SlashML

Release 0.1.4

slashml.com

Dec 21, 2023

DEPLOYMENT EXAMPLES

SlashML is a framework that provides the best performing ML models under one API. This enables the users to rapidly prototype their machine-learning solutions.

CHAPTER

ONE

QUICK START

To quickly play around with the model we recommend using the Python SDK client. You can install the client using pip.

```
pip install slashml
```

CHAPTER

TWO

EXAMPLE - TEXT TO SPEECH

```python
from slashml import TextToSpeech

# Replace `API_KEY` with your SlasML API token. This example still runs without
# the API token but usage will be limited
API_KEY = "YOUR_API_KEY"

service_provider = TextToSpeech.ServiceProvider.AWS
input_text = "To be or not to be, that is the question!"

# Find all the service providers that we support by running the choices() method
print(f"Available providers: {TextToSpeech.ServiceProvider.choices()}")
print(f"Selected provider: {service_provider}")

model = TextToSpeech(api_key=API_KEY)

# Submit request
job = model.execute(text=input_text, service_provider=service_provider)

print(f"\n\n\n You can access the audio file here: {job.audio_url}")
```

CHAPTER

THREE

EXAMPLE - TEXT SUMMARIZATION

```
from slashml import TextSummarization


# Replace `API_KEY` with your SlasML API token. This example still runs without
# the API token but usage will be limited
API_KEY = None
service_provider = TextSummarization.ServiceProvider.OPENAI

input_text = """A good writer doesn't just think, and then write down what he thought,␣
↪as a sort of transcript. A good writer will almost always discover new things in the␣
↪process of writing. And there is, as far as I know, no substitute for this kind of␣
↪discovery. Talking about your ideas with other people is a good way to develop them.␣
↪But even after doing this, you'll find you still discover new things when you sit down␣
↪to write. There is a kind of thinking that can only be done by writing."""

model = TextSummarization(api_key=API_KEY)

# Find all the service providers that we support by running the choices() method
print(f"Available providers: {TextSummarization.ServiceProvider.choices()}")
print(f"Selected provider: {service_provider}")

response = model.execute(text=input_text, service_provider=service_provider)

print(f"Summary = {response.summarization_data}")
```

CHAPTER

FOUR

EXAMPLE - SPEECH TO TEXT

```python
from slashml import SpeechToText

# Replace `API_KEY` with your SlasML API token. This example still runs without
# the API token but usage will be limited
API_KEY = "YOUR_API_KEY"

service_provider = SpeechToText.ServiceProvider.AWS

# we support many formats, including mp3, mp4, wav, etc.
file_path = "test.mp4"

# Find all the service providers that we support by running the choices() method
print(f"Available providers: {SpeechToText.ServiceProvider.choices()}")
print(f"Selected provider: {service_provider}")


model = SpeechToText(api_key=API_KEY)

# Upload audio
uploaded_file = model.upload_audio(file_path)
print(f"file uploaded: {uploaded_file}")

response = model.execute(
    upload_url=uploaded_file["upload_url"], service_provider=service_provider
)

print(f"\n\n\n\nTranscription = {response.transcription_data.transcription}")
```

CHAPTER

FIVE

EXAMPLE - MODEL DEPLOYMENT

CHAPTER

SIX

EXAMPLES

You can find production ready examples here here.

CHAPTER

SEVEN

API TOKEN

There is a daily limit (throttling) on the number of calls the user performs. The code can run without specifying the API key. The throttling kicks in and prevents new jobs after exceeding 10 calls per minute.

If the user intends on using the service more frequently, it is recommended to generate a token or API key from here. You can pass the API key when creating a model, if you don't the API will still work but you will be throttled.

CHAPTER

EIGHT

DEPLOYING THE ABOVE EXAMPLES

- Deploy Demos: It takes around 2-5 minutes to deploy our demos.

8.1 Dash + Render Deployment

8.1.1 Render

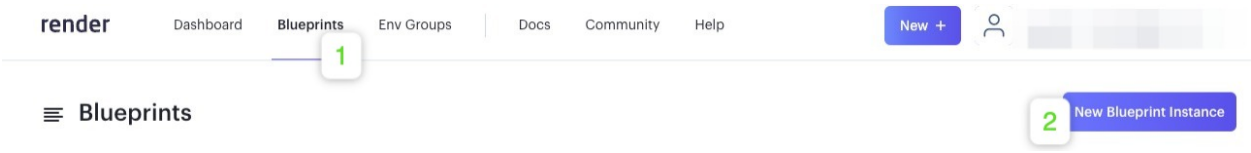
Render is a unified cloud to build and run all your apps and websites with free TLS certificates, a global CDN, DDoS protection, private networks, and auto deploys from Git.

8.1.2 Create Account

To deploy this demo on Render, you first need to login or create an account on Render and navigate to your Render Dashboard.

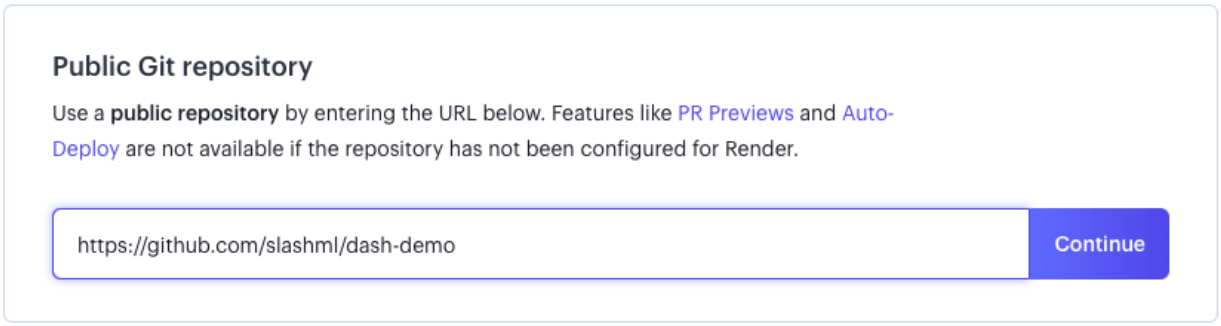
8.1.3 Blueprint Instances

Then, naviagate to `Blueprints` → `New Blueprint Instance`



8.1.4 Connect Repo

Under `Public Git repository`, set the URL to `https://github.com/slashml/dash-demo`

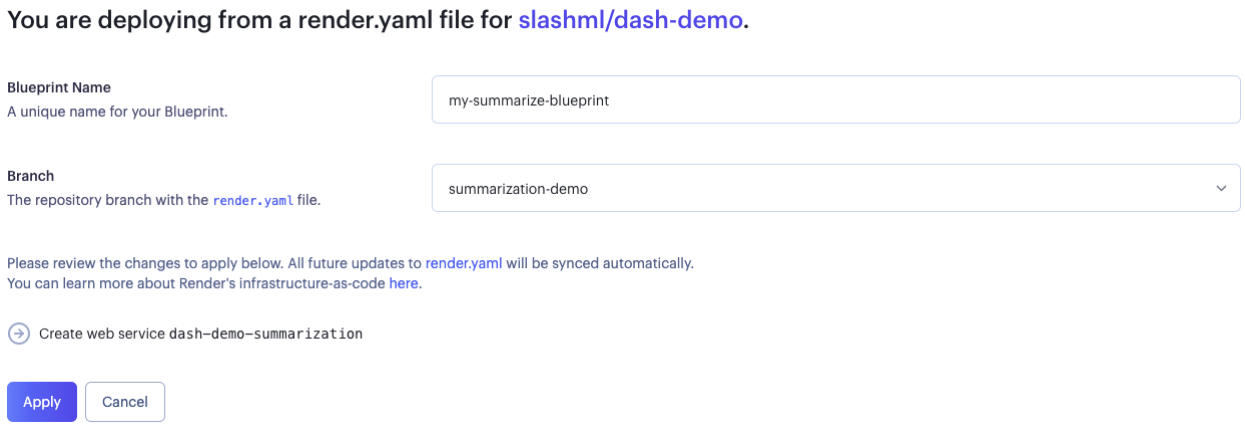


`Blueprint Name`: Give your Blueprint a unique name.

`Branch`: Each demo is on a different branch. Select the demo that you want to deploy from the dropdown

8.1.5 Create Blueprint

Once you have set `Blueprint Name` and `Branch`, press `Apply`

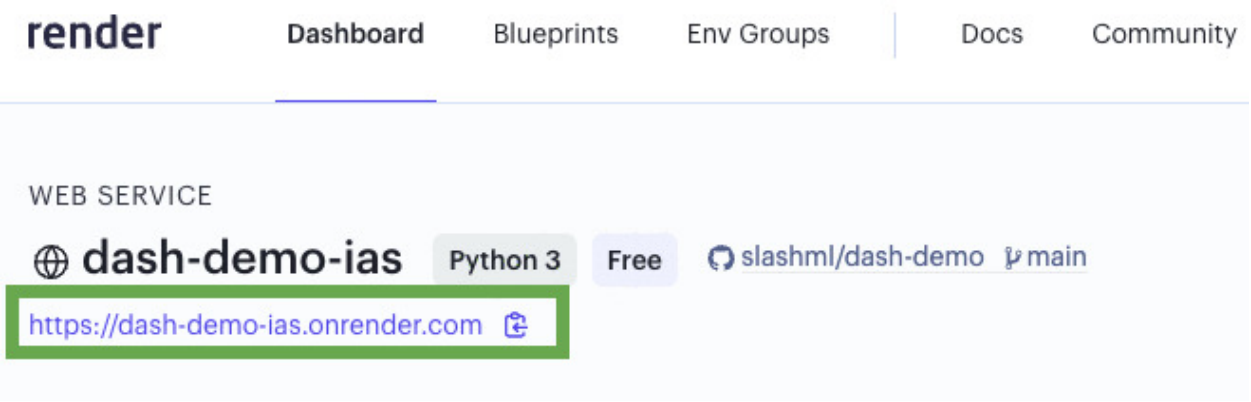


Wait for the blueprint to deploy the web-service, which you can see in the Dashboard.

8.1.6 View Deployment

Navigate to Dashboard and click on your newly created service.

You can find the URL of the deployed web-service at the top of the page.



CHAPTER

NINE

PYTHON SDK

Complete documentation on all methods and classes for SlashML.

9.1 Python SDK Reference

Complete documentation on all the methods and classes for the Python SDK

9.1.1 Speech To Text

This is the API Reference documentation for Speech to Text service.

With this endpoint, you can convert your audio files to text.

class slashml.**SpeechToText**(*api_key: str = None*)

Speech to Text Service

class ServiceProvider(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

ASSEMBLY = 'assembly'

AWS = 'aws'

DEEPGRAM = 'deepgram'

GOOGLE = 'google'

REV = 'rev'

WHISPER = 'whisper'

classmethod choices()

execute(*upload_url: str*, *service_provider:* slashml.speech_to_text.SpeechToText.ServiceProvider)

Waits for the job to be completed before returning a response

status(*job_id: str*, *service_provider:* slashml.speech_to_text.SpeechToText.ServiceProvider)

Check job status

submit_job(*upload_url: str*, *service_provider:* slashml.speech_to_text.SpeechToText.ServiceProvider)

Submit job

upload_audio(*file_location: str*)

Upload audio to server

9.1.2 Text Summarization

This is the API Reference documentation for Text Summarization service.

This endpoint summarizes a given text input.

class slashml.**TextSummarization**(*api_key: str = None*)

Text Summarization Service

class ServiceProvider(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

HUGGING_FACE = 'hugging-face'

OPENAI = 'openai'

classmethod choices()

execute(*text: str*, *service_provider:* slashml.text_summarization.TextSummarization.ServiceProvider)

Waits for the job to be completed before returning a response

status(*job_id: str*, *service_provider:* slashml.text_summarization.TextSummarization.ServiceProvider)

Check job status

submit_job(*text: str*, *service_provider:* slashml.text_summarization.TextSummarization.ServiceProvider)

Submit Job to server

9.1.3 Text To Speech

This is the API Reference documentation for Text to Speech service.

This endpoint converts a given text input into speech. The response contains the link to an audio file.

class slashml.**TextToSpeech**(*api_key: str = None*)

Text To Speech Service

class ServiceProvider(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

AWS = 'aws'

GOOGLE = 'google'

classmethod choices()

execute(*text: str*, *service_provider:* slashml.text_to_speech.TextToSpeech.ServiceProvider)

Waits for the job to be completed before returning a response

status(*job_id: str*, *service_provider:* slashml.text_to_speech.TextToSpeech.ServiceProvider)

Check job status

submit_job(*text: str*, *service_provider:* slashml.text_to_speech.TextToSpeech.ServiceProvider)

Submit Job to server

9.1.4 Model Deployment

This is the API Reference documentation for the Model Deployment Class

This class deploys a user created model and exposes and endpoing for prediction

class slashml.**ModelDeployment**(*api_key: str = None*)

create_tar_gz(*, *folder_path*, *tar_gz_filename*)

deploy(*, *model_name: str*, *model: str*, *requirements: Optional[list] = None*)

Submit job

predict(*model_version_id: str*, *model_input: str*)

Check job status

status(*, *model_version_id: str*)

Check job status

CHAPTER

TEN

MODEL DEPLOYMENT

Complete documentation on all methods and classes for SlashML.

10.1 Hugging Face to SlashML

First make sure you have the following dependencies installed

```
pip install slashml
pip install transformer, torch
```

Then follow the following example to deploy a Hugging Face model to SlashML

10.2 Pytorch to SlashML

First make sure you have the following dependencies installed

```
pip install torch
pip install torchvision
pip install sklearn
```

Then follow the following example to deploy a Hugging Face model to SlashML

10.3 Scikit-Learn to SlashML

First make sure you have the following dependencies installed

```
pip install slashml
pip install sklearn
```

Then follow the following example to deploy a Hugging Face model to SlashML

CHAPTER

ELEVEN

FULL API REFERENCE

Models on SlashML can be used in languages other than Python. The full API reference can be found here. If you would like to create a client in another language, please contact us at *faizank@slashml.com*. We would love to help you out!.

11.1 Model Deployment API

This services allows you to fetch the status of a deployed model and call predictions on the exposed API.

11.1.1 Time to Integrate

Less than 5 minute

11.1.2 Instructions

1. (Optional) For deployment we recommend using the python SDK. However, you can also push your model using an API. To do this, you will first have to install `truss` to create a truss folder locally. Then compress that truss folder into a `.tar.gz` file. Then send a post `POST` request to `https://api.slashml.com/model-deployment/v1/models` with the compressed file as the body of the request. Save the `id` in the response object.
2. Check the status of the model deployment by sending a `GET` request to `https://api.slashml.com/model-deployment/v1/models/YOUR-MODEL-ID/status`
3. You can make predctions on the deployed model by sending a `POST` request to `https://api.slashml.com/model-deployment/v1/models/YOUR-MODEL-ID/predict`. The body should contain a json object with `model_input` which is the input prompt to the model.

11.1.3 Code Blocks

Submit model for deployment

Install truss

```
pip install truss
```

You can the create a truss object by running the following command from within Python

```
# you might have to install transfomers and torch
from transformers import pipeline

def train_model():
    # Bring in model from huggingface
    return pipeline('fill-mask', model='bert-base-uncased')

my_model = train_model()

# save the model
truss.create(my_model, 'my_model')
```

Then convert the folder into a `.tar.gz` file

```
tar -czvf my_model.tar.gz my_model
```

Request

Then send a post `POST` request to `https://api.slashml.com/model-deployment/v1/models` with the compressed file as the body of the request. Save the `id` in the response object.

```
import requests

url = "https://api.slashml.com/model-deployment/v1/models/"

payload={'model_name': 'test-dep-model'}

files=[
  ('model_file',('my_model.tar.gz',open('path/to/my_model.tar.gz','rb'),'application/
↪octet-stream'))
]

headers = {
  'Authorization': 'Token YOUR_TOKEN'
}

response = requests.request("POST", url, headers=headers, data=payload, files=files)

print(response.json())
```

Response (200)

```
{
    "id": "a5822206-9680-444c-87ec-4b66a7bcfc26",
    "created": "2023-06-13T06:38:55.311751Z",
    "status": "IN_PROGRESS",
    "name": "'test-dep-model'"
}
```

Response (400)

```
{
    "error" : {
        "message" : "something bad happened",
    }
}
```

Check status of model

```
GET https://api.slashml.com/model-deployment/v1/models/YOUR-MODEL-ID/status
```

Request

```
import requests

url = 'https://api.slashml.com/model-deployment/v1/models/YOUR-MODEL-ID/status'

headers = {
  'Authorization': 'Token <YOUR_API_KEY>'
}

response = requests.get(url, headers=headers, data=payload)
print(response.json())
```

Response (200) - MODEL-READy

```
{
    # keep track of the id for later
    "id": "ozfv3zim7-9725-4b54-9b71-f527bc21e5ab",
    "created": "2023-06-13T06:38:55.311751Z",
    "name": "test-dep-model",
    "status": "MODEL_READY",
    "name": "test-dep-model",
}
```

Response (400) - Error

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The status will go from 'QUEUED' to 'BUILDING_MODEL' to 'DEPLOYING_MODEL' to 'MODEL_READY'. If there's an error processing your input, the status will go to 'FAILURE' and there will be an 'ERROR' key in the response JSON which will contain more information.

Submit a prediction to the model

Request

Then send a post `POST` request to `https://api.slashml.com/model-deployment/v1/models/YOUR-MODEL-ID/predict` with the model-input as the body of the request.

```
import requests
import json

url = "https://api.slashml.com/model-deployment/v1/models/YOUR-MODEL-ID/predict"

payload = json.dumps({
  "model_input": [
    "steve jobs is the [MASK] of apple"
  ]
})

headers = {
  'Authorization': 'Token a7011983a0f3d64ee113317b1e36f8e5bf56c14a',
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

Response (200)


```
{
    "id": "a5822206-9680-444c-87ec-4b66a7bcfc26",
    "model_input": [
        "steve jobs is the [MASK] of apple"
    ],
    "model_response": {
        "predictions": [
            {
                "score": 0.516463041305542,
                "sequence": "steve jobs is the founder of apple",
                "token": 3910,
                "token_str": "founder"
            },
            {
                "score": 0.3604991137981415,
                "sequence": "steve jobs is the ceo of apple",
                "token": 5766,
                "token_str": "ceo"
            },
            {
                "score": 0.04929964989423752,
                "sequence": "steve jobs is the president of apple",
```


(continues on next page)

(continued from previous page)


```
                "token": 2343,
                "token_str": "president"
            },
            {
                "score": 0.021112028509378433,
                "sequence": "steve jobs is the creator of apple",
                "token": 8543,
                "token_str": "creator"
            },
            {
                "score": 0.008550147525966167,
                "sequence": "steve jobs is the father of apple",
                "token": 2269,
                "token_str": "father"
            }
        ]
    }
}
```


Response (400)


```
{
    "error": "some error occured when requesting job status",
    "full_message": [
        "{'error': ErrorDetail(string='model not ready', code='permission_denied'),
↪'reasons': [ErrorDetail(string='model not ready', code='permission_denied'),␣
↪ErrorDetail(string='model is still being built or deployed', code='permission_denied
↪')]}"
    ]
}
```


11.2 Speech To Text API

This service exposes the speech-to-text models from a variety of different vendors.

11.2.1 Time to Integrate

Less than 5 minute

11.2.2 Service Providers

You can find a list of all service providers here: *slashml.SpeechToText.ServiceProvider*

11.2.3 Instructions

1. (Optional) Upload files to a static server by sending `POST` request to `https://api.slashml.com/speech-to-text/v1/upload/` where the data points to your audio file. Save the `upload_url`. You can use this url link in the rest of the calls.
2. Submit your audio file for transcription by sending `POST` request to `https://api.slashml.com/speech-to-text/v1/jobs/`. The body should contain a json object with `audio_url` which points to an audio file that is publicly available. Save the `id` in the response object.
3. Check the status of the transcription by sending a `GET` request to `https://api.slashml.com/speech-to-text/v1/jobs/YOUR-TRANSCRIPT-ID/`

   Note: The uploaded_url will be used when submitting a transcription job.

11.2.4 Code Blocks

Upload audio file to static storage

If your audio files aren't accessible via a URL already (like in an S3 bucket, or a static file server), you can upload your audio file using this API. All uploads have a 24hr deletion policy.

```
POST https://api.slashml.com/speech-to-text/v1/upload/
```

Request

```
import requests

url = "https://api.slashml.com/speech-to-text/v1/upload/"

headers = {'authorization': "Token <YOUR_API_KEY>"}
payload = { 'service_provider':'assembly' }
files=[
  ('audio',('test_audio.mp3',open('/path/to/your_audio.mp3','rb'),'audio/mpeg'))
]

response = requests.post(url, headers = headers, data = payload, files = files)

print(response.text)
```

Response (200)

```
{
    "uploaded_url": "https://cdn.slashml.com/upload/ccbbbfaf-f319-4455-9556-272d48faaf7f"
}
```

Response (400)

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The uploaded_url will be used when submitting a transcription job.

Convert audio to text

The body of the request should contain a field `uploaded_audio_url`, the value of which shall contain the link to the uploaded audio url, and `service_provider` which is the name of the service provider you want to use.

```
POST https://api.slashml.com/speech-to-text/v1/jobs/
```

Request

```
import requests

url = 'https://api.slashml.com/speech-to-text/v1/jobs/'

payload = {
  "uploaded_audio_url": "https://cdn.slashml.com/upload/ccbbbfaf-f319-4455-9556-
↪272d48faaf7f",
  "service_provider": 'assembly'
}
headers = {
    "Authorization": "Token <YOUR_API_KEY>",
}

response = requests.post(url, headers=headers, data=payload)
print(response.text)
```

Response (200)

```
{
    # keep track of the id for later
    "id": "ozfv3zim7-9725-4b54-9b71-f527bc21e5ab",
    # note that the status is now "processing"
    "status": "IN_PROGESS",
    "audio_duration": null,
    "audio_url": "https://cdn.slashml.com/upload/ccbbbfaf-f319-4455-9556-272d48faaf7f",
    "text": null,
}
```

Response (400)

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The 'id' will be used to fetch the status of the job, in the status endpoint.

Check status of job

The request API is similar to all job submissions. We can make requests to GET the status of the jobs, and eventually the result of the submitted job, i.e. transcription, or speechification.

```
GET https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/
```

Request

```
import requests

url = 'https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/?service_
↪provider=assembly'

headers = {
  'Authorization': 'Token <YOUR_API_KEY>'
}

response = requests.get(url, headers=headers, data=payload)
print(response.text)
```

Response (200) - In Progress

```
{
    # keep track of the id for later
    "id": "ozfv3zim7-9725-4b54-9b71-f527bc21e5ab",
    # note that the status is now "processing"
    "status": "IN_PROGESS",
    "acoustic_model": "assemblyai_default",
    "language_model": "assemblyai_default",
    "audio_duration": null,
    "audio_url": "https://s3-us-west-2.amazonaws.com/blog.assemblyai.com/audio/8-7-2018-
↪post/7510.mp3",
    "confidence": null,
    "dual_channel": null,
    "text": null,
    "words": null
}
```

Response (200) - Completed

```
{
    "id": "5551722-f677-48a6-9287-39c0aafd9ac1",
    "status": "COMPLETED",
    "acoustic_model": "assemblyai_default",
    "language_model": "assemblyai_default",
    "audio_duration": 12.0960090702948,
    "audio_url": "https://s3-us-west-2.amazonaws.com/blog.assemblyai.com/audio/8-7-2018-
↪post/7510.mp3",
    "confidence": 0.956,
    "dual_channel": null,
    "text": "You know Demons on TV like that and and for people to expose themselves to␣
↪being rejected on TV or humiliated by fear factor or.",
    "words": [
        {
            "confidence": 1.0,
            "end": 440,
            "start": 0,
            "text": "You"
        },
        ...
        {
            "confidence": 0.96,
            "end": 10060,
            "start": 9600,
            "text": "factor"
        },
        {
            "confidence": 0.97,
            "end": 10260,
            "start": 10080,
            "text": "or."
```

(continues on next page)

(continued from previous page)

```
        }
    ]
}
```

Response (400) - Error

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The status will go from 'QUEUED' to 'IN_PROGRESS' to 'COMPLETED'. If there's an error processing your input, the status will go to 'ERROR' and there will be an 'ERROR' key in the response JSON which will contain more information.

11.3 Text Summarization API

This service exposes the best performing summarization models from a variety of different vendors.

11.3.1 Time to Integrate

Less than 1 minute

11.3.2 Service Providers

You can find a list of all service providers here: *slashml.TextSummarization.ServiceProvider*

11.3.3 Instructions

1. Copy your text and pass it in the body of a `POST` request to `https://api.slashml.com/speech-to-text/v1/jobs/`. The body should contain a json object with text field. Which contains the body of the text that you want to summarize. Save the `id` in the response object.
2. Check the status of the transcription by sending a `GET` request to `https://api.slashml.com/speech-to-text/v1/jobs/`

11.3.4 Code Blocks

Submit text input for summarization

If your audio files aren't accessible via a URL already (like in an S3 bucket, or a static file server), you can upload your audio file using this API. All uploads are immediately deleted after transcription, we do not store the uploads.

```
POST https://api.slashml.com/summarization/v1/summarize/
```

Request

```
import requests

  url = 'https://api.slashml.com/speech-to-text/v1/jobs/'

  payload = {
  "text":

  '"One reason programmers dislike meetings so much is that they're on a different type of␣
↪schedule from other people. Meetings cost them more.

  There are two types of schedule, which I'll call the manager's schedule and the maker's␣
↪schedule. The manager's schedule is for bosses. It's embodied in the traditional␣
↪appointment book, with each day cut into one hour intervals. You can block off several␣
↪hours for a single task if you need to, but by default you change what you're doing␣
↪every hour."',

  "service_provider": "openai"
  }

  headers = {
      "Authorization": "Token <YOUR_API_KEY>",
  }

  response = requests.post(url, headers=headers, data=payload)

  print(response.text)
```

Response (200)

```
{
    # keep track of this id for later
    "id": "94f52a16-f2d9-4212-be98-ac87eb068d22",
    "status": "IN_PROGRESS",
    "created": "2022-12-24T01:14:18.543943Z"
}
```

Response (400)

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The 'id' will be used to fetch the status of the summarization job.

Check status of job

The request API is similar to all job submissions. We can make requests to GET the status of the jobs, and eventually the result of the submitted job, i.e. transcription, or speechification.

```
GET https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/
```

Request

```
import requests

url = 'https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/?service_
↪provider=assembly'

headers = {
  'Authorization': 'Token <YOUR_API_KEY>'
}

response = requests.get(url, headers=headers, data=payload)
print(response.text)
```

Response (200) - In Progress

```
{
    # keep track of the id for later
    "id": "ozfv3zim7-9725-4b54-9b71-f527bc21e5ab",
    # note that the status is now "processing"
    "status": "IN_PROGESS",
    "acoustic_model": "assemblyai_default",
    "language_model": "assemblyai_default",
    "audio_duration": null,
    "audio_url": "https://s3-us-west-2.amazonaws.com/blog.assemblyai.com/audio/8-7-2018-
↪post/7510.mp3",
    "confidence": null,
    "dual_channel": null,
    "text": null,
    "words": null
}
```

Response (200) - Completed


```
{
    "id": "5551722-f677-48a6-9287-39c0aafd9ac1",
    "status": "COMPLETED",
    "acoustic_model": "assemblyai_default",
    "language_model": "assemblyai_default",
    "audio_duration": 12.0960090702948,
    "audio_url": "https://s3-us-west-2.amazonaws.com/blog.assemblyai.com/audio/8-7-2018-
↪post/7510.mp3",
    "confidence": 0.956,
    "dual_channel": null,
    "text": "You know Demons on TV like that and and for people to expose themselves to␣
↪being rejected on TV or humiliated by fear factor or.",
    "words": [
        {
            "confidence": 1.0,
            "end": 440,
            "start": 0,
            "text": "You"
        },
        ...
        {
            "confidence": 0.96,
            "end": 10060,
            "start": 9600,
            "text": "factor"
        },
        {
            "confidence": 0.97,
            "end": 10260,
            "start": 10080,
            "text": "or."
        }
    ]
}
```


Response (400) - Error


```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```


Note: The status will go from 'QUEUED' to 'IN_PROGRESS' to 'COMPLETED'. If there's an error processing your input, the status will go to 'ERROR' and there will be an 'ERROR' key in the response JSON which will contain more information.

11.4 Text To Speech API

This service provides the best performing automatic speechification models from a variety of different vendors.

11.4.1 Time to Integrate

Less than 1 minute

11.4.2 Service Providers

You can find a list of all service providers here: *slashml.TextToSpeech.ServiceProvider*

11.4.3 Instructions

1. Send a POST request to `https://api.slashml.com/text-to-speech/v1/upload/` with the text in the body. Save the id in the response object returned.
2. Check the status of the speechification by sending a `GET` request to `https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/`

11.4.4 Code Blocks

Convert text to audio

The body of the request should contain a field `text`, the value of which shall contain the relevant text that you want to generate an audio file for.

```
POST https://api.slashml.com/text-to-speech/v1/jobs/
```

Request

```python
import requests

url = 'https://api.slashml.com/text-to-speech/v1/jobs/'

payload = {
  "text": "this is my text that I want to convert to an audio file"
  "service_provider": 'google'
}
headers = {
    "Authorization": "Token <YOUR_API_KEY>",
}

response = requests.post(url, headers=headers, data=payload)
print(response.text)
```

Response (200)

```
{
    # keep track of the id for later
    "id": "dbe42f9c-f0ce-4683-af80-e6c250b21460",
    "status": "COMPLETED",

    # url to publicly hosted audio file
    "audio_url": "https://slashml.s3.ca-central-1.amazonaws.com/70c34009-8149-4b47-8a6e-
↪29dcb5bd3d3d.mp3"
}
```

Response (400)

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The 'id' will be used to fetch the status of the job, in the status endpoint.

Check status of job

The request API is similar to all job submissions. We can make requests to GET the status of the jobs, and eventually the result of the submitted job, i.e. transcription, or speechification.

```
GET https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/
```

Request

```
import requests

url = 'https://api.slashml.com/speech-to-text/v1/jobs/YOUR-JOB-ID/?service_
↪provider=assembly'

headers = {
  'Authorization': 'Token <YOUR_API_KEY>'
}

response = requests.get(url, headers=headers, data=payload)
print(response.text)
```

Response (200) - In Progress

```
{
    # keep track of the id for later
    "id": "ozfv3zim7-9725-4b54-9b71-f527bc21e5ab",
    # note that the status is now "processing"
    "status": "IN_PROGESS",
    "acoustic_model": "assemblyai_default",
    "language_model": "assemblyai_default",
    "audio_duration": null,
    "audio_url": "https://s3-us-west-2.amazonaws.com/blog.assemblyai.com/audio/8-7-2018-
↪post/7510.mp3",
    "confidence": null,
    "dual_channel": null,
    "text": null,
    "words": null
}
```

Response (200) - Completed

```
{
    "id": "5551722-f677-48a6-9287-39c0aafd9ac1",
    "status": "COMPLETED",
    "acoustic_model": "assemblyai_default",
    "language_model": "assemblyai_default",
    "audio_duration": 12.0960090702948,
    "audio_url": "https://s3-us-west-2.amazonaws.com/blog.assemblyai.com/audio/8-7-2018-
↪post/7510.mp3",
    "confidence": 0.956,
    "dual_channel": null,
    "text": "You know Demons on TV like that and and for people to expose themselves to␣
↪being rejected on TV or humiliated by fear factor or.",
    "words": [
        {
            "confidence": 1.0,
            "end": 440,
            "start": 0,
            "text": "You"
        },
        ...
        {
            "confidence": 0.96,
            "end": 10060,
            "start": 9600,
            "text": "factor"
        },
        {
            "confidence": 0.97,
            "end": 10260,
            "start": 10080,
            "text": "or."
```

(continues on next page)

(continued from previous page)

```
        }
    ]
}
```

Response (400) - Error

```
{
    "error" : {
        "message" : "something bad happened"
    }
}
```

Note: The status will go from 'QUEUED' to 'IN_PROGRESS' to 'COMPLETED'. If there's an error processing your input, the status will go to 'ERROR' and there will be an 'ERROR' key in the response JSON which will contain more information.

CHAPTER

TWELVE

USE CASES

This section contains some of the use-cases that we have found for our services.

12.1 Deploy a Hugging Face model

In this example we will show how to deploy a Hugging Face model to SlashMLO. We will use the Hugging Face Transformers and deploy it to SlashML. Then we will use the endpoint exposed by SlashML to make predictions on our model.

```
from slashml import ModelDeployment
import time

# you might have to install transfomers and torch
from transformers import pipeline
```

```
def train_model():
    # Bring in model from huggingface
    return pipeline('fill-mask', model='bert-base-uncased')

my_model = train_model()
```

```
# Replace `API_KEY` with your SlasML API token. Fetch the api key from https://slashml.
↪com/settings/api-key

model = ModelDeployment(api_key=None)

# deploy model
response = model.deploy(model_name='my_model_3', model=my_model)
```

```
# wait for it to be deployed
time.sleep(2)
status = model.status(model_version_id=response.id)

while status.status != 'READY':
    print(f'status: {status.status}')
    print('trying again in 5 seconds')
    time.sleep(5)
    status = model.status(model_version_id=response.id)
```

(continues on next page)

(continued from previous page)

```
    if status.status == 'FAILED':
        raise Exception('Model deployment failed')
```

```
# submit prediction
input_text = 'Steve jobs is the [MASK] of Apple.'
prediction = model.predict(model_version_id=response.id, model_input=input_text)
print(prediction)
```

12.2 Speech Summarizer

In this example we will show how to combine the power of the Speech Recognition and the Summarization models to create a Speech Summarizer.

```
from slashml import SpeechToText, TextSummarization
```

```
# Replace `API_KEY` with your SlasML API token. This example still runs without
# the API token but usage will be limited
API_KEY = "YOUR_API_KEY"


# 10 minute audio file already uploaded
uploaded_url = (
    "https://slashml.s3.ca-central-1.amazonaws.com/fda70f6a-6057-4541-adf1-2cf4f4182929"
)
```

```
service_provider_speech_to_text = SpeechToText.ServiceProvider.WHISPER
service_provider_summarize = TextSummarization.ServiceProvider.OPENAI


transcribe = SpeechToText(api_key=API_KEY)
summarize = TextSummarization(api_key=API_KEY)
```

```
response = transcribe.execute(
    upload_url=uploaded_url, service_provider=service_provider_speech_to_text
)

print('starting pipeline, the first response might take 10 secs')
transcribed_text = response.transcription_data.transcription
print (f"Transcribed Text = {transcribed_text}")
```

```
response_summarize = summarize.execute(transcribed_text, service_provider_summarize)

summary = response_summarize.summarization_data

print (f"Summarized Text = {summary}")
```

CHAPTER

THIRTEEN

ADDITIONAL RESOURCES

Additional resources we think may be useful as you develop your application!

- Website: Visit our website to manage your slashml account.
- Twitter: Follow us on twitter to receive the latest updates!

INDEX