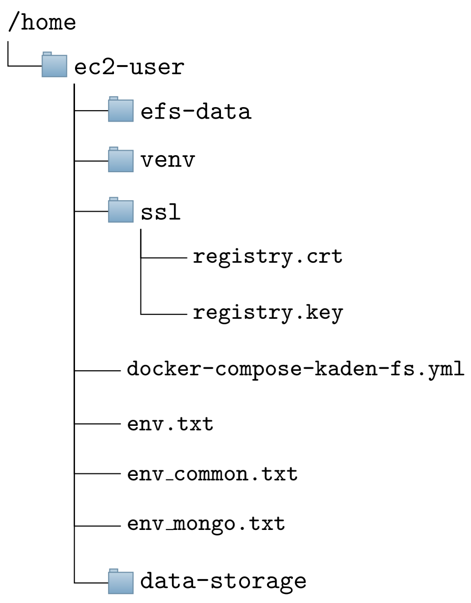
- Install docker-compose

```
(venv)$ pip install docker-compose==1.23.*
#----or-----
(venv)$ pip install docker-compose==1.23
```

- Log in to Docker Repository to pull images

```
(venv)$ docker login docker.exm-platform.com -u Authorized-Email-Address
//Enter the Password
```

- Copy SSL certificates (registry.crt, registry.key) into the ssl folder. The directory structure on the Amazon EC2 instance is shown below



- TLTS and TLACS utilize applications like mongodb, minio, redis, and postgress to store information in a persistent manner. These applications store data at a fixed directory location and are specified in the env.txt file. The default directory locations are shown below:

/home/ec2-user/env.txt

```
MONGODB_VOLUME=/home/ec2-user/data-storage/mongodb/
MINIO_VOLUME=/home/ec2-user/data-storage/minio/
REDIS_VOLUME=/home/ec2-user/data-storage/redis/
TRANSCRIPTION_DB_VOLUME=/home/ec2-user/data-storage/postgres/
```

- The location of audio files that are accessible by TLTS service is also specified in the env.txt file. These audio files were obtained by mounting an Amazon EFS storage described in Section Mount Amazon EFS

/home/ec2-user/env.txt

```
MONGODB_VOLUME=/home/ec2-user/data-storage/mongodb/
MINIO_VOLUME=/home/ec2-user/data-storage/minio/
REDIS_VOLUME=/home/ec2-user/data-storage/redis/
TRANSCRIPTION_DB_VOLUME=/home/ec2-user/data-storage/postgres/
TRANSCRIPTION_DATA_VOLUME=/home/ec2-user/efs-data/
```

- To access files generated by TLACS, edit the env.txt file and update the value of parameter *EXTERNAL_HOSTNAME* to the public hostname of your EC2 instance.

/home/ec2-user/env.txt

```
MONGODB_VOLUME=/home/ec2-user/data-storage/mongodb/
MINIO_VOLUME=/home/ec2-user/data-storage/minio/
REDIS_VOLUME=/home/ec2-user/data-storage/redis/
TRANSCRIPTION_DB_VOLUME=/home/ec2-user/data-storage/postgres/
TRANSCRIPTION_DATA_VOLUME=/home/ec2-user/efs-data/
EXTERNAL_HOSTNAME=ec2-X-XXX-XXX-XXX.compute-1.amazonaws.com
```

- For Docker Compose to access the values specified in the env.txt, copy them to a .env file.

```
(venv)$ cp env.txt .env
```

- Obtain the Required Docker images

```
(venv)$ docker-compose -f /home/ec2-user/docker-compose-kaden-fs.yml pull
```

2.3 Start and Stop Stella™ Transcribe

- To Start Telepathy Labs Transcription Service together with Telepathy Labs ASR Customization Service run the following command:

```
(venv) docker-compose -f /home/ec2-user/docker-compose-kaden-fs.yml up -d
```

- To Stop Telepathy Labs Transcription Service together with Telepathy Labs ASR Customization Service run the following command:

```
(venv) docker-compose -f /home/ec2-user/docker-compose-kaden-fs.yml down -v
```

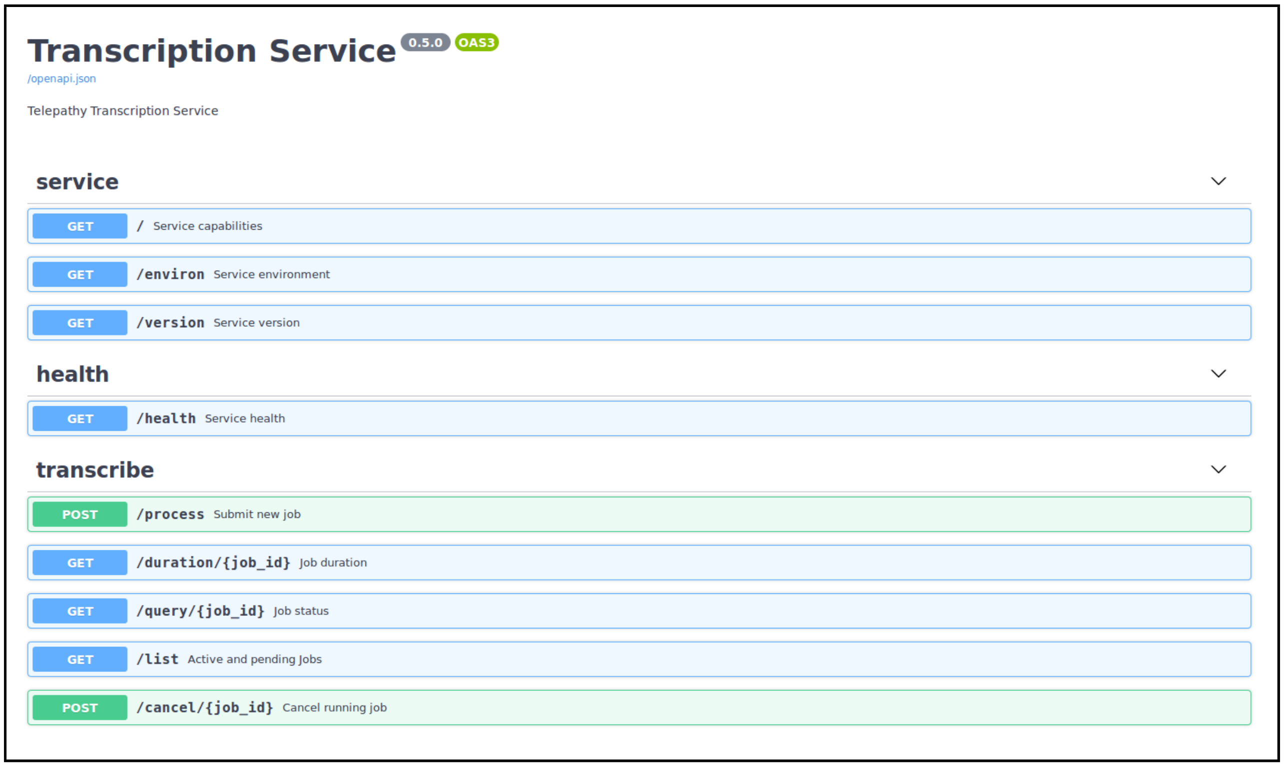
3 Telepathy Labs Transcription Service (TLTS) Usage

The TLTS allows the user to derive a textual representation from spoken material contained in media such as audio and video files. This process is generally referred to as "Transcribing the speech signal".

3.1 Transcription Services OpenAPI interface

The TLTS installation process provides a baseline interface, following the OpenAPI standard, that can be used to verify the correct installation. In the following we describe the usage of this interface for a quick test, and detail important aspects of some key parameters.

1. To access the TLTS OpenAPI interface, point your browser to `https://"PublicDNS(IPv4)ofEC2Instance":8082/docs`.



Transcription Service 0.5.0 OAS3
/openapi.json

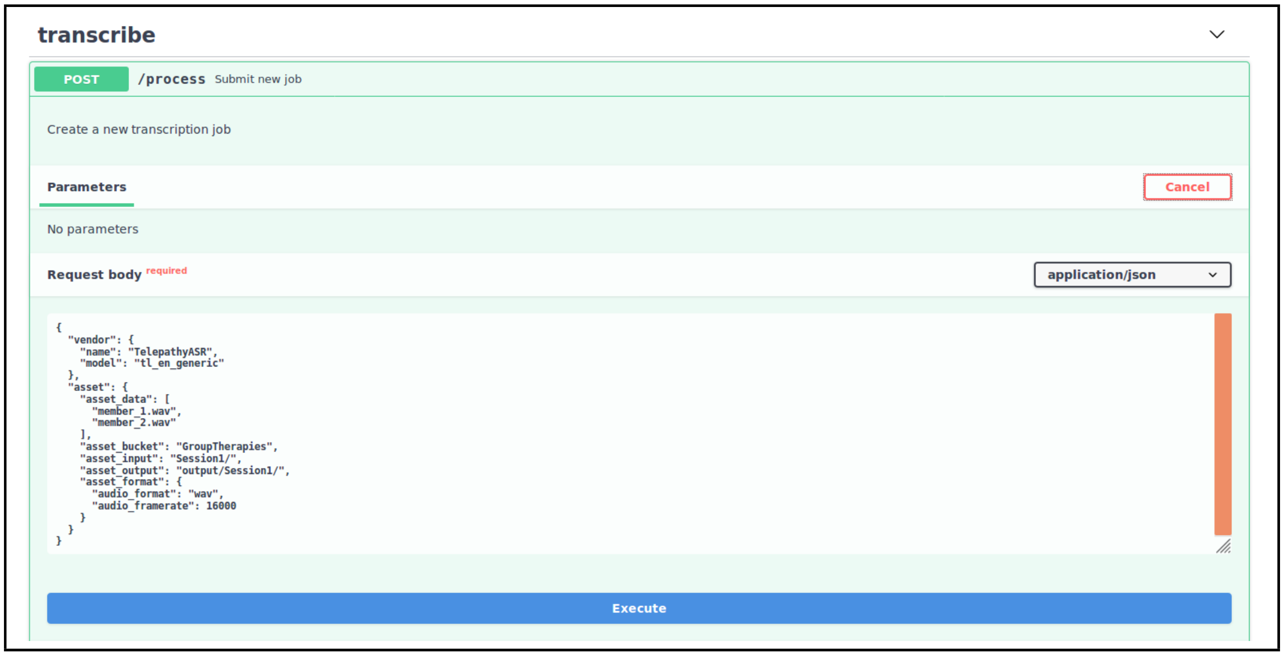
Telepathy Transcription Service

service
GET / Service capabilities
GET /environ Service environment
GET /version Service version

health
GET /health Service health

transcribe
POST /process Submit new job
GET /duration/{job_id} Job duration
GET /query/{job_id} Job status
GET /list Active and pending Jobs
POST /cancel/{job_id} Cancel running job

2. Submit a Transcription Job.
   - Click on the transcribe/process to view the API specification.
   - Click on the Try it out button to prepare parameters for a sample transcription job.
   - Transcribe the audio of two sample signals provided by default by the install process, *(member_1.wav and member_2.wav)* in the *Session1* of *Group Therapy* folder. The corresponding API POST request is shown below. The transcription results for each audio file are stored in the folder specified in the *asset_output* key.
   - Click on the Execute button to submit the sample transcription job.



transcribe

POST /process Submit new job

Create a new transcription job

Parameters

Cancel

No parameters

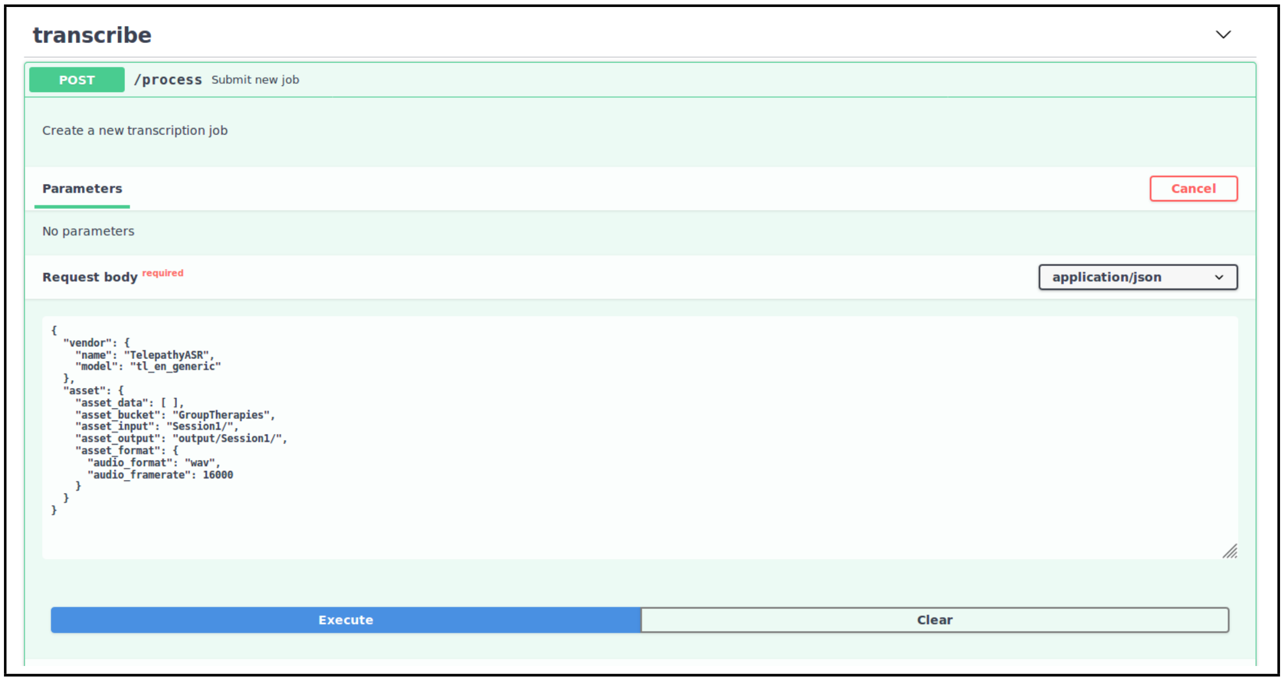
Request body required

application/json

```
{
  "vendor": {
    "name": "TelepathyASR",
    "model": "tl_en_generic"
  },
  "asset": {
    "asset_data": [
      "member_1.wav",
      "member_2.wav"
    ],
    "asset_bucket": "GroupTherapies",
    "asset_input": "Session1/",
    "asset_output": "output/Session1/",
    "asset_format": {
      "audio_format": "wav",
      "audio_framerate": 16000
    }
  }
}
```

Execute

- To Transcribe the audio of all members of Group Therapy *Session1*, the API POST request is shown below. The transcription of each audio file is stored in the folder specified in the *asset_output* key.



transcribe

POST /process Submit new job

Create a new transcription job

Parameters

Cancel

No parameters

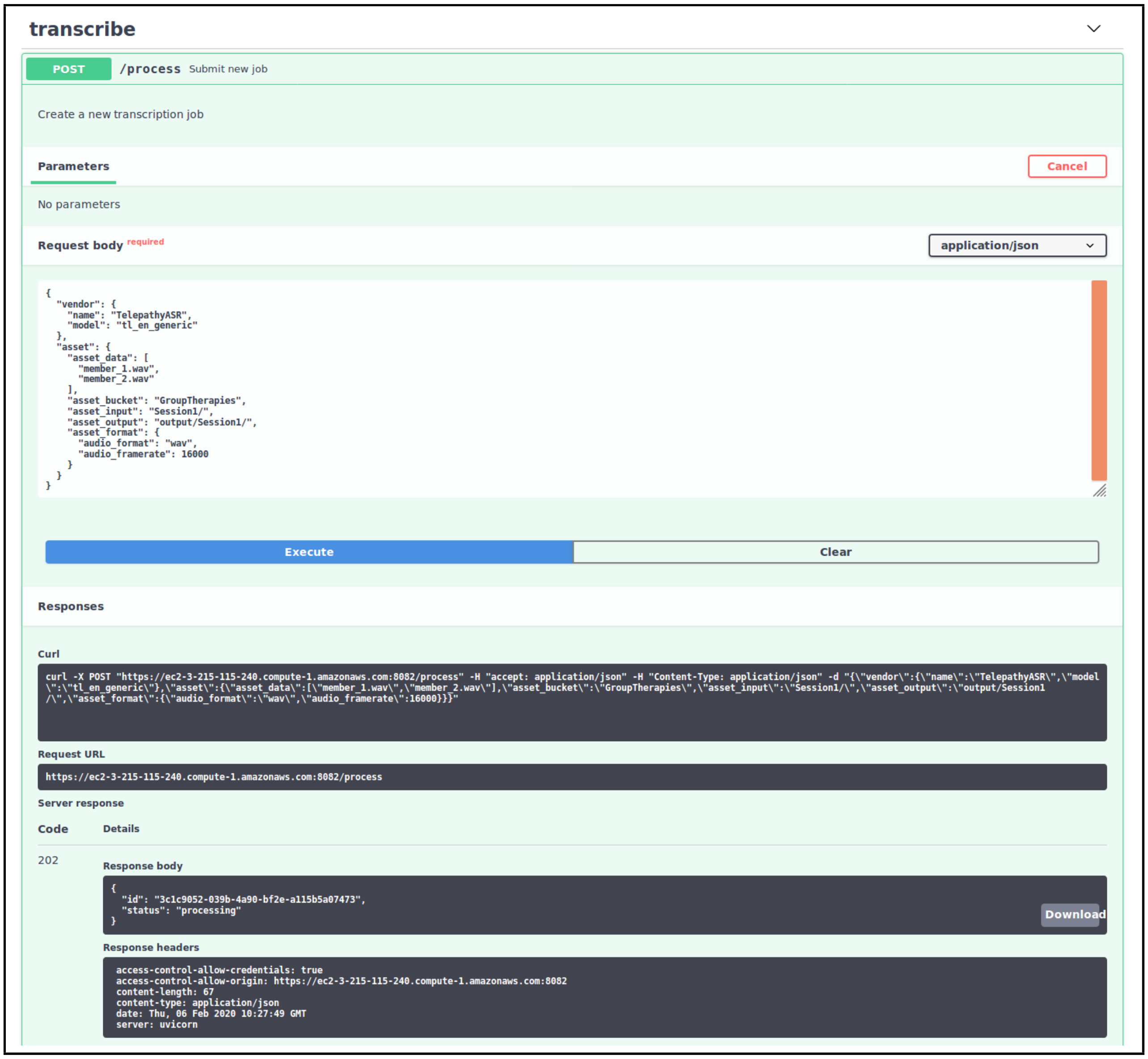
Request body required

application/json

```
{
  "vendor": {
    "name": "TelepathyASR",
    "model": "tl_en_generic"
  },
  "asset": {
    "asset_data": [ ],
    "asset_bucket": "GroupTherapies",
    "asset_input": "Session1/",
    "asset_output": "output/Session1/",
    "asset_format": {
      "audio_format": "wav",
      "audio_framerate": 16000
    }
  }
}
```

Execute Clear

- In response to a Transcription Job, TLTS will provide an id and status of the transcription job. We can track the status of the transcription job using the id.



```
{
  "vendor": {
    "name": "TelepathyASR",
    "model": "tl_en_generic"
  },
  "asset": {
    "asset_data": [
      "member_1.wav",
      "member_2.wav"
    ],
    "asset_bucket": "GroupTherapies",
    "asset_input": "Session1/",
    "asset_output": "output/Session1/",
    "asset_format": {
      "audio_format": "wav",
      "audio_framerate": 16000
    }
  }
}
```

Curl

```
curl -X POST "https://ec2-3-215-115-240.compute-1.amazonaws.com:8082/process" -H "accept: application/json" -H "Content-Type: application/json" -d "{\"vendor\":{\"name\":\"TelepathyASR\",\"model
\":\"tl_en_generic\"},\"asset\":{\"asset_data\":[\"member_1.wav\",\"member_2.wav\"],\"asset_bucket\":\"GroupTherapies\",\"asset_input\":\"Session1/\",\"asset_output\":\"output/Session1
/\",\"asset_format\":{\"audio_format\":\"wav\",\"audio_framerate\":16000}}}"
```

Request URL

```
https://ec2-3-215-115-240.compute-1.amazonaws.com:8082/process
```

Server response

Code Details

202

Response body

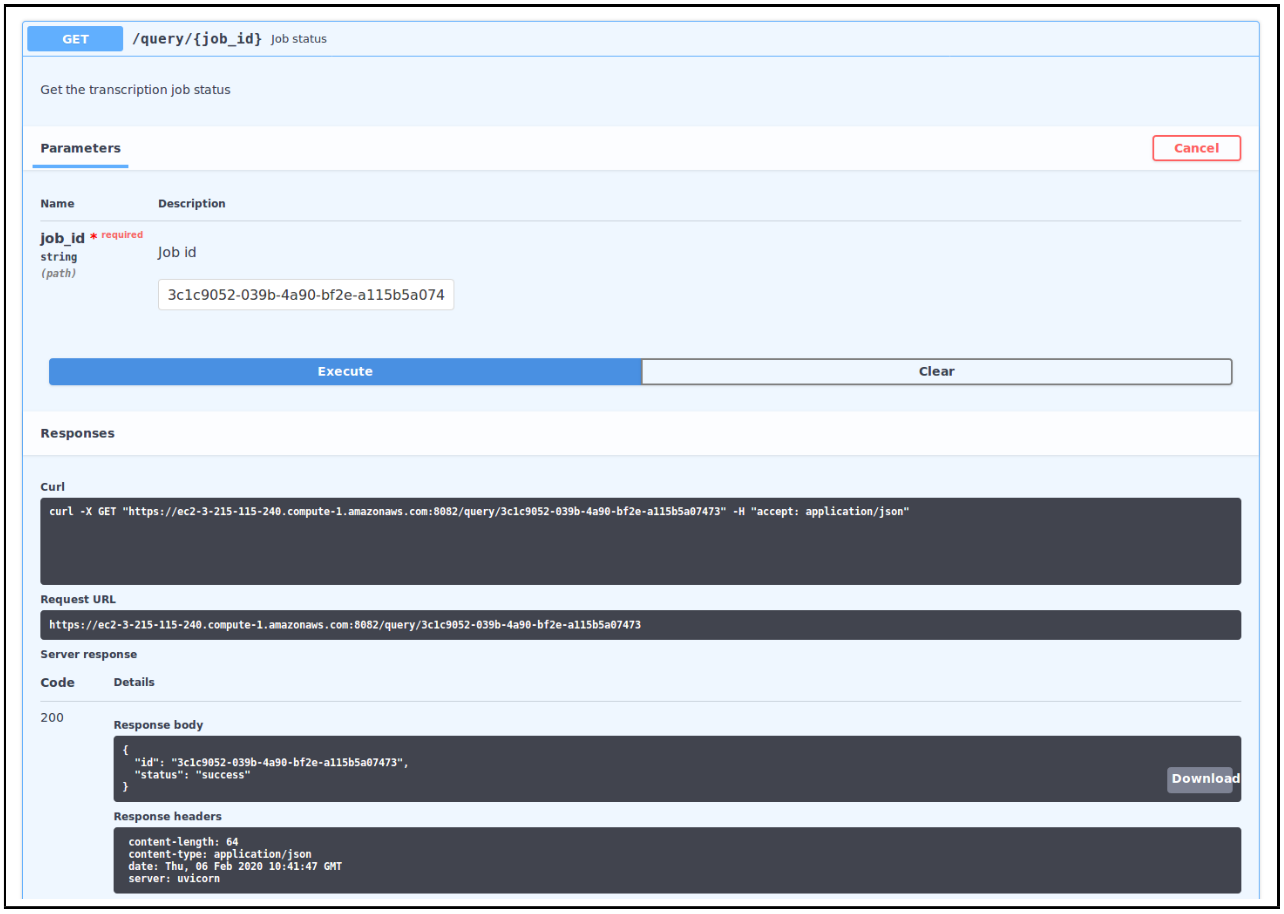
```
{
  "id": "3c1c9052-039b-4a90-bf2e-a115b5a07473",
  "status": "processing"
}
```

Response headers

```
 access-control-allow-credentials: true
 access-control-allow-origin: https://ec2-3-215-115-240.compute-1.amazonaws.com:8082
 content-length: 67
 content-type: application/json
 date: Thu, 06 Feb 2020 10:27:49 GMT
 server: uvicorn
```

3. Check the Status of a Transcription Job

- Click on the transcribe/query to view the API specification to query the status of a transcription job. Use the id of transcription job to obtain the current status of transcription.
- Click on the Try it out button to prepare a sample query.
- Enter the Job id and click on Execute button to submit the sample query.



/query/{job_id} Job status

Get the transcription job status

Parameters

Name	Description
job_id * required string (path)	Job id 3c1c9052-039b-4a90-bf2e-a115b5a074

Responses

Curl

```
curl -X GET "https://ec2-3-215-115-240.compute-1.amazonaws.com:8082/query/3c1c9052-039b-4a90-bf2e-a115b5a07473" -H "accept: application/json"
```

Request URL

```
https://ec2-3-215-115-240.compute-1.amazonaws.com:8082/query/3c1c9052-039b-4a90-bf2e-a115b5a07473
```

Server response

Code: 200

Response body

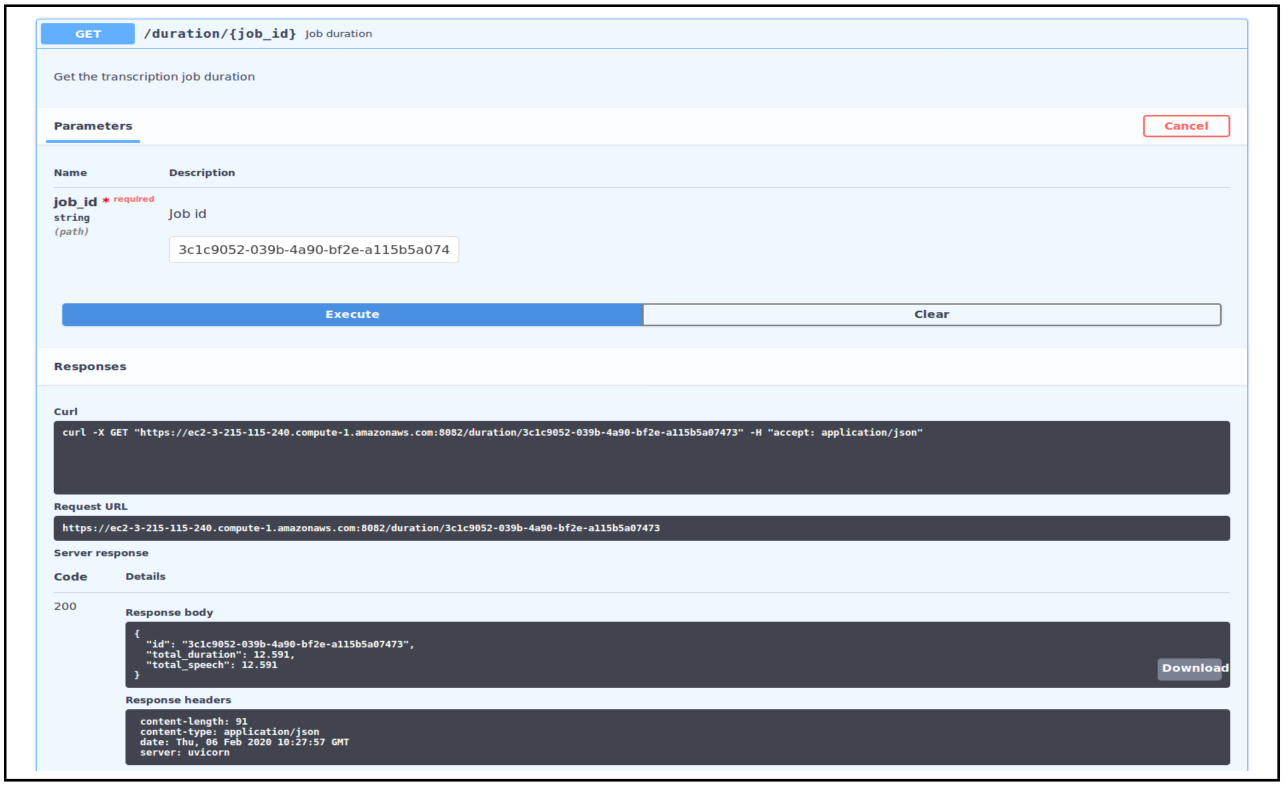
```
{
  "id": "3c1c9052-039b-4a90-bf2e-a115b5a07473",
  "status": "success"
}
```

Response headers

```
 content-length: 64
 content-type: application/json
 date: Thu, 06 Feb 2020 10:41:47 GMT
 server: uvicorn
```

4. Get the duration of audio transcribed for a Transcription Job

   - Click on the transcribe/duration to view the API specification to query the audio duration for a given transcription job. Use the id of transcription job to obtain the duration of audio transcribed.
   - Click on the Try it out button to prepare a sample query.
   - Enter the Job id and click on Execute button to submit the sample query



Get the transcription job duration

job_id: 3c1c9052-039b-4a90-bf2e-a115b5a074

Curl

```
curl -X GET "https://ec2-3-215-115-240.compute-1.amazonaws.com:8082/duration/3c1c9052-039b-4a90-bf2e-a115b5a07473" -H "accept: application/json"
```

Request URL

```
https://ec2-3-215-115-240.compute-1.amazonaws.com:8082/duration/3c1c9052-039b-4a90-bf2e-a115b5a07473
```

Server response

Code: 200

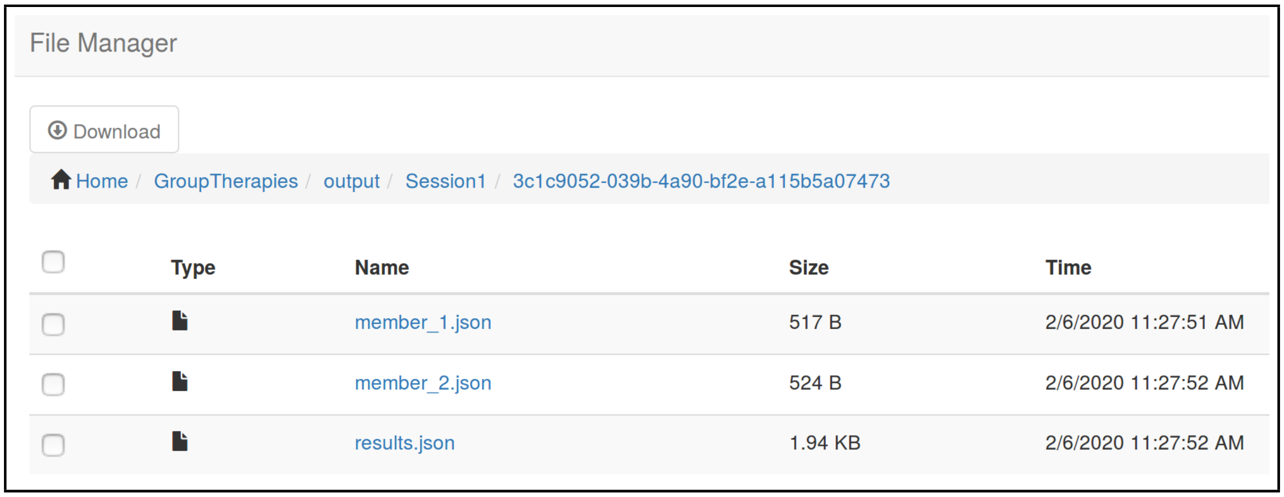
Response body

```
{
  "id": "3c1c9052-039b-4a90-bf2e-a115b5a07473",
  "total_duration": 12.591,
  "total_speech": 12.591
}
```

Response headers

```
content-length: 91
content-type: application/json
date: Thu, 06 Feb 2020 10:27:57 GMT
server: uvicorn
```

5. Downloading the Audio Transcripts with the OpenAPI interface
   - All the Audio Transcripts are stored in the folder specified in the *asset_output* key when the transcription job was submitted.
   - For the above sample transcription job, the audio transcripts can be downloaded using a file browser user interface. Open `https://"PublicDNS(IPv4)ofEC2Instance":8083` in your browser and navigate to the folder *GroupTherapies* → *output* → *Session1* → *JobId*


File Manager
Download
Home / GroupTherapies / output / Session1 / 3c1c9052-039b-4a90-bf2e-a115b5a07473

Type	Name	Size	Time
	member_1.json	517 B	2/6/2020 11:27:51 AM
	member_2.json	524 B	2/6/2020 11:27:52 AM
	results.json	1.94 KB	2/6/2020 11:27:52 AM

3.2 Splitting sentences in audio transcripts

Starting with version 1.1.0 user can control how transcriptions are formatted in the results output.

This is possible after the introduction of a new *process* endpoint optional parameter, called **split_sentences**.

With this parameter it is possible to control the way results are formatted in the output, selecting one the following modalities.

- **split_sentences**:*false* : the output will present a single decoding result per file, whereby sentences are separated by . (space, dot, space)
- **split_sentences**:*true* : the output will present multiple decoding results per file, whereby each decoding result represents a fraction of the decoded sentence in the speech.
- NOTE : all decoding results (sentences) have a trailing dot (.)

The following examples show how to use this parameter and how the output will change accordingly.

a)On *Process* end point, the new optional parameter **split_sentences** is not specified. This option is equivalent to **split_sentences**=*false* (see below) and is allowed by design, to guarantee backwards compatibility with applications developed for

$$TLTS \leq (1.0.x)$$

.

```
{
  "vendor": {
    "name": "TelepathyASR",
    "model": "tl_en_generic",
  },
  "asset": {
    "asset_data": [],
    "asset_bucket": "GroupTherapies",
    "asset_input": "Session1/",
    "asset_output": "output/Session1/",
    "asset_format": {
      "audio_format": "wav",
      "audio_framerate": 16000
    }
  }
}
```

Listing 1: omitting the new parameter **split_sentences** in *process* endpoint

b)On *Process* end point, the new optional parameter is specified and set to **split_sentences**:*false*

```
{
  "vendor": {
    "name": "TelepathyASR",
    "model": "tl_en_generic",
    "split_sentences": false
  },
  "asset": {
    "asset_data": [],
    "asset_bucket": "GroupTherapies",
    "asset_input": "Session1/",
    "asset_output": "output/Session1/",
    "asset_format": {
      "audio_format": "wav",
      "audio_framerate": 16000
    }
  }
}
```

Listing 2: specifying the new parameter **split_sentences**:*false* in *process* endpoint

The output formatting for both cases a) and b) will be the same, and will present a single *chunk* with the corresponding transcribed text. This text may contain several punctuation symbols, showing the points where the transcription engine may have found a pause in the speech signal.

```
"results":[
  {
    ...,
    "final":true,
    "likelihood":3.8957203229,
    "processing_time":1.9699664116,
    "sample_rate":16000,
    "text":"THIS IS THE FIRST TIME THAT A DEVICE CAN FUNCTION AT THE SAME
            VOLTAGE LEVEL AS THE BRAIN . IT IS A CONCEPT BREAKTHROUGH ."
  }
],
"success":true
...
```

Listing 3: transcription output on a single chunk

c)On *Process* end point, the new optional parameter is specified and set to **split_sentences**=*true*

```
{
  "vendor": {
    "name": "TelepathyASR",
    "model": "tl_en_generic",
    "split_sentences": true
  },
  "asset": {
    "asset_data": [],
    "asset_bucket": "GroupTherapies",
    "asset_input": "Session1/",
    "asset_output": "output/Session1/",
    "asset_format": {
      "audio_format": "wav",
      "audio_framerate": 16000
    }
  }
}
```

Listing 4: specifying the new parameter as **split_sentences**:*true* in *process* endpoint

The output formatting for case c) will be different, and will show a number of different *chunks* with the transcribed text. At the end of each chunk text, a single punctuation symbol is added.

```
...
"results":[
  {
    ...,
    "final":true,
    "likelihood":3.8957203229,
    "processing_time":1.5699664116,
    "sample_rate":16000,
    "text":"THIS IS THE FIRST TIME THAT A DEVICE CAN FUNCTION AT THE SAME
            VOLTAGE LEVEL AS THE BRAIN . "
  },
  {
    ...,
    "final":true,
    "likelihood":3.8957203229,
    "processing_time":0.3699664116,
    "sample_rate":16000,
    "text":"IT IS A CONCEPT BREAKTHROUGH ."
  },
],
"success":true
...
```

Listing 5: transcription output in different chunks

Final note on the new parameter **split_sentences** : concatenating the text from different chunks as found in the output shown for case c), must give the same result provided by the output shown for cases a) and b).

3.3 More info on endpoints and capabilties

6. For a more detailed description of the different HTTPS REST endpoints and capabilities offered by TLTS please refer to the online documentation available after installation at: `https://"PublicDNS(IPv4) ofEC2Instance":8082/docs`.

4 Telepathy Labs Transcription Service (TLTS): ASR Channel Requirements, Usage and Configuration

- Requirements
  - Minimum requirement: 1GB RAM per CPU
  - Recommendation: 4 CPU & 8GB RAM
- Utilization
  - By Default, Telepathy Labs ASR handles 2 channels per CPU
    - 1 CPU can handle more channels, but performance issues are to be expected in this case
  - The life cycle of an ASR channel is as follows:
    - acquire channel on POST request
    - process audio data (+response)
    - free channel
- Configuration
  - To specify the number of ASR service workers
    - Explicitly: The number of workers to be created is set using the *GUNICORN_WEB_CONCURRENCY* environment variable.

      ```
      GUNICORN_WEB_CONCURRENCY=1  # instantiate One service worker
      ```

    - Implicitly: We specify the number of workers for each core using the *GUNICORN_WORKERS_PER_CORE* environment variable. The total number of service workers created will be *GUNICORN_WORKERS_PER_CORE * AVAILABLE_CORES*

      ```
      GUNICORN_WORKERS_PER_CORE=1  # default 1, instantiate N service workers,
                                   # where N is number of cores on the machine.
      ```

4.1 Important Features of TLTS

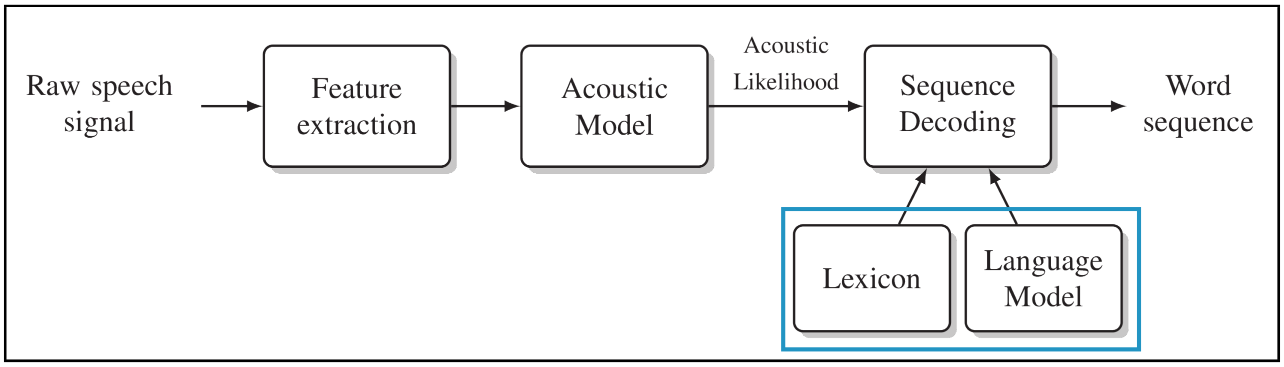
- A Transcription job is performed sequentially to minimize the number of open ASR channels.
- One transcription job **acquires** one ASR channel. The channel is acquired only after the audio data is received. Once the transcription job is complete, the ASR channel is free for other jobs.
- When all the ASR channels are acquired by transcription jobs, a new Transcription job will wait for ASR channels to become free.

5 TLACS: Telepathy Labs ASR Customization Service

TLACS is designed to improve the transcription capability of Telepathy Labs Transcription Service (TLTS). TLACS aims to improve the quality of TLTS, to make it more reliable and provide better outputs which can then be used for any purpose.

5.1 What is an ASR Language Model?

The task of an ASR engine is to convert an input audio signal into an output text. The TLTS ASR engine is composed of several parts that have different purposes, representing different steps between the input speech and the output text.



The TLACS Tool allows the user to work on the “language” side in the sense that it does not modify the acoustic side of the system. This corresponds to the part inside the blue box in the above diagram. The language side includes the language model, which is able to predict sentences which are likely to make sense given some word sequence hypotheses, and a phonetic lexicon, which predicts words which make sense given some phonetic sequence hypotheses.

For example, if the language model receives the following hypotheses:

```
1. i am waiting for you
2. i am waiting four you
3. i am way tinfoil you
```

it should predict that the first hypothesis is more likely than the second, given the sequence of words, as it is closer to spoken language.

5.2 Why create a new Language Model?

Creating a new language model that **represents well** what needs to be recognized by ASR – i.e. a target domain – will help **improving accuracy** on this target domain.

For example, if we want to recognize the names of medicines well, it is recommended to have examples of sentences that contain these names of medicines.

5.3 What type of data will make my Language Model good?

The best data which can be given to the tool are text sentences that are close to the sentences the user wants to recognize in the transcription step. This means that the data should:

- be in the target domain (contain things related to the topic).
- be as close as possible to expected spoken inputs (sentences such as what the audio would contain, not only “word definitions” or more formal text).

6 TLACS: Creating a New Language Model

6.1 Training Data Format

The material to train a new Language Model can be in any one of the following formats:

File Format

- .txt (Recommended)
- .pdf
- .doc
- .docx
- .zip
- .gz
- .html

We recommend to provide plain text (.txt) file or a (.zip) file will multiple text files. The quality of a trained language model while using other file formats will significantly depend on the efficiency of text automatically extracted.

File Size Limits Our recommendation is to provide an input file that does not exceed 500MB in size. Large files will naturally take a longer time to build a language model and also significantly depends on the computing power of the machine hosting the TLACS.

Data Content The content of the data can contain special characters, numbers, and abbreviations which will be normalized by the tool. For optimal results, however, it is best to provide data which are as clean as possible, with no special characters, and with "spoken text", meaning text written as it is spoken. For example "1980" would typically be spoken as "nineteen eighty", and although the normalization should convert from the former form into the latter, this conversion may not always be accurate.

It is also recommended to provide the input with one sentence per line, rather than paragraphs, as the tool will try to split according to punctuation but may not always split sentences perfectly.

Let's say you want to recognize news which talks about *Kaden*. The best data you can give would be previous examples of news about *Kaden*. A good format for input sentences looks like:

```
Kaden, the digital behavioral health company formerly known as Thrivee, announced today its
 commitment to address the opioid addiction crisis in New Hampshire.
The company is exploring a collaborative effort with Catholic Medical Center CMC to provide
 support and resources desperately needed by those struggling with opioid use disorder in the
 state.
Kaden is pleased to be a sponsor of CMC's fifth Annual Summit: Management of the
 Opioid-Dependent Patient, taking place Friday, November fifteen, in Nashua, New Hampshire.
This annual conference highlights the latest strategies and partnerships in the prevention,
 identification, and treatment of opioid-dependent patients, serving as an ideal venue to
 educate providers about Kaden's groundbreaking virtual treatment services.
We're proud to be part of C M C's efforts to provide individuals struggling with opioid
 addiction greater access to support and services, said Dave Henderson, CEO of Kaden.
New Hampshire is the ideal setting to launch this platform, which will give patients real
 hope for recovery.
```

Source: `https://www.kadenhealth.com/kaden-commits-to-address-new-hampshires-opioid-epidemic/`

6.2 Create a Language Model

There are two ways to train a new language model:

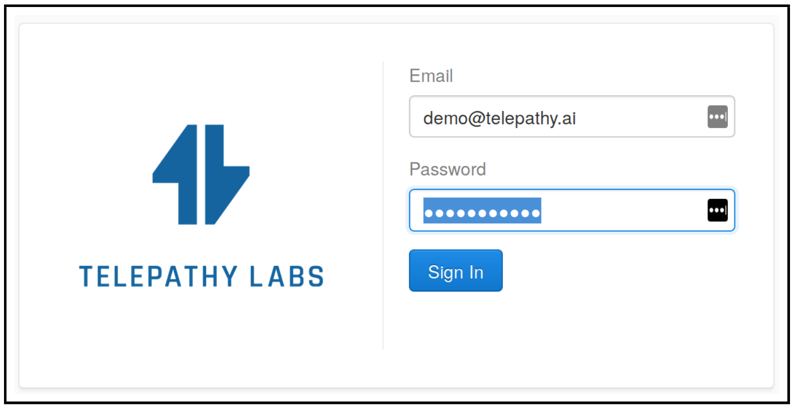
1. A model trained only using the words contained in the input training file (section 6.2.1).
2. Retrain an existing model with words contained in the input training file (section 6.2.2).

6.2.1 Create a Language Model: starting from scratch

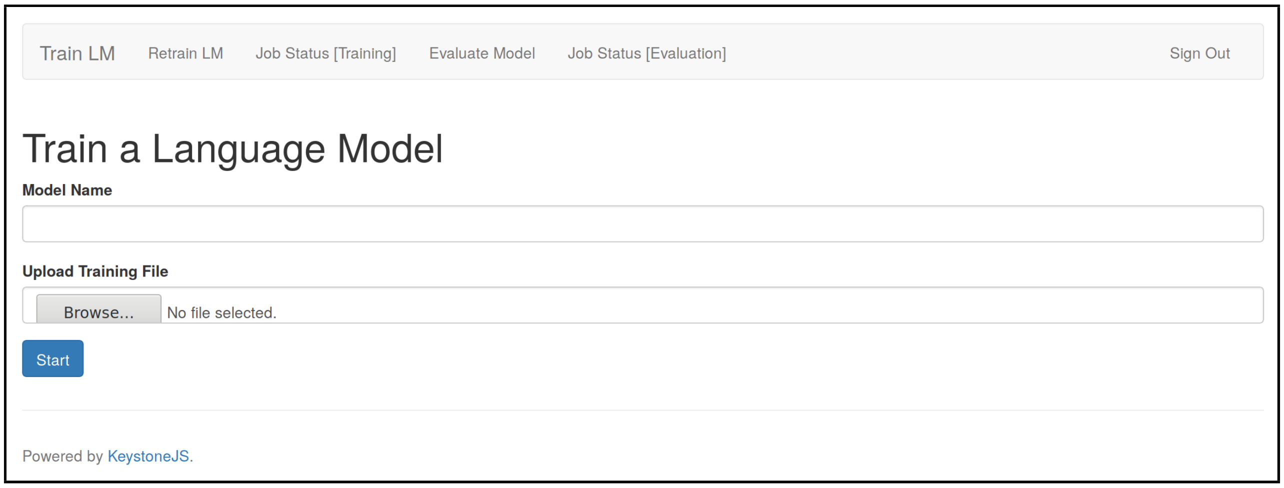
This section focuses on using only the data provided as input to create a new model. This means that the transcription model will be capable of recognizing well the words and sentences given in the input, and some variations. It will not be able to recognize words that are never seen in the input training data.

Step 1 To securely access the UI for TLACS, open `https://"PublicDNS(IPv4)ofEC2Instance":3000` in your browser, when opening it for the first time ensure that you accept the ssl certificate.

Enter your login credentials to gain access to TLACS



After you successfully log in, you will see an interface (shown below) with a menu bar on the top. Click on "**Train LM**" item on the top left menu to start the process of creating a new language model.



Train a Language Model

Model Name

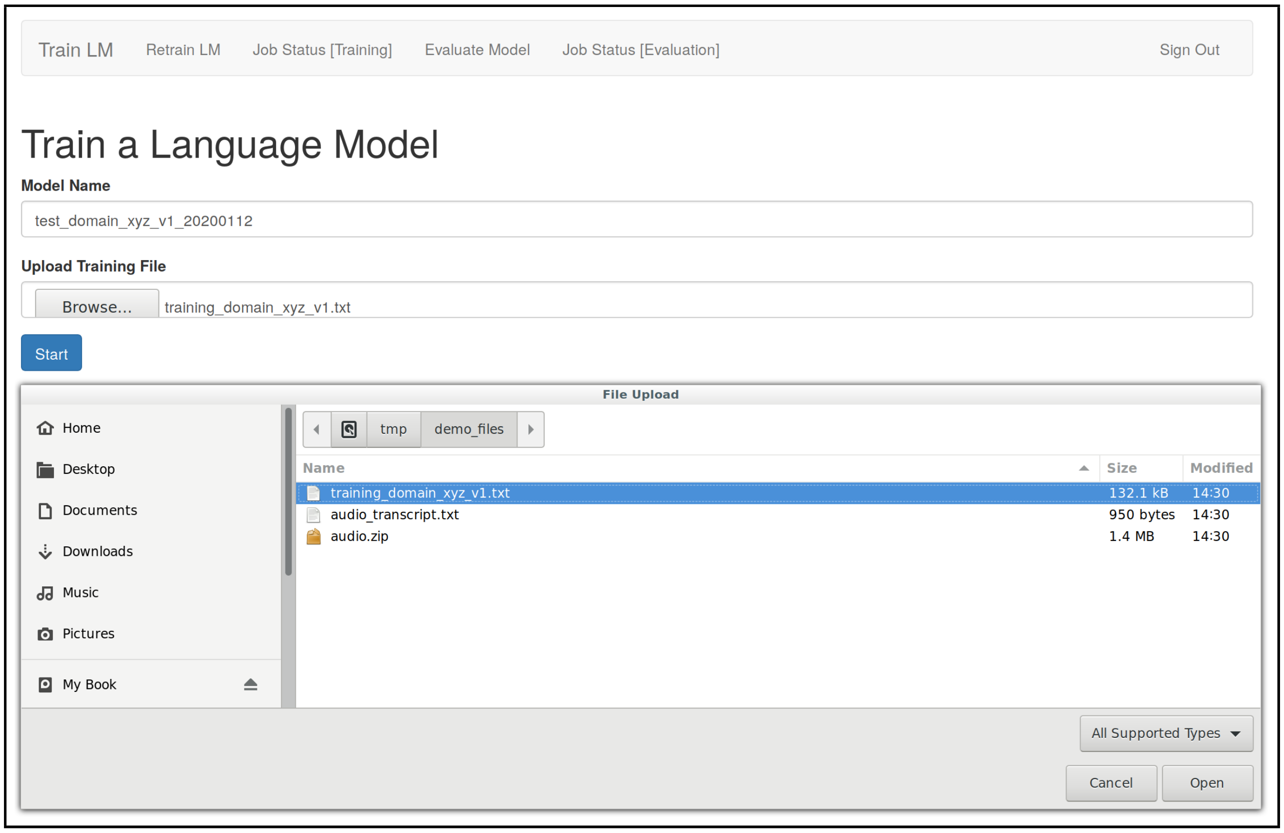
Upload Training File

No file selected.

Powered by KeystoneJS.

Step 2 Specify the name you want to assign to your language model in the "**Model Name**" text box. We recommend using a unique and descriptive name for each new model.

Step 3 Click on the "**Browse**" button to upload your file that contains training material. A sample model name and training file is shown below:



Train a Language Model

Model Name

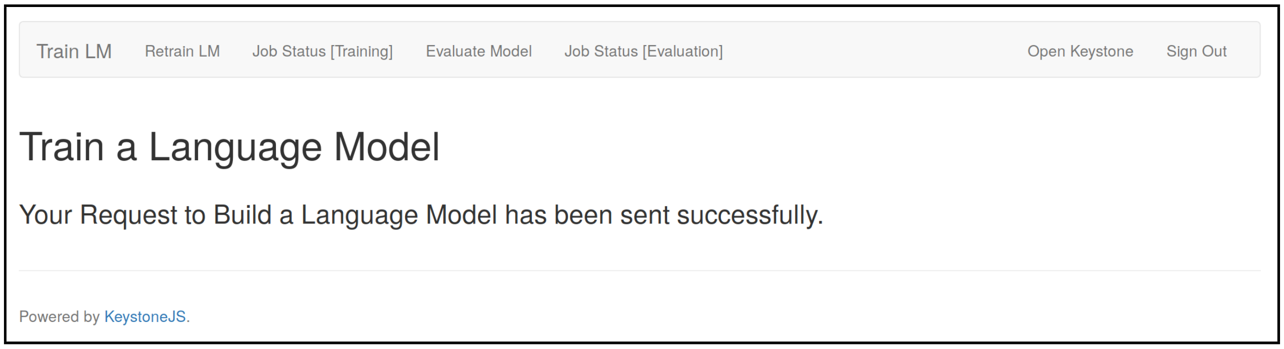
test_domain_xyz_v1_20200112

Upload Training File

Browse... training_domain_xyz_v1.txt

Name	Size	Modified
training_domain_xyz_v1.txt	132.1 kB	14:30
audio_transcript.txt	950 bytes	14:30
audio.zip	1.4 MB	14:30

Step 4 Click on the **Start** button to start training the model. On successful submission of the request you will see the image below:

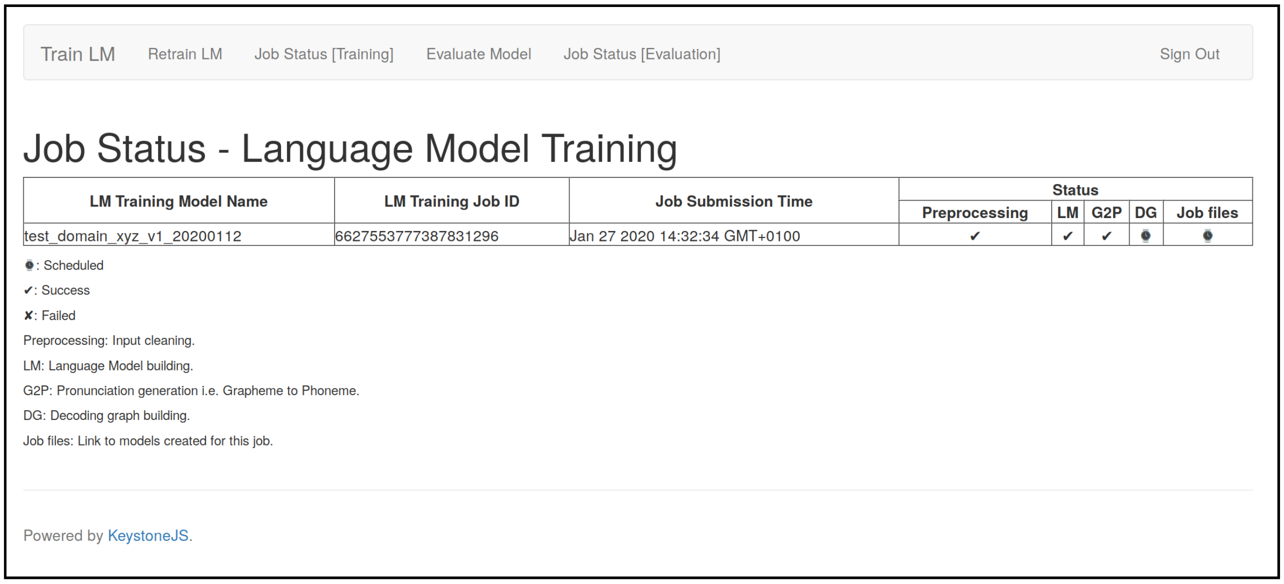


Train a Language Model

Your Request to Build a Language Model has been sent successfully.

Powered by KeystoneJS.

Step 5 To monitor the status of your job click on the tab **Job Status [Training]**. On this interface, you can keep a track of the progress of every step of model training. The interface shows which step of the training is scheduled to execute or completed or has failed.



Train LM | Retrain LM | Job Status [Training] | Evaluate Model | Job Status [Evaluation] | Sign Out

Job Status - Language Model Training

LM Training Model Name	LM Training Job ID	Job Submission Time	Status				
			Preprocessing	LM	G2P	DG	Job files
test_domain_xyz_v1_20200112	6627553777387831296	Jan 27 2020 14:32:34 GMT+0100	✓	✓	✓	●	●

●: Scheduled

✓: Success

✗: Failed

Preprocessing: Input cleaning.

LM: Language Model building.

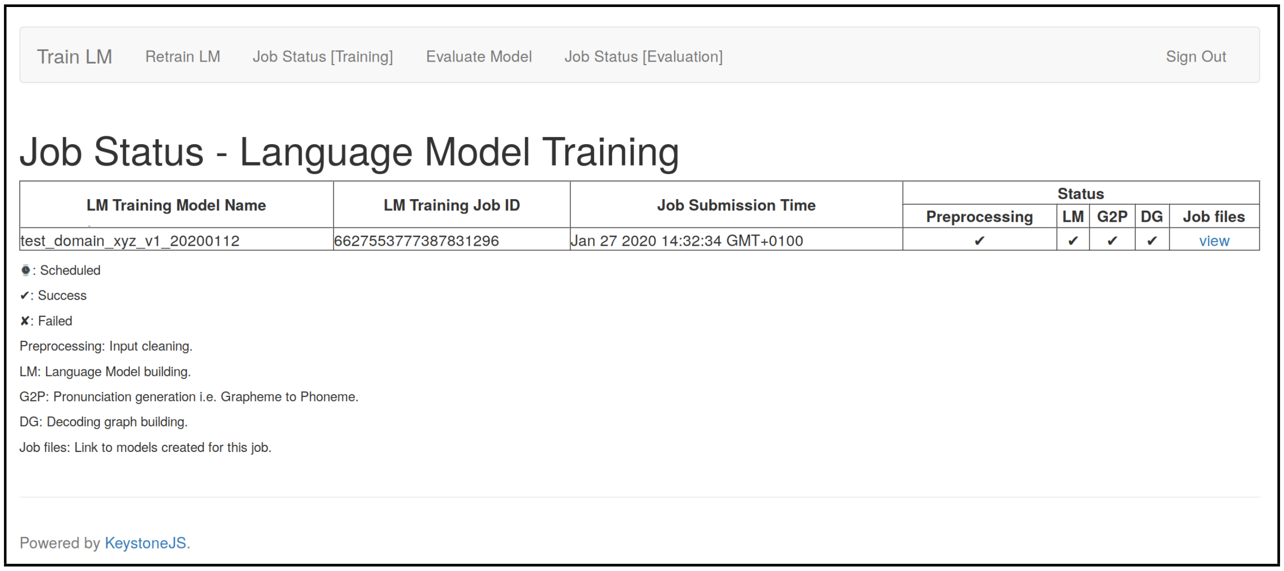
G2P: Pronunciation generation i.e. Grapheme to Phoneme.

DG: Decoding graph building.

Job files: Link to models created for this job.

Powered by KeystoneJS.

Once your model has been trained successfully, you should have a link (“view”) to the output files under the “Job files”.



Train LM | Retrain LM | Job Status [Training] | Evaluate Model | Job Status [Evaluation] | Sign Out

Job Status - Language Model Training

LM Training Model Name	LM Training Job ID	Job Submission Time	Status				
			Preprocessing	LM	G2P	DG	Job files
test_domain_xyz_v1_20200112	6627553777387831296	Jan 27 2020 14:32:34 GMT+0100	✓	✓	✓	✓	view

●: Scheduled

✓: Success

✗: Failed

Preprocessing: Input cleaning.

LM: Language Model building.

G2P: Pronunciation generation i.e. Grapheme to Phoneme.

DG: Decoding graph building.

Job files: Link to models created for this job.

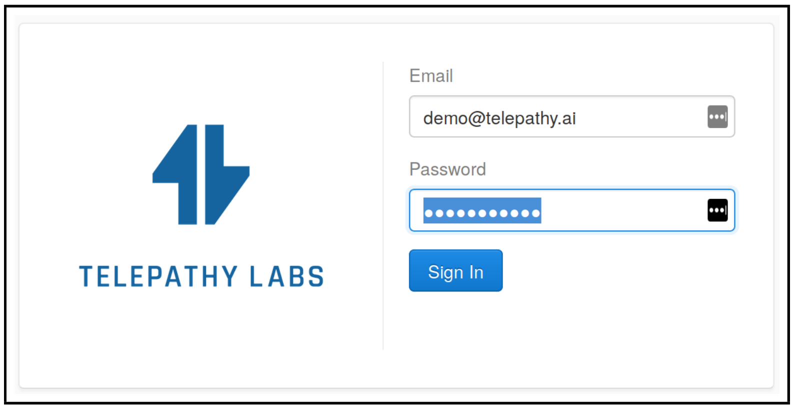
Powered by KeystoneJS.

6.2.2 Create a Language Model: use an existing model

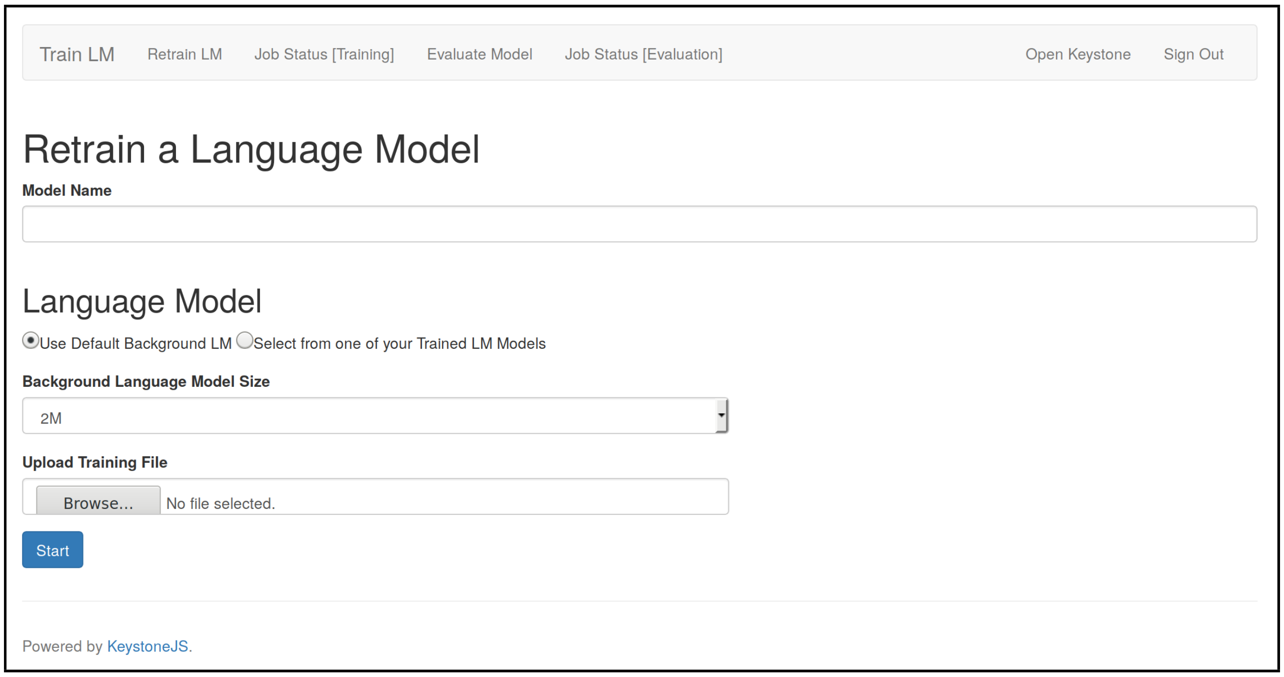
This section focuses on retraining an existing model with new input data. Apart from sentences that an existing model was trained to recognize, the retrained language model will be capable of recognizing well the words and sentences given in the input along with some variations.

Step 1 To securely access the UI for TLACS, open `https://"PublicDNS(IPv4)ofEC2Instance":3000` in your browser, when opening it for the first time ensure that you accept the ssl certificate.

Enter your login credentials to gain access to TLACS



After you successfully log in, click on the "**Retrain LM**" item on the top-left menu to start the process of retraining an existing language model.



Retrain a Language Model

Model Name

Language Model

● Use Default Background LM ○ Select from one of your Trained LM Models

Background Language Model Size

2M

Upload Training File

Browse... No file selected.

Start

Powered by KeystoneJS.

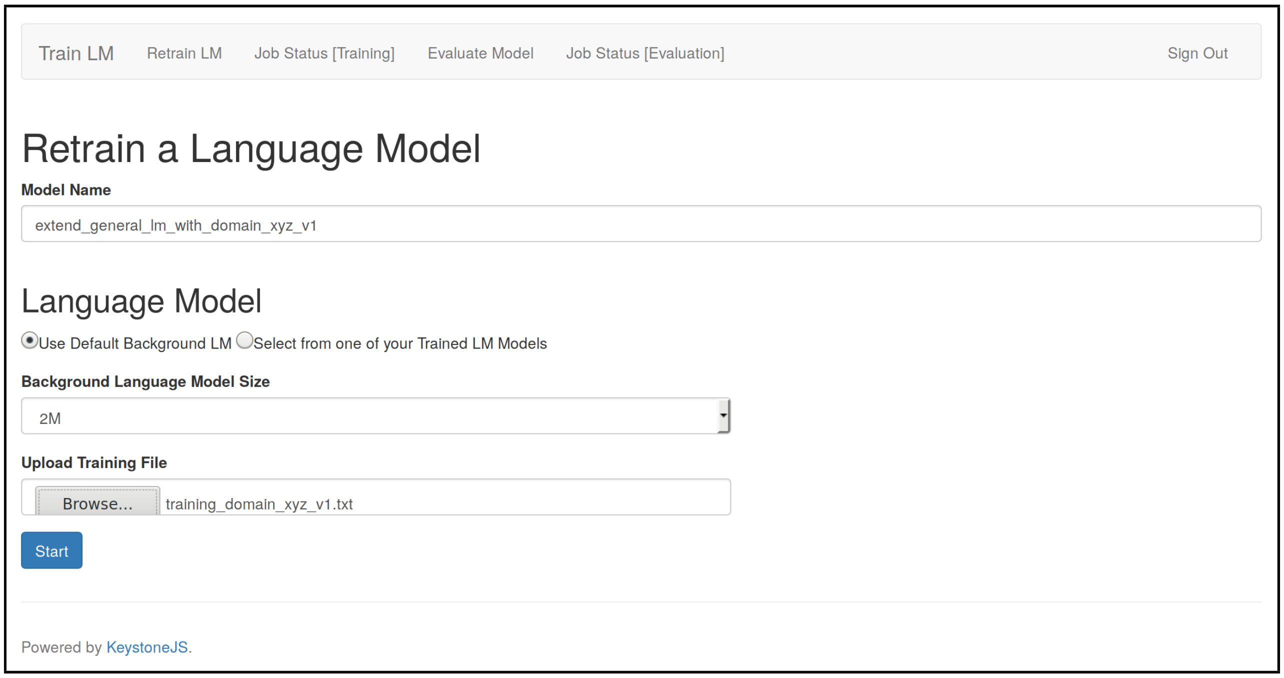
Step 2 Specify the name you want to assign to your retrained language model in the "**Model Name**" text box. We recommend using a unique and descriptive name for each new model.

There are two options to retrain a language model

a Model retrained using a Default Background LM provided by Telepathy Labs.

b Model retrained using one of your trained language models.

Step 2a: Use Default Background LM In this option, you choose to use a general LM as background model and improve it by adding new data. Click on "**Use Default Background LM**" and select one of the different models based on the size in the list: the bigger the model, the higher the ASR performance and the training time. We recommend using the **2M** model. Click on the "**Browse**" button to upload your file that contains training material. A sample background language model and training file is shown below:



Retrain a Language Model

Model Name: extend_general_lm_with_domain_xyz_v1

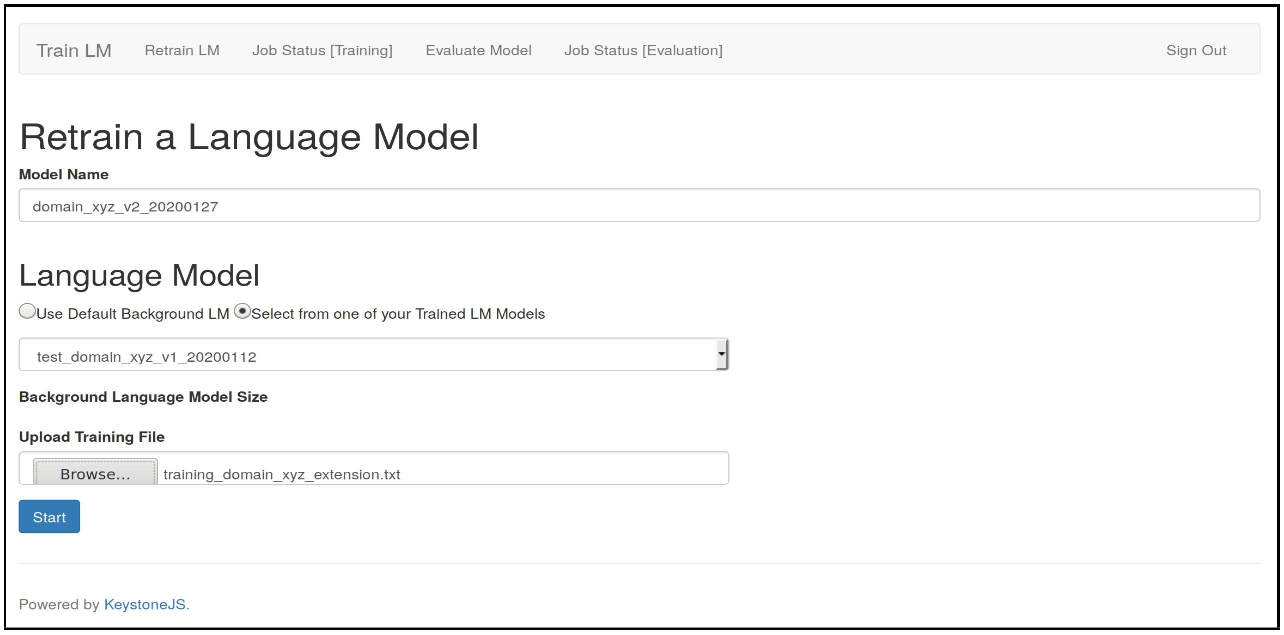
Language Model

[✓ Use Default Background LM] Select from one of your Trained LM Models

Background Language Model Size: 2M

Upload Training File: Browse... training_domain_xyz_v1.txt

Step 2b: Use one of your LM Builds In this option, you choose to retrain a previously trained model and improve it by adding new data. Click on "**Select from one of your LM Builds**" and choose one of your previously trained models from the list of model names. Click on the "**Browse**" button to upload your file that contains training material. A sample Tranined LM model and training file is shown below:



Retrain a Language Model

Model Name: domain_xyz_v2_20200127

Language Model

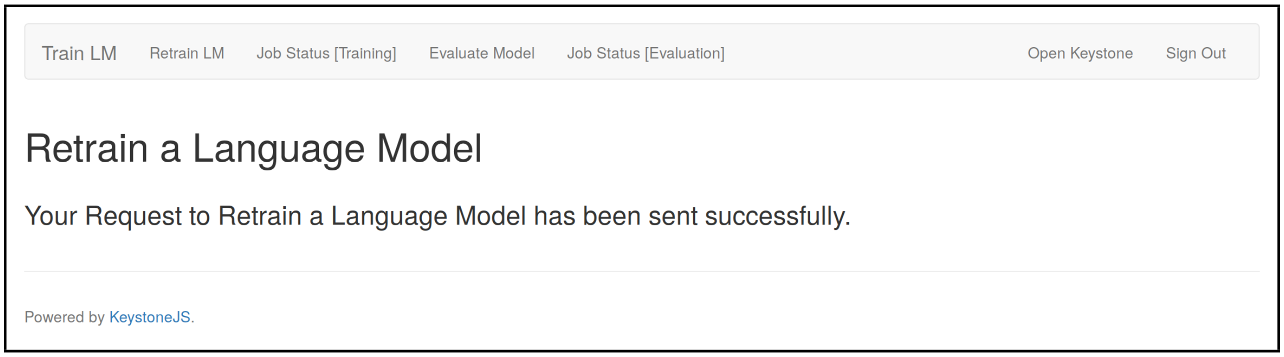
Use Default Background LM [✓ Select from one of your Trained LM Models]

test_domain_xyz_v1_20200112

Background Language Model Size

Upload Training File: Browse... training_domain_xyz_extension.txt

Step 3 Click on the **Start** button to start retraining the model. On successfull submission of the request, you will see the image below:

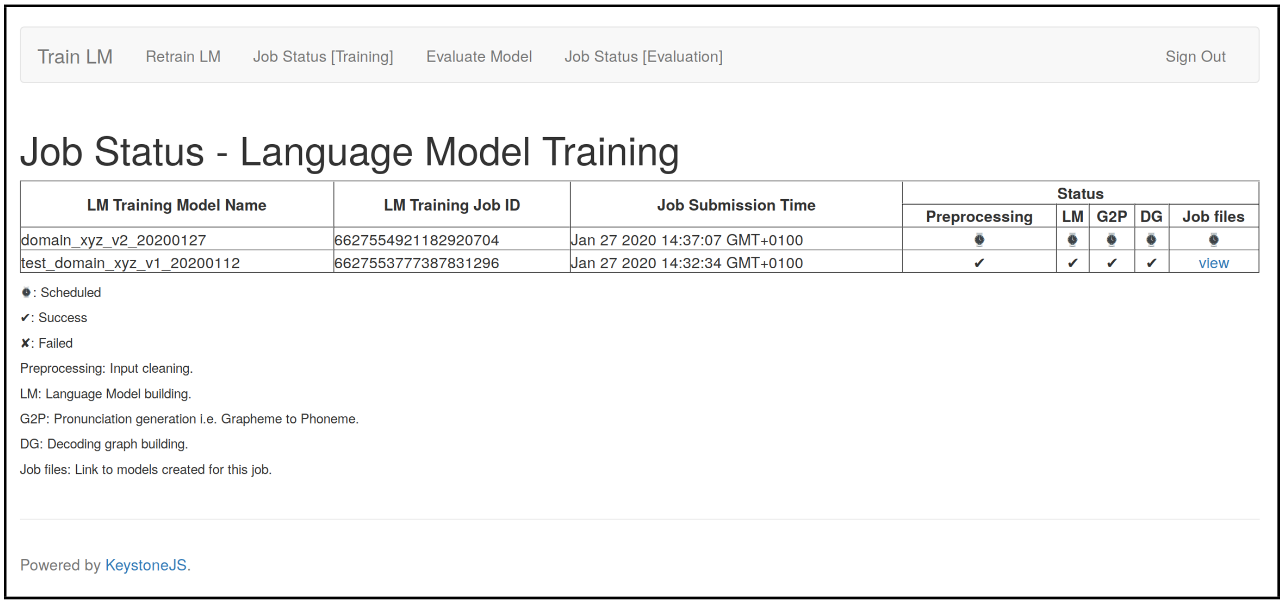


Retrain a Language Model

Your Request to Retrain a Language Model has been sent successfully.

Powered by KeystoneJS.

Step 4 To monitor the status of your job click on the tab **Job Status [Training]**. On this interface, you can keep a track of the progress of every step of model training. The interface shows which step of the training is scheduled to execute or successfully completed or has failed.



Job Status - Language Model Training

LM Training Model Name	LM Training Job ID	Job Submission Time	Status				
			Preprocessing	LM	G2P	DG	Job files
domain_xyz_v2_20200127	6627554921182920704	Jan 27 2020 14:37:07 GMT+0100	●	●	●	●	●
test_domain_xyz_v1_20200112	6627553777387831296	Jan 27 2020 14:32:34 GMT+0100	✓	✓	✓	✓	view

●: Scheduled

✓: Success

✗: Failed

Preprocessing: Input cleaning.

LM: Language Model building.

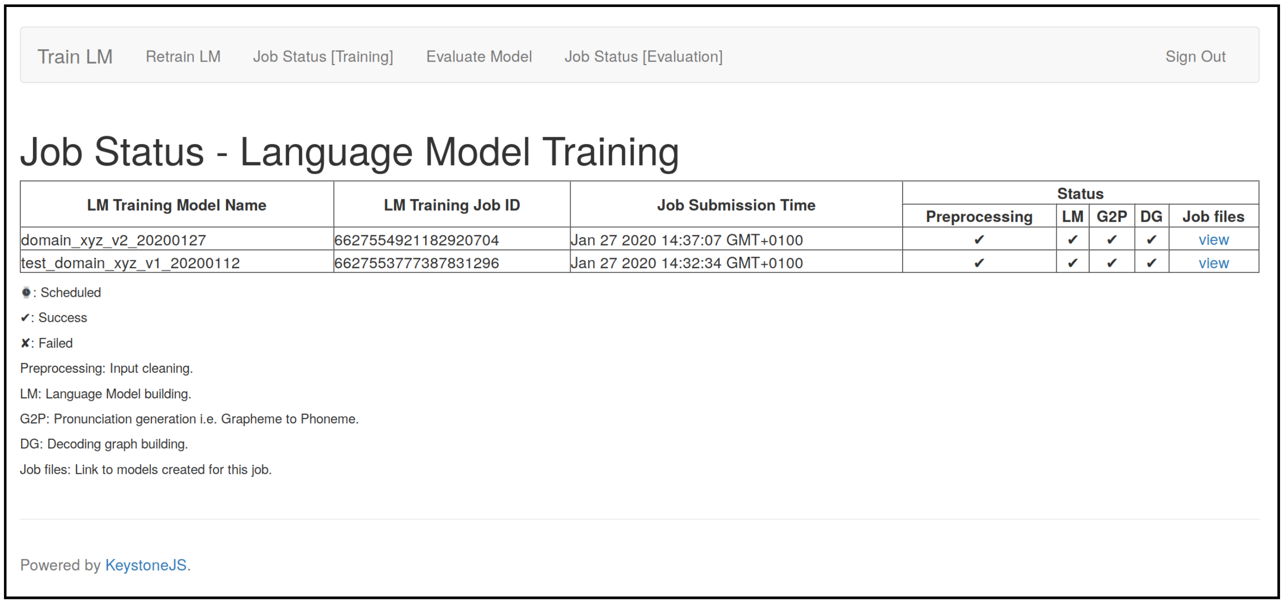
G2P: Pronunciation generation i.e. Grapheme to Phoneme.

DG: Decoding graph building.

Job files: Link to models created for this job.

Powered by KeystoneJS.

Once your model has been trained successfully, you should have a link ("view") to the output files under the "Job files".



Job Status - Language Model Training

LM Training Model Name	LM Training Job ID	Job Submission Time	Status				
			Preprocessing	LM	G2P	DG	Job files
domain_xyz_v2_20200127	6627554921182920704	Jan 27 2020 14:37:07 GMT+0100	✓	✓	✓	✓	view
test_domain_xyz_v1_20200112	6627553777387831296	Jan 27 2020 14:32:34 GMT+0100	✓	✓	✓	✓	view

●: Scheduled

✓: Success

✘: Failed

Preprocessing: Input cleaning.

LM: Language Model building.

G2P: Pronunciation generation i.e. Grapheme to Phoneme.

DG: Decoding graph building.

Job files: Link to models created for this job.

Powered by KeystoneJS.

6.3 Evaluating a Model

6.3.1 Why evaluate a model?

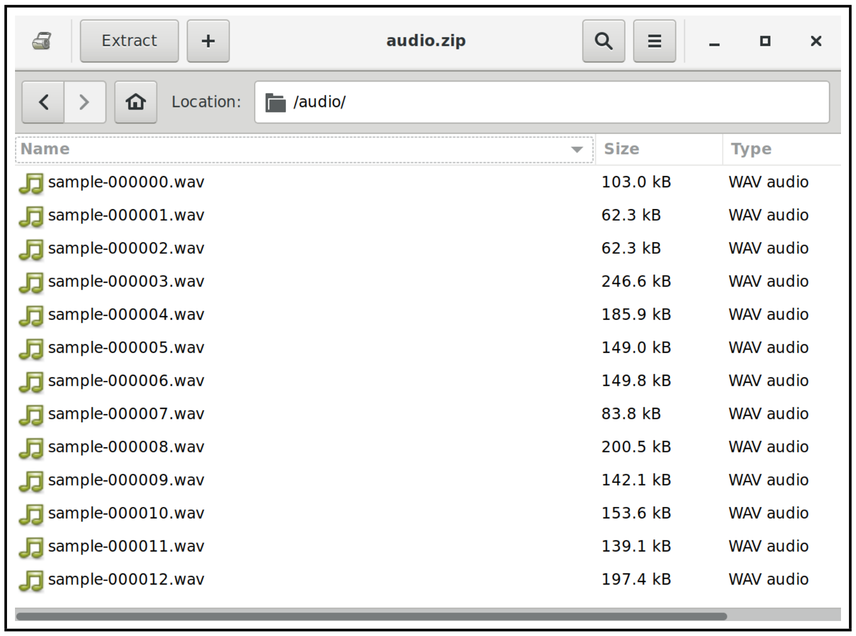
The goal of training a new language model is to improve the accuracy of the transcription service on a given domain. To verify that the accuracy indeed improves, you need to measure if the new model performs better than previous ones. To evaluate a model, the user must prepare a test set that consists of a pair of audio files and transcriptions. For each test audio file, there must be an associated correct (or ground truth) transcription, to evaluate if the ASR (using the trained language model) can decode the audio correctly.

6.3.2 Input data format

There are two inputs to evaluate a trained language model:

- the audio files, in **.wav format**, in a compressed **.zip archive**. The filename for each audio file should not contain spaces and avoid special characters. The recommendation is to have only standard characters (a-z, A-Z), numbers, dashes, and underscores.
- Corresponding ground truth transcription, in **.txt format** with each line containing the exact name of the corresponding audio file without its path, followed by a tab, followed by the transcription in lower case, with no punctuation.

A sample set of audio files stored in **audio.zip** and their transcripts stored in the **audio_transcript.txt** file is shown below. The filenames of audio files and their transcripts must be an exact match, otherwise, sentences will not be evaluated.



Name	Size	Type
sample-000000.wav	103.0 kB	WAV audio
sample-000001.wav	62.3 kB	WAV audio
sample-000002.wav	62.3 kB	WAV audio
sample-000003.wav	246.6 kB	WAV audio
sample-000004.wav	185.9 kB	WAV audio
sample-000005.wav	149.0 kB	WAV audio
sample-000006.wav	149.8 kB	WAV audio
sample-000007.wav	83.8 kB	WAV audio
sample-000008.wav	200.5 kB	WAV audio
sample-000009.wav	142.1 kB	WAV audio
sample-000010.wav	153.6 kB	WAV audio
sample-000011.wav	139.1 kB	WAV audio
sample-000012.wav	197.4 kB	WAV audio

audio_transcript.txt

```
sample-000000.wav    without the dataset the article is useless
sample-000001.wav    i've got to go to him
sample-000002.wav    and you know it
sample-000003.wav    down below in the darkness were hundreds of people sleeping in peace
sample-000004.wav    hold your nose to keep the smell from disabling your motor functions
sample-000005.wav    down below in the darkness were hundreds of people sleeping in peace
sample-000006.wav    strange images passed through my mind
sample-000007.wav    the sheep had taught him that
sample-000008.wav    this was the strangest of all things that ever came to earth
from outer space
sample-000009.wav    it was glaringly hot not a cloud in the sky nor a breath of wind
sample-000010.wav    your son went to serve at a distant place and became a centurion
sample-000011.wav    they made the boy continue digging but he found nothing
sample-000012.wav    one can imagine these two covered with sand running up the little
street
```

6.3.3 Evaluation Metric

How is accuracy measured? The typical measure of accuracy for an ASR system is called Word Error Rate (WER[1]).

The WER is defined as follows:

$$WER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C} \tag{1}$$

where:

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N=S+D+C)

As WER measures error, the **lower the WER value** the **better the model** on the test data. This means, if your WER goes down when you evaluate on a new model, it means your new model has improved over the specific evaluation data.

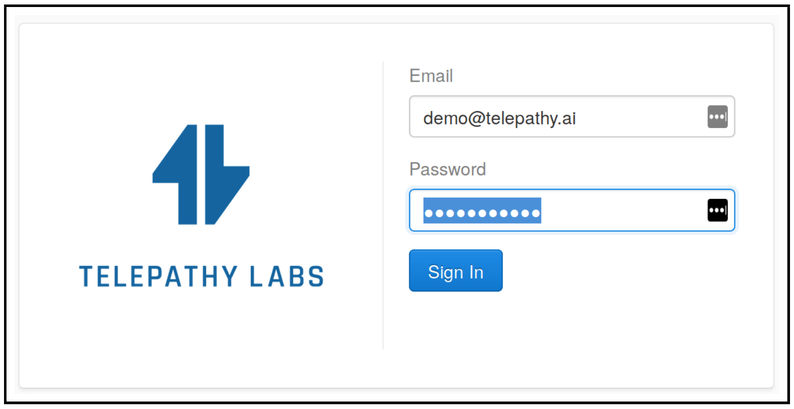
What is the evaluation doing? The model evaluation will run the ASR on the test audio files, using the model to be evaluated, and then calculate the accuracy of the model given the transcription. It also does some sanity checks to make sure that the model will run smoothly in the transcription service.

[1]For more details, refer to `https://en.wikipedia.org/wiki/Word_error_rate`

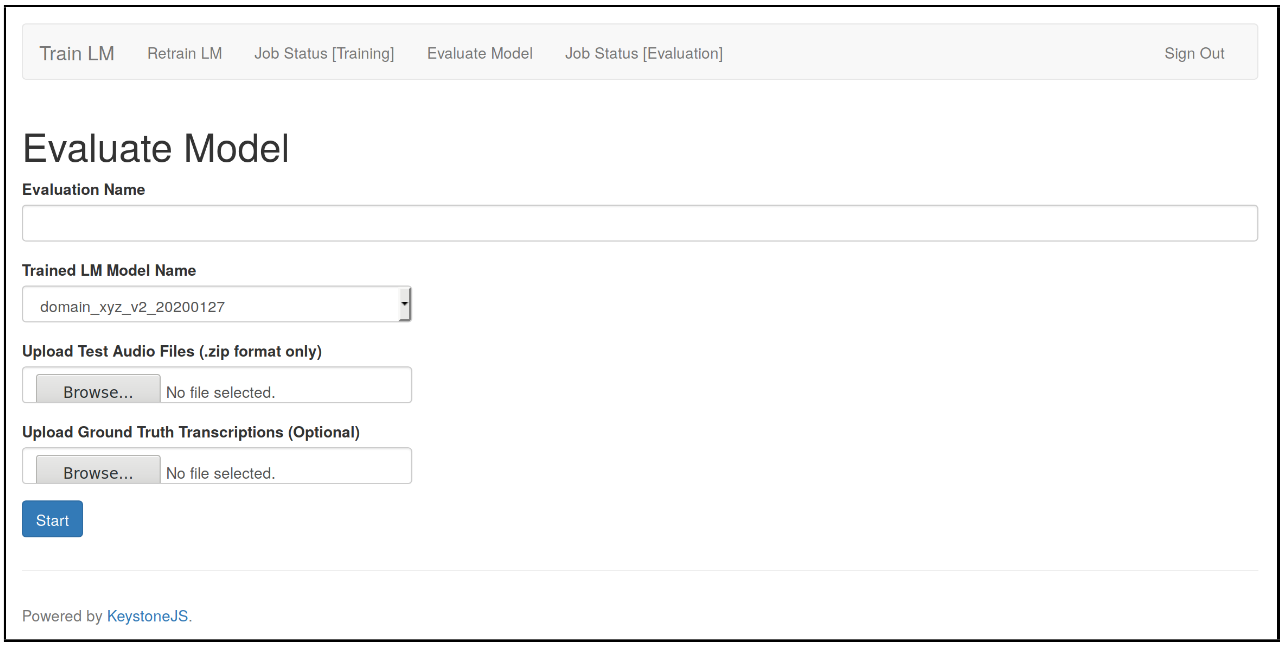
6.3.4 Evaluate a Language Model

Step 1 To securely access the UI for TLACS, open `https://"PublicDNS(IPv4)ofEC2Instance":3000` in your browser, when opening it for the first time ensure that you accept the ssl certificate.

Enter your login credentials to gain access to TLACS



After you successfully log in, click on the "**Evaluate Model**" item on the top-right menu to start the process of evaluating the language model.



Evaluate Model

Evaluation Name

Trained LM Model Name
domain_xyz_v2_20200127

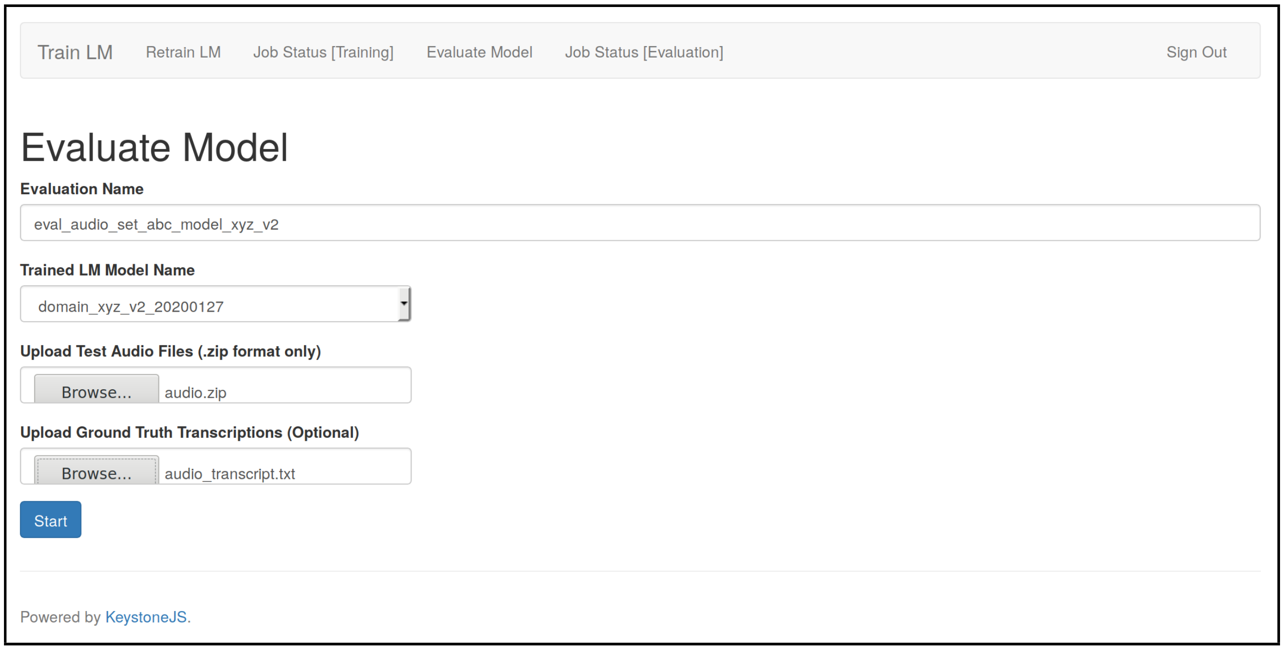
Upload Test Audio Files (.zip format only)
No file selected.

Upload Ground Truth Transcriptions (Optional)
No file selected.

Powered by KeystoneJS.

Step 2 Specify the name you want to assign to your language model evaluation job in the "**Evaluation Name**" text box. We recommend using a unique and descriptive name for each new evaluation.

Step 3 Choose the model you want to evaluate in the "**Trained LM Model Name**" list. A sample evaluation name and trained language model name is shown below:



Evaluate Model

Evaluation Name: eval_audio_set_abc_model_xyz_v2

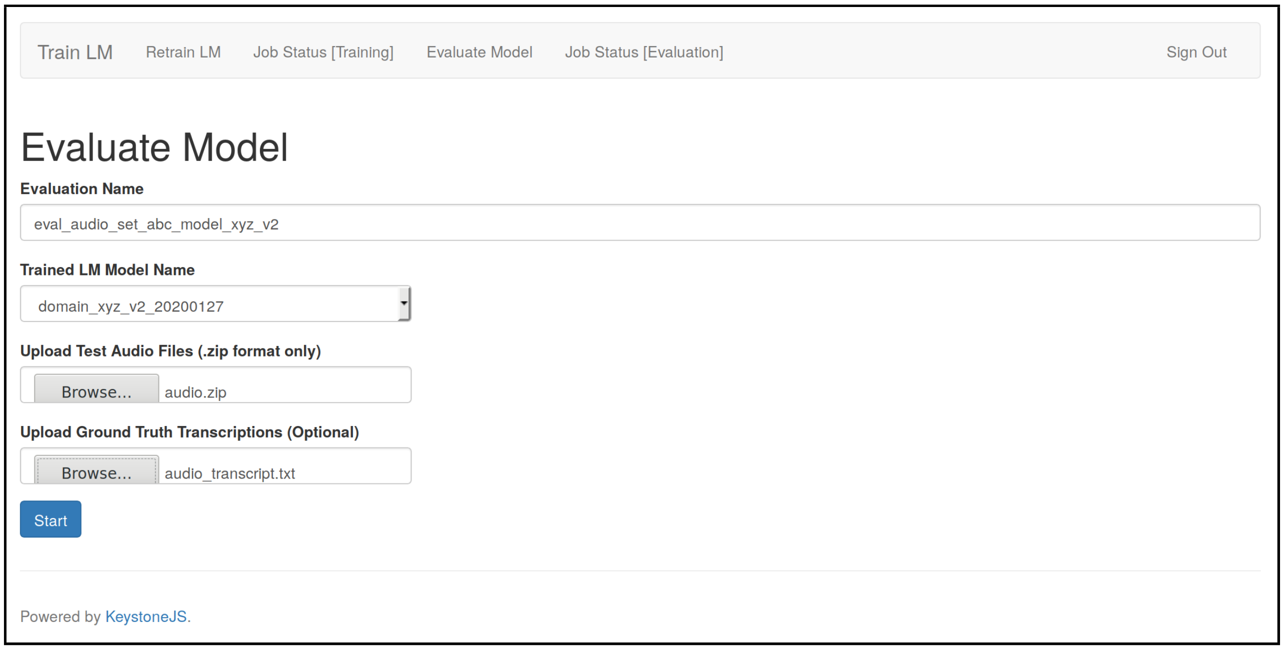
Trained LM Model Name: domain_xyz_v2_20200127

Upload Test Audio Files (.zip format only): audio.zip

Upload Ground Truth Transcriptions (Optional): audio_transcript.txt

Step 4 Provide two inputs to the tool as explained in section 6.3.2: one **.zip** archive containing **.wav** files, and one **.txt** file containing the transcriptions of the test audio files. Click on the "**Upload Test Audio**" button to upload zipped audio files and click on the "**Upload Ground Truth Transcription (Optional)**" to upload the transcriptions file.

Note that if you do not provide the Ground Truth Transcription, the tool will still perform ASR using the selected trained language model, and run the sanity check, but it will not compute the WER.



Evaluate Model

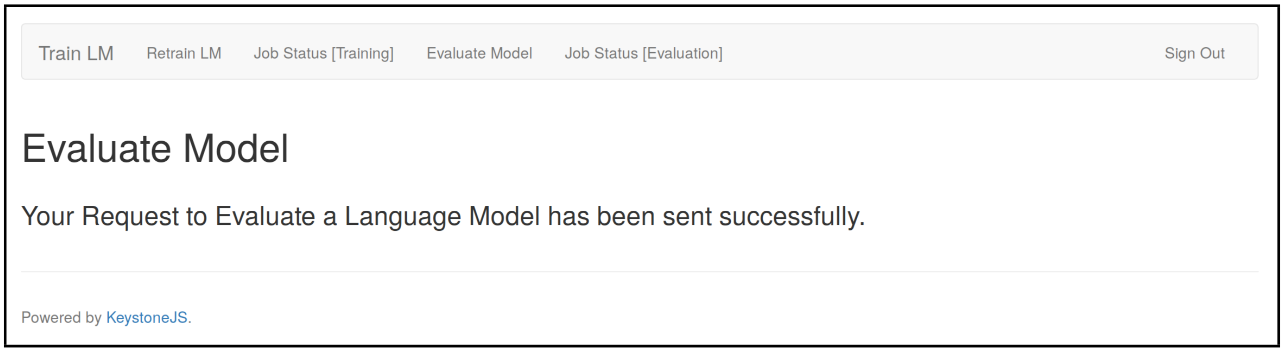
Evaluation Name: eval_audio_set_abc_model_xyz_v2

Trained LM Model Name: domain_xyz_v2_20200127

Upload Test Audio Files (.zip format only): audio.zip

Upload Ground Truth Transcriptions (Optional): audio_transcript.txt

Step 5 Click on the **Start** button to start evaluating the trained model. On the successful submission of the request, you will see the message below:

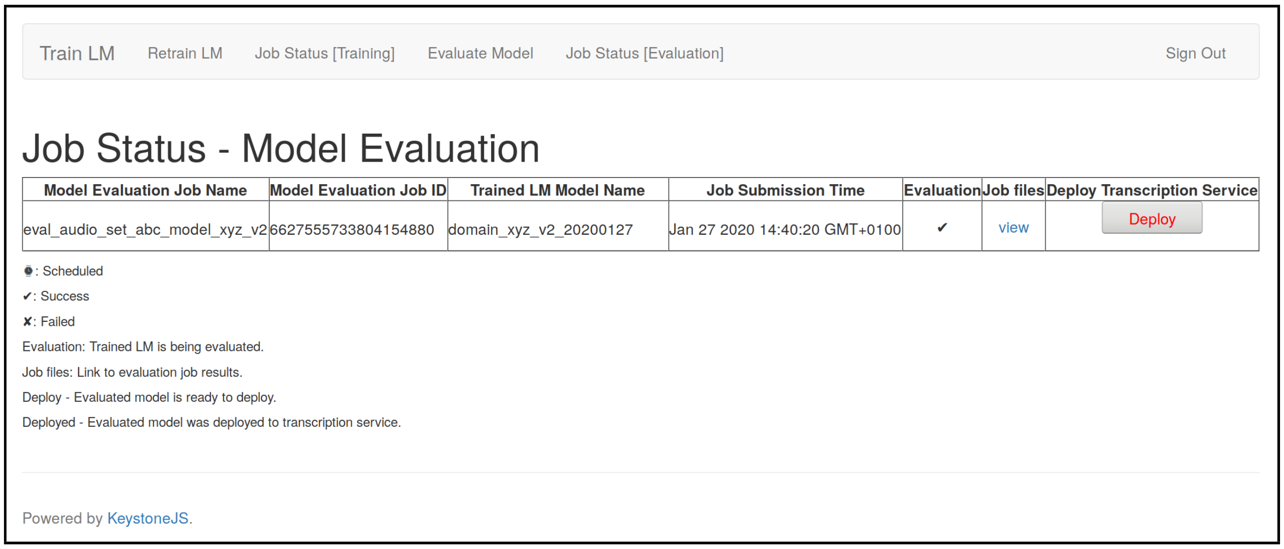


Evaluate Model

Your Request to Evaluate a Language Model has been sent successfully.

Powered by KeystoneJS.

Step 6 To monitor the status of your evaluation job click on the tab **Job Status** [**Evaluation**]. On this interface, you can keep track of the model evaluation progress. If the evaluation is successful, you should have a link ("view") to the output files under the "Job files". Further, you will have the option to deploy the evaluated language model to the production instance of TLTS deployed at `https://"PublicDNS(IPv4)ofEC2Instance":8082`. For further information related to the model deployment, refer to section 6.5.



Job Status - Model Evaluation

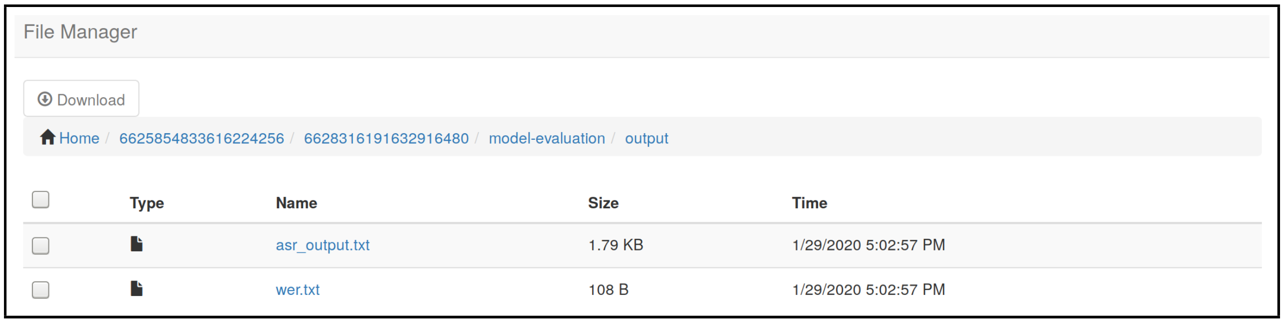
Model Evaluation Job Name	Model Evaluation Job ID	Trained LM Model Name	Job Submission Time	Evaluation	Job files	Deploy Transcription Service
eval_audio_set_abc_model_xyz_v2	6627555733804154880	domain_xyz_v2_20200127	Jan 27 2020 14:40:20 GMT+0100	✔	view	Deploy

●: Scheduled
✔: Success
✘: Failed
Evaluation: Trained LM is being evaluated.
Job files: Link to evaluation job results.
Deploy - Evaluated model is ready to deploy.
Deployed - Evaluated model was deployed to transcription service.

Powered by KeystoneJS.

6.3.5 Interpretation of Evaluation Output

To access the files generated for an evaluation job, Click on "**Job Status [Evaluation]**" tab, click on "view" to access the file browser user interface and navigate to the folder *model-evaluation* → *output*


File Manager

Download

Home / 6625854833616224256 / 6628316191632916480 / model-evaluation / output

	Type	Name	Size	Time
		asr_output.txt	1.79 KB	1/29/2020 5:02:57 PM
		wer.txt	108 B	1/29/2020 5:02:57 PM

The folder contains two files:

- **asr_output.txt** which contains the output of ASR using this model
- **wer.txt** which contains the WER

The *asr_output.txt* should look similar to your input transcription file (i.e., *audio_transcript.txt*), with possible errors from ASR. The sample contents of *asr_output.txt* are shown below:

asr_output.txt

```
sample-000000.wav without the data set the article is useless
sample-000001.wav i've got to go to him
sample-000002.wav and you know it
sample-000003.wav down below and the darkness were hundreds of people sleeping in peace
sample-000004.wav hold your nose to keep the smell from disabling your motor functions
sample-000005.wav down below in the darkness were hundreds of people sleeping in peace
sample-000006.wav strange images passed through my mind
sample-000007.wav the sheep had taught him that
sample-000008.wav this was the strangest of all things that ever came to earth from outer
space
sample-000009.wav it was glaringly hot not a cloud in the sky nor a breath of wind
sample-000010.wav your son went to serve at a distant place and became a centurion
sample-000011.wav they made the boy continue digging but he found nothing
sample-000012.wav one can imagine these two covered with sand running up the little
street in the bright sunlight
```

The sample contents of the WER file (i.e., *wer.txt*) are shown below. SER stands for Sentence Error Rate.

wer.txt

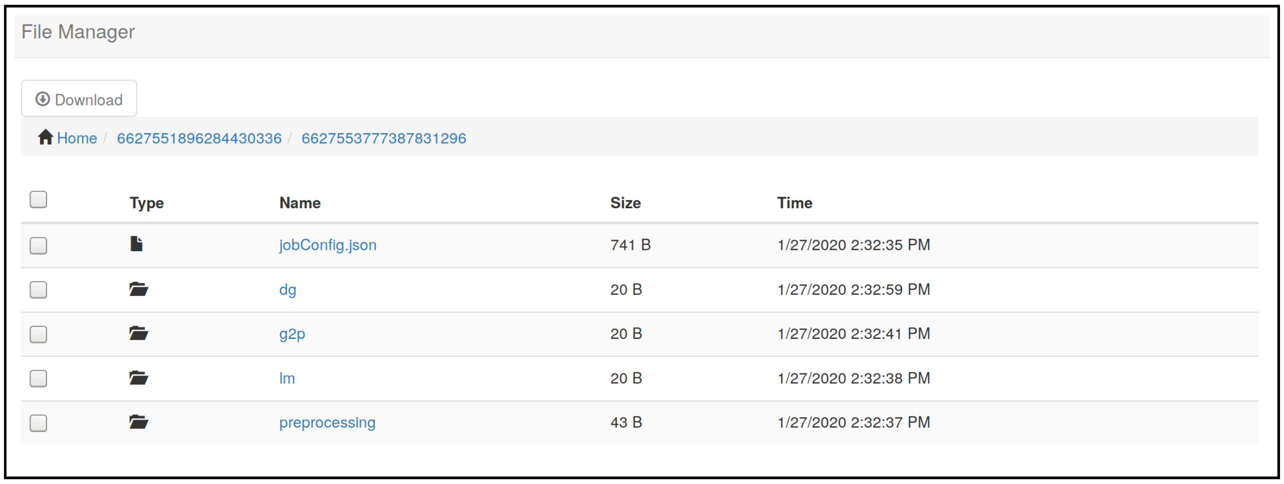
```
%WER 2.22 [ 3 / 135, 1 ins, 0 del, 2 sub ]
%SER 15.38 [ 2 / 13 ]
Scored 13 sentences, 0 not present in hyp.
```

Based on the above sample WER file, we evaluate that you have about 2 words wrong out of a hundred, with details about substitutions, insertions, and deletions. The SER means that overall 15% of the sentences were not 100% accurate.

Typically, a WER below 10% is considered as good or really good, depending on the domain on which it is tested (noise conditions and environment, but also the difficulty of the task given the topic).

6.4 Download a Model

Once your model has been trained successfully, you should have a link ("view") in the "**Job Status [Training]**" tab to access the output files using a file browser user interface. A sample output of a training job is shown below



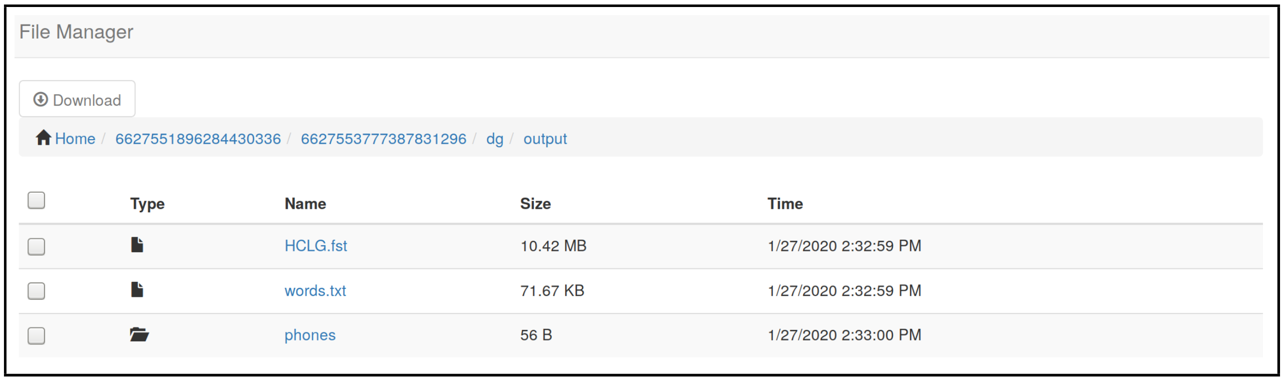
File Manager

Download

Home / 6627551896284430336 / 6627553777387831296

	Type	Name	Size	Time
☐		jobConfig.json	741 B	1/27/2020 2:32:35 PM
☐		dg	20 B	1/27/2020 2:32:59 PM
☐		g2p	20 B	1/27/2020 2:32:41 PM
☐		lm	20 B	1/27/2020 2:32:38 PM
☐		preprocessing	43 B	1/27/2020 2:32:37 PM

Here you can find the output files of each subcomponent. The most relevant one to perform ASR Transcription using a trained language model is the output of the *DG* component, which is responsible to generate the final transcribed output.



File Manager

Download

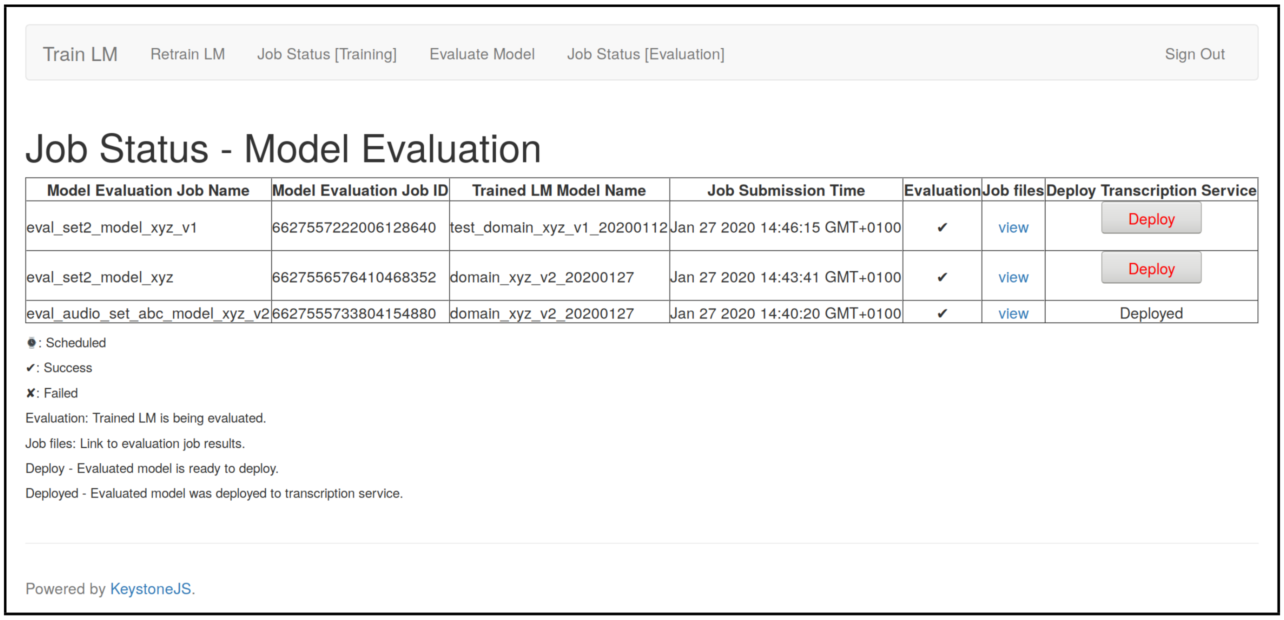
Home / 6627551896284430336 / 6627553777387831296 / dg / output

	Type	Name	Size	Time
☐		HCLG.fst	10.42 MB	1/27/2020 2:32:59 PM
☐		words.txt	71.67 KB	1/27/2020 2:32:59 PM
☐		phones	56 B	1/27/2020 2:33:00 PM

6.5 Deploy a model

As the goal of the TLACS Tool is to improve the transcription accuracy on a given domain, you will want to update your transcription service to use a new model *when you believe that you have trained a **better model** than the one currently in use.*

When you run your evaluation you can look at the given results. If they are better than the ones obtained with the model you are using, you will have the option to deploy this new model to your transcription service. To deploy a model to the transcription service, click on the "**Job Status [Evaluation]**" tab on the top right, decide upon the evaluated model you want to deploy and simply press the "**Deploy**" button.



Job Status - Model Evaluation

Model Evaluation Job Name	Model Evaluation Job ID	Trained LM Model Name	Job Submission Time	Evaluation	Job files	Deploy Transcription Service
eval_set2_model_xyz_v1	6627557222006128640	test_domain_xyz_v1_20200112	Jan 27 2020 14:46:15 GMT+0100	✔	view	Deploy
eval_set2_model_xyz	6627556576410468352	domain_xyz_v2_20200127	Jan 27 2020 14:43:41 GMT+0100	✔	view	Deploy
eval_audio_set_abc_model_xyz_v2	6627555733804154880	domain_xyz_v2_20200127	Jan 27 2020 14:40:20 GMT+0100	✔	view	Deployed

●: Scheduled
✔: Success
✘: Failed
Evaluation: Trained LM is being evaluated.
Job files: Link to evaluation job results.
Deploy - Evaluated model is ready to deploy.
Deployed - Evaluated model was deployed to transcription service.

Powered by KeystoneJS.

WARNING When you decide to deploy a new model to your transcription service, you may affect the performance of the service positively or negatively. This means you should always think twice about whether or not to deploy a new model.

6.6 Recommended Usage

Generally speaking, it is best to submit all the data that you want your model to be trained on. This means that when you get more data you should add it to your original data and train a model using all the available data.

If you want to do fast improvements on a specific domain, however, the recommended approach is to iterate training given new data for a small number of iterations. This implies a sequence of steps, given you already have an existing model (*model A*):

1. Transcribe a set of recordings using the current model.
2. Manually correct the output transcriptions.
3. Upload the manually corrected transcriptions to the tool, and choose the retraining option, using *model A* as background model. The result should be a new model (*model Abis*).
4. Evaluate the new model (*model Abis*) with the Model Evaluation function.
5. If the accuracy is better than the previous model *model A*, you can choose to deploy the new model (*model Abis*).

You can iterate over these steps as you collect new data: Transcribe, Correct, Update the language model (=create a new language model), Evaluate it, and Switch to the new model (if it is better than the previous one). But, as mentioned earlier, it is **better to submit the full data** to obtain more realistic statistics about the language, so the iterative approach should be used with caution.

7 References

- TLTS Capabilities `https://https://"PublicDNS(IPv4)ofEC2Instance":8082/`.
- TLTS API `https://https://"PublicDNS(IPv4)ofEC2Instance":8082/docs`.
- TLACS Application `https://https://"PublicDNS(IPv4)ofEC2Instance":3000/`.

End of Document (this page was intentionally left blank)