Language Identification Container 1.0.0

Table of Contents

Language Identification Container

Versions

1.0.0

This is the first release of Speechmatics' Language Identification (ID) Container. Language ID currently identifies the predominant language spoken in a media file and helps with automating the process of selecting which language pack to use for transcription. 12 languages are supported.

Language Identification Container Quick Start Guide

This guide will walk you through the steps needed to deploy the Speechmatics Batch Language Identification container.

This container will allow you to predict the predominant, most likely language spoken in a media file. You can use the predicted language to select the correct transcriber when the language spoken in your file is unknown.

The following steps are required to use this in your environment:

- Check system requirements
- Pull the Docker Image into your local Docker Registry
- Run the container

System Requirements

Speechmatics containerized deployments are built on the Docker platform. In order to operate the containers, the following requirements will need to be met.

A single docker image can be used to create and run multiple containers concurrently, each running container will require the following resources:

- 1 vCPU with AVX2 support
- 1 GB RAM

The raw image size of the Language Identification Container is around 800MB.

Prerequisites

- Requires either a license file or license token before running the language identification
- Access to our docker repository
- Audio file (we recommend at least 60 seconds of speech in it for a high accuracy)

Note: If you do not have a license or access to the docker repository, please contact Speechmatics support support@speechmatics.com.

Workflow

- Run the Language ID docker container with an audio file
- Receive the output JSON with the predicted language code
- Use that language code to run transcription with any of the Speechmatics deployments

Supported Languages

The following languages are supported (language codes adhere the ISO-639 standard):

Language	ISO Code
Chinese Mandarin	cmn
German	de
English	en
Spanish	es
French	fr
Hindi	hi
Italian	it
Japanese	ja
Korean	ko
Dutch	nl
Portuguese	pt
Russian	ru

Only these language codes can be predicted and match with the language codes used for transcription.

Supported File Formats

Only the following file formats are supported:

- aac
- amr
- flac
- m4a
- mov
- mp3
- mp4
- mpeg
- ogg
- wav

Limitations

- This container only works in 'batch' mode - it receives a whole file, processes the file and then returns the results, without any intermediate output
- The observed accuracy increases with the amount of speech content inside the file. We see the best accuracy when there are 60 seconds or more of speech
- It's not possible to predict the language of each channel independently in a multi-channel media file; any multi-channel files are converted to mono before identifying the language
- Inverted multi-channel audio is not supported, this is where the second channel is the inverse of first
- The container uses CPU and doesn't run on a GPU

Accessing the Image

Software Repository Login

The Language Identification container can be accessed from the Speechmatics Docker repository (jfrog.io). If you do not have a Speechmatics software repository account or have lost your details, please contact Speechmatics support support@speechmatics.com.

Log into the Speechmatics Docker repository to retrieve the container

```
docker login https://speechmatics-docker-public.jfrog.io
```

You will be prompted for a username and password. If successful, you will see the response:

```
Login Succeeded
```

If unsuccessful, please verify your credentials and URL. If problems persist, please contact Speechmatics support.

Pulling the Image

Please run the following Docker command to download the Language Identification container to your local environment. For example to download version 1.0.0 of the container, you can use

```
docker pull speechmatics-docker-public.jfrog.io/langid:1.0.0
```

Note: Speechmatics require all customers to cache a copy of the Docker image(s) within their own environment. Please do not pull directly from the Speechmatics software repository for each deployment.

The image will start to download. This could take a while depending on your connection speed.

Licensing

You should have received a confidential license file from Speechmatics containing a token to use to license your container. The contents of the file received should look similar to this:

```
{
    "contractid": 1,
    "creationdate": "2022-06-01 09:04:11",
    "customer": "Speechmatics",
    "id": "c18a4eb990b143agadeb384cbj7b04c3",
    "metadata": {
        "key_pair_id": 1,
        "request": {
            "customer": "Speechmatics",
            "features": [
                "MAPBA",
                "ALID"
            ],
            "notValidAfter": "2023-01-01",
            "validFrom": "2022-01-01"
        }
    },
    "signedclaimstoken": "example"
}
```

There are two ways to apply the license to the container.

- As a volume-mapped file

  The license file should be mapped to the path `/license.json` within the container. For example:

  ```
  docker run ... -v /my_license.json:/license.json:ro speechmatics-docker-
  public.jfrog.io/langid:1.0.0
  ```

- As an environment variable

  Setting an environment variable named `LICENSE_TOKEN` is also a valid way to license the container. The contents of this variable should be set to the value of the `signedclaimstoken` from within the license file. For example, copy the `signedclaimstoken` from the license file (without the quotation marks) and set the environment variable as below. The token example is not a full example:

  ```
  docker run ... -e LICENSE_TOKEN=eyJhbGciOiJ... speechmatics-docker-
  public.jfrog.io/langid:1.0.0
  ```

There should be no reason to do this, but if both a volume-mapped file and an environment variable are provided simultaneously then the volume-mapped file will be ignored.

Using the Container

Once the Docker image has been pulled into a local environment, it can be started using the Docker run command. More details about operating and managing the container are available in the Docker API documentation.

There are two different methods for passing a media file into a container:

- STDIN: Streams media file into the container through the standard command line entry point
- File Location: Pulls media file from a file location

Here are some examples below to demonstrate these modes of operating the container.

Example 1: passing a file using the cat command to the container

```
cat ~/$AUDIO_FILE | docker run -i -e LICENSE_TOKEN=eyJhbGciOiJ... speechmatics-docker-
public.jfrog.io/langid:1.0.0
```

Example 2: pulling a media file from a mapped directory into the container

```
docker run -v $AUDIO_FILE:/input.audio -e LICENSE_TOKEN=eyJhbGciOiJ... speechmatics-docker-
public.jfrog.io/langid:1.0.0
```

NOTE: the media file must be mapped into the container with :/input.audio

The Docker `run` options used are:

Name	Description
`--env, -e`	Set environment variables
`--volume , -v`	Bind mount a volume

See Docker docs for a full list of the available options.

Both the methods will produce the same identification result. STDOUT is used to provide the result in JSON format. Here's an example of the returned JSON:

```
{
  "format": "1.0",
  "metadata": {
    "created_at": "2022-06-13T11:27:19+0100",
    "duration": 300,
    "processed_duration": 247.1,
    "language_identification_config": {},
```

```
    "type": "language_identification"
  },
  "results": [
    {
      "alternatives": [
        {
          "language": "en",
          "confidence": 0.97
        },
        {
          "language": "de",
          "confidence": 0.02
        },
        {
          "language": "es",
          "confidence": 0.01
        },
        {
          "language": "fr",
          "confidence": 0
        },
        {
          "language": "nl",
          "confidence": 0
        },
        {
          "language": "hi",
          "confidence": 0
        },
        {
          "language": "cmn",
          "confidence": 0
        },
        {
          "language": "pt",
          "confidence": 0
        },
        {
          "language": "it",
          "confidence": 0
        },
        {
          "language": "ko",
          "confidence": 0
        },
        {
          "language": "ru",
          "confidence": 0
        },
        {
          "language": "ja",
          "confidence": 0
        }
      ],
      "start_time": 0,
      "end_time": 300
    }
  ]
}
```

Note: If the `results` are returned empty and `processed_duration` is 0 then no speech was detected to classify.

Note: Confidence scores are rounded to two decimal places. This means that they won't necessarily sum up to 1.

API Reference

Identification Response

Field	Type	Description	Required
format	string	Speechmatics identification JSON format version number.	Yes
metadata	Identification Metadata		Yes
results	[Identification Result]		Yes

Identification Metadata

Field	Type	Description	Required
created_at	dateTime	The UTC date time the job was created.	Yes
duration	float	The duration of the media file provided (in seconds).	Yes
processed_duration	float	The duration of detected speech in the media file provided.	Yes
language_identification_config	string	Any configuration requested when the job was submitted.	Yes
type	string	Job type. This is always "language_identification".	Yes

Identification Result

Field	Type	Description	Required
alternatives	[Identification Alternative]	List of objects sorted with descending confidence	No
start_time	float	The start of the file, marked in seconds	Yes
end_time	float	The end of the file, marked in seconds	Yes

Identification Alternative

Field	Type	Description	Required

language	string	The language code.	Yes
confidence	float	The confidence of the prediction. This is a score between 0 and 1, rounded to two decimal places. This means that they won't necessarily sum up to 1.	Yes

Ability to run a container with multiple cores

For customers who are looking to improve job turnaround time and who are able to assign sufficient resources, it is possible to pass a parallel parameter to the container to take advantage of multiple CPUs. The parameter is called `parallel` and the following example shows how it can be used. In this case to use 2 cores to process the audio you would run the container like this:

```
docker run ... speechmatics-docker-public.jfrog.io/langid:1.0.0 --parallel=2
```

Depending on your hardware, you may need to experiment to find the optimum performance. We've noticed an improvement in turnaround time for jobs by using this approach.

If you limit or are limited on the number of CPUs you can use (for example your platform places restrictions on the number of cores you can use, or you use the --cpu flag in your docker run command), then you should ensure that you do not set the parallel value to be more than the number of available cores. If you attempt to use a setting in excess of your free resources, then the container will only use the available cores.

If you are running the container on a shared resource, you may experience different results depending on what other processes are running at the same time.

Determining success

The exit code of the container will determine if the identification was successful. There are two exit code possibilities:

- Exit Code == 0: The identification was a success; the output will contain a JSON output defining the identification result
- Exit Code != 0: the output will contain useful information why the job failed. This output should be used in any communication with Speechmatics support to aid understanding and resolution of any problems that may occur

Enable Logging

If you are seeing problems then we recommend that you open a ticket with Speechmatics support: support@speechmatics.com. Please include the logging output from the container if you do open a ticket, and ideally enable verbose logging.

Verbose logging is enabled by running the container with the environment variable DEBUG set to true. All logs are written to STDERR.

```
docker run -e DEBUG=true ... speechmatics-docker-public.jfrog.io/langid:1.0.0
```