CSSR for Africa

Culturally Sensitive Social Robotics
for Africa

# D4.3.2 Speech Event

Due date: **31/03/2024**
Submission Date: **20/02/2025**
Revision Date: **30/07/2025**

Start date of project: **01/07/2023** Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa**

Responsible Person: **Clifford Onyonka** Revision: **1.3**

| **Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | **PU** |
| **PP** | Restricted to other programme participants (including Afretec Administration) | |
| **RE** | Restricted to a group specified by the consortium (including Afretec Administration) | |
| **CO** | Confidential, only for members of the consortium (including Afretec Administration) | |

## Executive Summary

Deliverable D4.3.2 concerns the results of Task 4.3.2, a task whose objective was to train, test, and finally deploy a speech-to-text model on a ROS node that will enable speech utterances in Kinyarwanda and English languages captured by Pepper's microphones to be transcribed into written text.

This report details the output of each phase of the software development process used in the fulfillment of Deliverable D4.3.2. The requirements definition section specifies the functional requirements of the users of Speech Event, the module specification section outlines the functional characteristics of Speech Event, the interface design section outlines the specification of the inputs and outputs of Speech Event, the module design section describes the deep neural networks that perform speech recognition of Kinyarwanda and English utterances, the testing section shows the results and descriptions of unit and end-to-end tests of Speech Event, and the user manual section outlines how to build and run Speech Event.

# Contents

**1 Introduction** **4**

**2 Requirements Definition** **5**

**3 Module Specification** **6**

**4 Interface Design** **7**
4.1 Source Code . . . . . 7
4.2 Configuration Files . . . . . 8
4.3 Data Files . . . . . 8
4.4 Topics Subscribing To . . . . . 9
4.5 Topics Publishing To . . . . . 9
4.6 Services Supported . . . . . 9

**5 Module Design** **10**
5.1 Model Architecture . . . . . 10
5.1.1 Preprocessor . . . . . 10
5.1.2 Encoder Architecture . . . . . 10
5.1.3 Decoder/Predictor Architecture . . . . . 11
5.1.4 Joint Network Architecture . . . . . 12
5.1.5 Combined Model Architecture . . . . . 13

**6 Testing** **14**

**7 User Manual** **17**
7.1 Installation . . . . . 17
7.2 Usage . . . . . 18
7.3 Driver ROS Node . . . . . 18

**References** **19**

**Principal Contributors** **20**

**Document History** **21**

## 1 Introduction

Conversational systems are developed using the following main components: automatic speech recognition (speech-to-text), natural language understanding, dialogue manager, natural language generation, and text-to-speech. Speech-to-text converts an audio signal containing spoken speech utterances to written text transcriptions, natural language understanding analyses the transcribed text to extract meaning, the dialogue manager generates a response based on the inferred meaning of the text, natural language generation formulates text based on the response generated by the dialogue manager, and text-to-speech synthesises spoken speech utterances to be played to the other agent(s) in the conversation cycle [1]. Such a system is displayed in Fig 1.

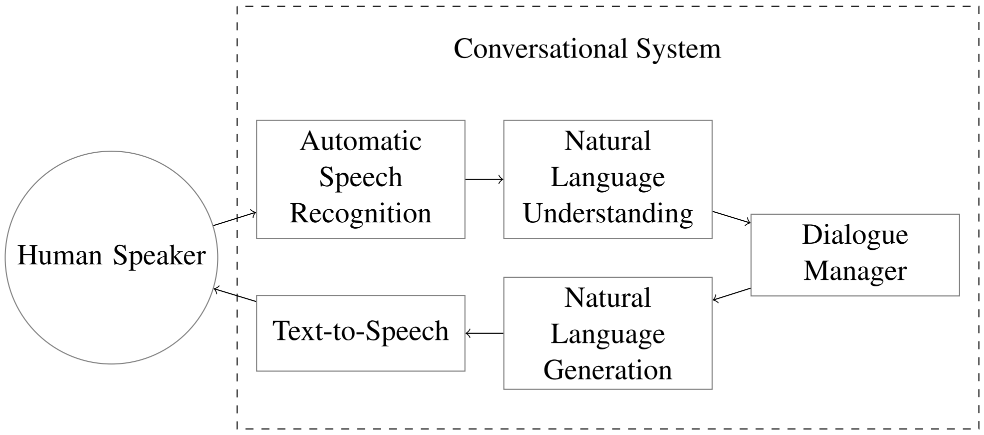
Conversational System
Human Speaker
Automatic Speech Recognition
Natural Language Understanding
Dialogue Manager
Natural Language Generation
Text-to-Speech

Figure 1: A diagrammatic representation of a conversational system [1]

The Speech Event module only handles one component of the conversational system described in the previous paragraph, that is, speech-to-text (which is referred to as automatic speech recognition in Fig 1). It acquires audio signals published on the `/soundDetection/signal` ROS topic, and then passes these audio signals through a deep neural network that transcribes speech utterances contained within the audio to generate text string representations, which are then published on the `/speechEvent/text` ROS topic. Kinyarwanda and English are the two languages that are transcribed by Speech Event, with the choice of language being set by the Behaviour Controller which reads it from the Culture Knowledge Base and then passes it to Speech Event via the `/speechEvent/set_language` ROS service that is advertised by Speech Event.

## 2 Requirements Definition

A running Speech Event ROS node performs one main function - Kinyarwanda and English speech recognition. An audio signal is received via the `/soundDetection/signal` ROS topic, and Speech Event transcribes any speech utterances detected in the signal to either Kinyarwanda or English.

In order to successfully perform this function, the following functional requirements are fulfilled by Speech Event:

1. Acquire an audio signal from the `/soundDetection/signal` ROS topic
2. Pass the audio signal through an automatic speech recognition (ASR) model to transcribe any speech utterances the audio signal contains to either Kinyarwanda or English text
3. Publish the transcribed text to the `/speechEvent/text` ROS topic

The exact language between Kinyarwanda and English, which a running Speech Event ROS node transcribes, is decided based on a configuration option that is set during the initialisation stage when the ROS node is started. The language that is set during the initialisation of the Speech Event ROS node can be overridden by invoking the `/speechEvent/set_language` ROS service and passing a different language to it.

To supplement the core functionality that Speech Event performs (Kinyarwanda and English speech recognition), the text transcriptions that Speech Event transcribes are displayed on a graphical interface. The graphical interface is used as a replacement for the terminal interface, especially in presentation settings where displaying text transcriptions on the terminal is impractical due to the small font size used by the terminal and the extra verbosity of the results displayed on the terminal. This graphical interface is enabled when the ROS node is configured to run in verbose mode.

A ROS node that emulates Sound Detection is also availed, and its main purpose is to showcase the working of Speech Event in isolation, separated from the rest of the CSSR4Africa system and from the Pepper robot. This ROS node captures audio from the computer on which Speech Event is running on, and publishes the captured audio to the same ROS topic that Sound Detection publishes to (`/soundDetection/signal`), therefore mimicking Sound Detection's operation of acquiring audio from Pepper and publishing it to the aforementioned ROS topic.

A ROS service is also be provided for enabling and disabling the transcription process at runtime. This ROS service `/speechEvent/set_enabled` is invoked by the Behaviour Controller to disable the transcription process when Pepper is speaking so as to avoid the transcription of Pepper's utterances, and then re-enabling the transcription process once Pepper is no longer speaking.

It is also worth noting that Sound Detection may fail, and since the Behaviour Controller does not directly interface with Sound Detection, the Speech Event ROS node bears the burden of signaling failures that may arise in Sound Detection to the Behaviour Controller. This is done by publishing the message 'Error: soundDetection is down' on the `/speechEvent/text` ROS topic.

## 3 Module Specification

A running Speech Event ROS node performs speech-to-text in real-time, acquiring an audio signal from the `/soundDetection/signal` ROS topic, generating a text description of the acquired audio, and then publishing transcribed text to the `/speechEvent/text` ROS topic as soon as speech utterances are detected in the audio signal. The data that is input to Speech Event, the transformation that this data undergoes, and the data that Speech Event outputs is described below:

1. The language that Speech Event is to be configured to operate in is set when the ROS node is started by setting a language configuration option
2. The language set for Speech Event to operate in when it is first started can be overridden by making a `/speechEvent/set_language` ROS service call and passing a different language
3. When the Speech Event ROS node is running, it acquires audio signals published by the Sound Detection ROS node on the `/soundDetection/signal` ROS topic
4. The acquired audio signals are then passed through an ASR model that transcribes speech utterances present within the audio signals to text strings
5. The text strings output by the ASR model are then published to the `/speechEvent/text` ROS topic for the Behaviour Controller to use
6. The transcription process can be enabled or disabled by invoking the `/speechEvent/set_enabled` ROS service

This process, and the position of Speech Event within the CSSR4Africa system architecture, is captured in Fig 2.

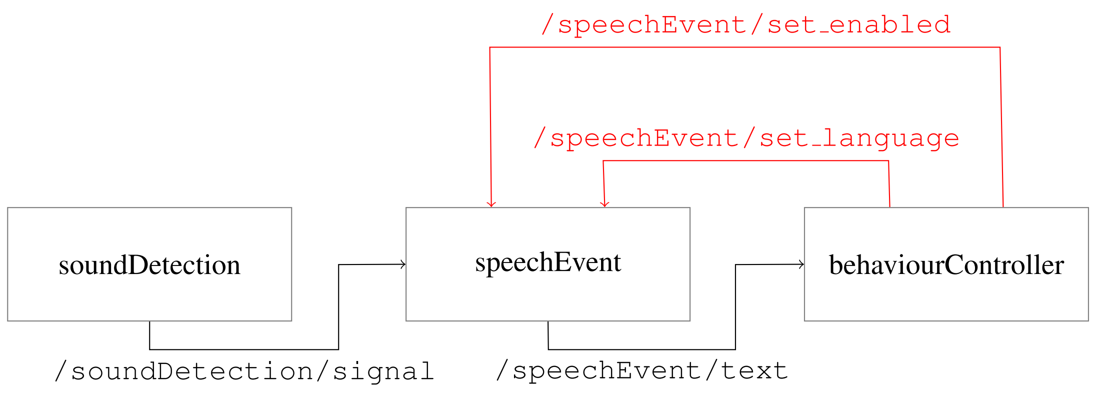

Figure 2: Speech Event within the CSSR4Africa architecture (`/soundDetection/signal` and `/speechEvent/text` are ROS topics, `/speechEvent/set_language` and `/speechEvent/set_enabled` are ROS services)

# 4 Interface Design

## 4.1 Source Code

The file structure of Speech Event is as shown below:

```
speech_event/
├── config/
│   └── speech_event_configuration.ini
├── data/
│   ├── audio_storage/
│   └── pepper_topics.dat
├── models/
│   ├── stt_en_conformer_transducer_large.nemo
│   └── stt_rw_conformer_transducer_large.nemo
├── src/
│   ├── speech_event_application.py
│   └── speech_event_implementation.py
├── srv/
│   ├── set_enabled.srv
│   └── set_language.srv
├── README.md
└── speech_event_requirements.txt
```

The `config/` directory houses configuration files that are used to configure different ROS nodes that are part of Speech Event. For Speech Event, only the `speech_event_configuration.ini` file exists in this directory, and it is used to configure the main Speech Event ROS node.

The `data/` directory contains data that is required by Speech Event when running. For Speech Event, the `pepper_topics.dat` file and `audio_storage/` sub-directory are contained in this directory:

1. `pepper_topics.dat`: list of ROS topics that Speech Event listens to when it is running
2. `audio_storage`: stores temporary audio files generated when Speech Event is transcribing speech utterances

The `models/` directory holds the two deep learning models used to transcribe speech utterances:

1. `stt_rw_conformer_transducer_large.nemo` for transcribing Kinyarwanda utterances
2. `stt_en_conformer_transducer_large.nemo` for transcribing English utterances

The `src/` directory holds all the Python source code files that implement the Speech Event system.

The `srv/` directory contains the `set_language.srv` service file specification for the ROS service `/speechEvent/set_language` and the `set_enabled.srv` service file specification for the ROS service `/speechEvent/set_enabled`.

The file `README.md` contains documentation of how to use Speech Event, and the Python requirements file `speech_event_requirements.txt` contains a list of versioned Python packages that Speech Event depends on.

## 4.2 Configuration Files

A Speech Event ROS node needs to be configured prior to starting it, a task that is accomplished via a configuration file. The configuration parameters are displayed in Table 1.

| Key | Value | Description |
|---|---|---|
| `language` | `kinyarwanda`, `english` | Specifies the language in which the utterance is spoken. |
| `verboseMode` | `true`, `false` | Specifies whether diagnostic data is to be printed to the terminal. |
| `cuda` | `true`, `false` | Specifies whether to use GPUs. The term ‘cuda’ is chosen as the key to alert the user that only NVIDIA GPUs are supported. |
| `confidence` | `<float>` | The confidence level on a scale of 0 to 1 above which transcriptions are assumed to be acceptable and correct. |
| `speechPausePeriod` | `<float>` | The time period above which one utterance is assumed to be separate from a preceding utterance. |
| `maxUtteranceLength` | `<integer>` | The maximum length (in seconds) of an utterance. Longer utterances are split when they go past this length. |
| `sampleRate` | `<integer>` | Specifies the sampling rate of the incoming audio sourced from the /soundDetection/signal ROS topic. |
| `heartbeatMsgPeriod` | `<integer>` | Specifies the time period in seconds at which a periodic heartbeat message is sent to the terminal. |

Table 1: Speech Event configuration file options

## 4.3 Data Files

The topics that Speech Event subscribes to are listed in the `pepper_topics.dat` file, as shown in Table 2.

| Key | Value |
|---|---|
| `soundDetection` | `/soundDetection/signal` |

Table 2: ROS topics that Speech Event subscribes to

## 4.4 Topics Subscribing To

A running Speech Event ROS node acquires audio signals from the `/soundDetection/signal` ROS topic. Table 3 shows this fact, while also mentioning the ROS node that publishes these audio signals.

| Topic | Node | Platform |
|---|---|---|
| `/soundDetection/signal` | `soundDetection` | `Physical robot` |

Table 3: Topics subscribing to

## 4.5 Topics Publishing To

The Speech Event ROS node publishes transcribed text to the `/speechEvent/text` ROS topic, publishing each time a transcription process completes. The published text transcriptions are of the `string` data type. Table 4 shows the ROS topic that Speech Event publishes to.

| Topic | Node | Platform |
|---|---|---|
| `/speechEvent/text` | `behaviourController` | `Physical robot` |

Table 4: Topics publishing to

## 4.6 Services Supported

A running Speech Event ROS node can have its language of operation (Kinyarwanda or English) updated by invoking the `/speechEvent/set_language` ROS service, or enabled or disabled at runtime by invoking the `/speechEvent/set_enabled` ROS service. Table 5 elaborates these ROS services.

| Service | Message Value | Effect |
|---|---|---|
| `/speechEvent/set_enabled` | `true`, `false` | Enable or disable the transcription process |
| `/speechEvent/set_language` | `kinyarwanda`, `english` | Set the language to either Kinyarwanda or English |

Table 5: Services supported

# 5 Module Design

## 5.1 Model Architecture

Both the Kinyarwanda and English speech recognition processes performed by Speech Event rely on deep learning models. Both of these deep learning models are based on the conformer transducer architecture, and the specific models that are used by Speech Event were acquired from NVIDIA's Nemo catalog of models.

The conformer transducer architecture is made up of a conformer encoder and a transducer decoder. The conformer encoder combines transformers and Convolution Neural Networks (CNNs) in an attempt to utilise the strengths of both, with transformers excelling at capturing global dependencies of an audio signal and CNNs excelling at capturing local dependencies of an audio signal [2]. The transducer decoder, on the other hand, makes use of Recurrent Neural Networks (RNNs) to form an autoregressive model whose next token prediction is conditioned on the previously predicted tokens.

### 5.1.1 Preprocessor

When an audio signal is received, it is first passed through a preprocessor block that extracts filterbank features from the audio signal. For this process, an audio signal is first converted to a spectrogram by a Short Time Fourier Transform (STFT), and then a series of filters is applied to the resultant spectrogram to obtain the filterbank features that each represent the magnitude of the energy within a distinct frequency band. This preprocessor block is visualised in Fig 3.

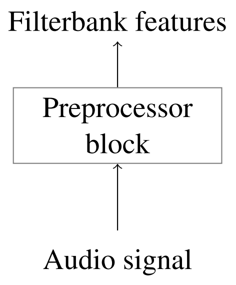

Figure 3: Audio to filterbank features preprocessor

### 5.1.2 Encoder Architecture

The filterbank features obtained from the preprocessor block are then passed to the encoder part of the model. The encoder transforms acoustic features in the original audio signal captured within the filterbank features into latent features that better represent the speech in the original audio signal.

A conformer encoder is shown in Fig 4. The encoder used in Speech Event has 17 conformer blocks.

The filterbank features are first passed through a SpecAugment block. SpecAugment is a data augmentation method that is suitable for end-to-end speech recognition models such as the Conformer Transducer model used by Speech Event. "The augmentation policy consists of warping the features, masking blocks of frequency channels, and masking blocks of time steps" [3].

The features are then processed by a convolution subsampling block, dropout applied, and then passed through a series of 17 conformer blocks in the encoder.

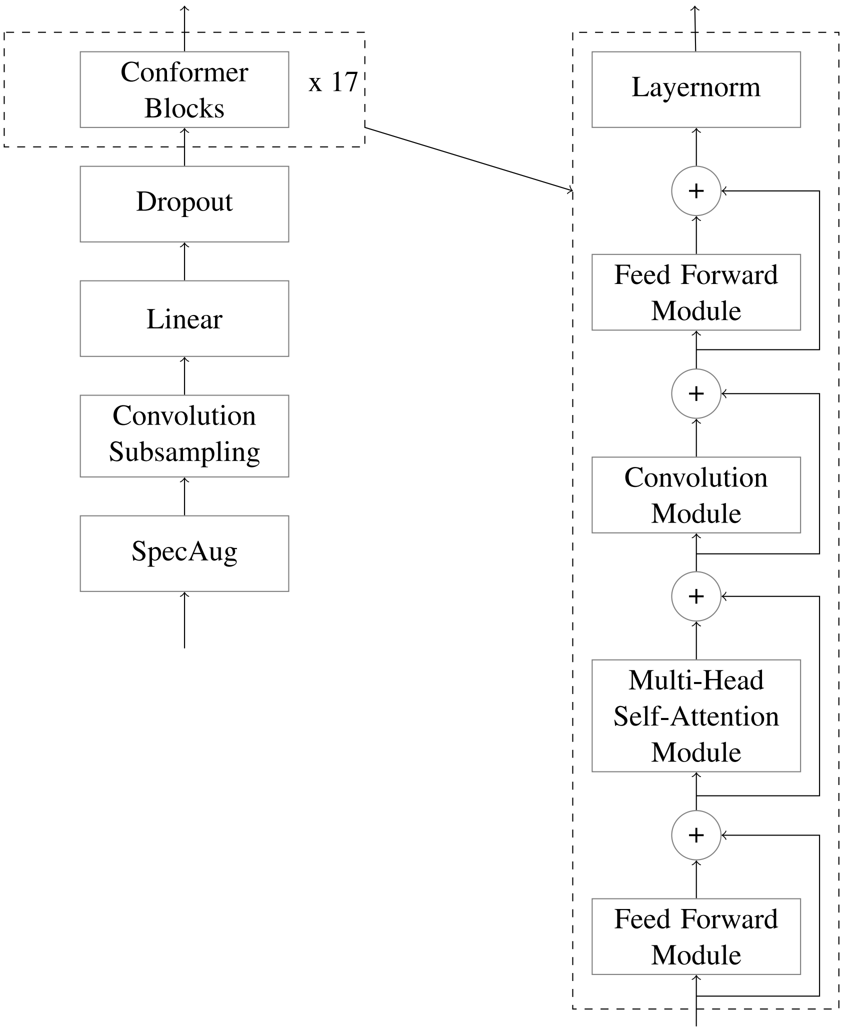

Figure 4: Conformer encoder architecture [2]

Each confomer block consists of a multi-head self-attention module and a convolution module collectively sandwiched between two feedforward modules [2]. The self-attention module is more adept at capturing global feature dependencies, while the convolution module is more adept at capturing local feature dependencies, thereby capturing the semantics of speech utterances much better than models that only specialise in capturing only one of either the global or local feature dependencies.

### 5.1.3 Decoder/Predictor Architecture

Keeping in line with Gulati et al., a single LSTM layer is used in the decoder [2]. An embedding layer converts input tokens into vectorised representations that are then passed to the LSTM layer. Additionally, drop out is applied both in the LSTM layer and on the output of the LSTM layer.

The decoder architecture used in Speech Event is shown in Fig 5.

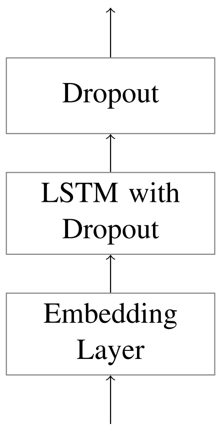

Figure 5: Transducer predictor architecture

#### 5.1.4 Joint Network Architecture

The joint network combines the output of the encoder and the output of the decoder in order to predict the text in the speech utterances contained in the original sound signal passed to the encoder. The combined outputs are then passed through a ReLU activation, dropout is applied, and lastly passed through a linear layer.

The joint network architecture used in Speech Event is shown in Fig 6.

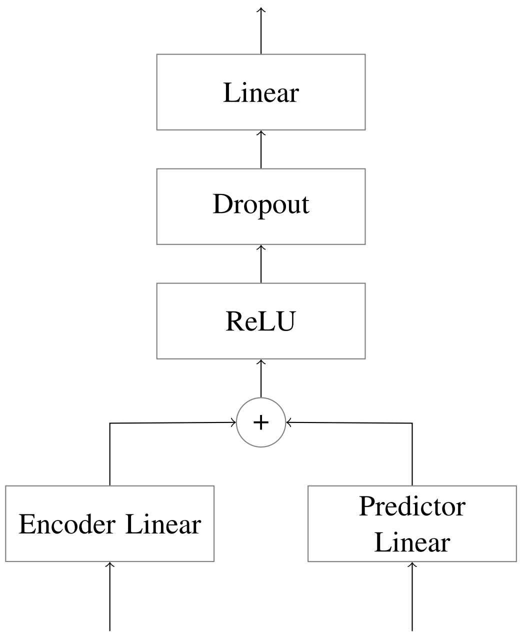

Figure 6: Joint architecture

### 5.1.5 Combined Model Architecture

The four different sections of the full conformer transducer architecture discussed above are combined together to form the whole network that performs end-to-end automatic speech recognition. This full model's architecture is summaried in Fig 7.

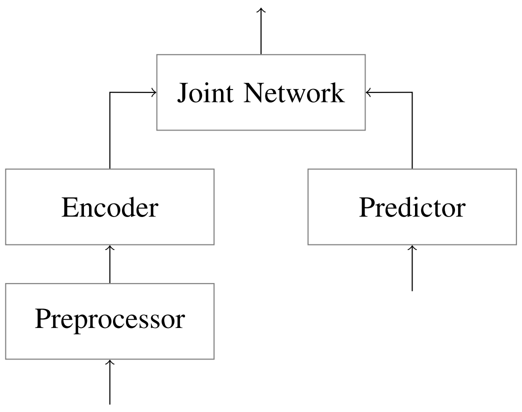

Figure 7: Combined model architecture

## 6 Testing

The `unit_tests` package contains unit tests for all CSSR4Africa ROS nodes, including Speech Event. Python's unittest testing framework is used for testing. The unittest package is part of the standard Python library, and therefore no extra packages need to be installed when running tests.

These tests check to ensure that Speech Event works as expected end-to-end, from when it receives a signal on the `/soundDetection/signal` ROS topic to when it publishes transcribed text on the `/speechEvent/text` ROS topic. They also check that the `/speechEvent/set_language` and `/speechEvent/set_enabled` ROS services work as expected.

A test report is generated in the `data/` directory, `speech_event_test_output.dat`. It shows all the tests that have been run and whether they have passed or failed. For tests that err, the reason for the error is also shown within the test report.

Before running tests, the ROS node that runs these tests has to be configured. The driver ROS node that emulates the Sound Detection ROS node also has to be configured, if the tests will not be run using the Pepper robot.

The configuration options for the ROS node that runs the tests are located in the configuration file `speech_event_test_configuration.ini`. They are shown in Table 6.

| Key | Value | Description |
|---|---|---|
| `heartbeatMsgPeriod` | `<integer>` | Specifies the time period in seconds at which a periodic heartbeat message is sent to the terminal. |
| `waitTimeout` | `<integer>` | Specifies how long to wait for speechEvent to start. |
| `mode` | `mic`, `file` | Use 'mic' when using real-time utterances from Pepper or the driver ROS node, and 'file' when using saved audio files. |

Table 6: Speech Event Test configuration file

When using a driver ROS node, the following is important when configuring the driver ROS node:

- Tweak the `speechAmplitudeThreshold` in case the driver is having trouble capturing speech utterances. Increase it in case speech is not being detected, and decrease it in case ambient noise is being captured together with the speech utterances. You may also tweak the sensitivity of the PC's microphones in case `sspeechAmplitudeThreshold` fails to solve the aforementioned issues that may arise.
- When using the driver to test Speech Event, ensure that `mode` in the driver configuration file matches `mode` in Speech Event Test configuration file (either set both to `file` or set both to `mic`). When using a running Sound Detection ROS node to test Speech Event, ensure that `mode` in the Speech Event Test configuration is set to `mic`.
- If using ROS launch files to launch the test harness, the argument passed to the `mode` parameter of the launch file will override whatever is set in the configuration files. Therefore when using the test harness launch file, no need to update the `mode` option in the configuration files, as setting it once while launching using the test harness launch file suffices.

The configuration options for the driver ROS node are located in the driver configuration file `speech_event_driver_configuration.ini`. They are shown in Table 7.

| Key | Value | Description |
|---|---|---|
| `channels` | `<integer>` | Number of audio channels to use when acquiring audio from PC's microphones. |
| `chunkSize` | `<integer>` | Number of samples to acquire at a go every time that audio is read from the PC's microphones. |
| `sampleRate` | `<integer>` | The sample rate to use when acquiring audio from the PC's microphones. |
| `speechAmplitudeThreshold` | `<float>` | Threshold above which a signal sample is assumed to contain a speech utterance. |
| `mode` | `mic`, `file` | Use 'mic' when using real-time utterances from Pepper or the driver ROS node, and 'file' when using saved audio files. |

Table 7: Speech Event Driver configuration file

The tests for Speech Event test the following:

1. The working of the `/speechEvent/set_enabled` ROS service
2. The working of the `/speechEvent/set_language` ROS service
3. The end-to-end working of Speech Event as a whole, from when an audio signal is received from the `/soundDetection/signal` ROS topic to when a text transcription is published on the `/speechEvent/text` ROS topic

A sample test report that is generated at the end of each testing process is displayed below (some lines have been artificially folded to the next line in this report to avoid overflowing out of the page):

```
Speech Event Test Report

Date: 2025-04-20 18:34:47


==================================================================

Test SpeechEvent's /speechEvent/set_enabled ROS service
---
As a:     behaviourController developer
I want:   a means of enabling and disabling the transcription process
          at runtime
So that:  I can stop the robot from transcribing its own speech by
          disabling the transcription process
```

```
---
Set unsupported status (agree): PASS
Set unsupported status (1): PASS
Set supported status (false): PASS
Set supported status (true): PASS

=====================================================================

Test SpeechEvent’s /speechEvent/set_language ROS service
---
As a:     behaviourController developer
I want:   a means of setting the transcription language of
          speechEvent at runtime
So that:  I don’t have to restart speechEvent every time I update the
          transcription language
---
Set unsupported language (Kiswahili): PASS
Set unsupported language (Arabic): PASS
Set supported language (English): PASS
Set supported language (Kinyarwanda): PASS

=====================================================================

Test SpeechEvent’s end-to-end transcription process
---
Given:    a running speechEvent ROS node
When:     an audio signal is detected on the /soundDetection/signal
          ROS topic
Then:     the audio needs to be transcribed and the text
          transcription published on the /speechEvent/text ROS topic
---

Transcribe ’rw-ingendo’: PASS
Transcribe ’rw-atandatu’: PASS
Transcribe ’rw-ibikenewe’: PASS
Transcribe ’rw-kabiri’: PASS
Transcribe ’en-turner construction was the construction manager
            for the project’: PASS
Transcribe ’en-he now resides in monte carlo’: PASS
Transcribe ’en-the council was held’: PASS
```

# 7 User Manual

## 7.1 Installation

The Speech Event ROS node needs to be installed before it can be used. To install Speech Event, clone the CSSR4Africa repository where it is housed into the CSSR4Africa ROS workspace, and then proceed with the following steps (the `README.md` file located within the Speech Event ROS node's directory of the cloned repository contains a similar but more elaborative list of installation steps):

1. Download ASR model files and move them to the correct directory (skip if you already have the ASR models in the correct directory):

```
# Install git-lfs
sudo apt-get update
sudo apt-get install git-lfs
git lfs install

# Clone CSSR4Africa models repository from Hugging Face
git clone https://huggingface.co/cssr4africa/cssr4africa_models

# Move speech event models to the models/ directory
mv cssr4africa_models/speech_event/models/* $HOME/workspace/
    ↪ pepper_rob_ws/src/cssr4africa/cssr_system/speech_event/models
```

2. Install required Linux tools:

```
# Install required Linux packages
sudo apt-get update
sudo apt-get install cython3 ffmpeg gfortran libopenblas-dev
    ↪ libopenblas64-dev patchelf pkg-config portaudio19-dev python3-
    ↪ testresources python3-tk python3-typing-extensions sox
```

3. Create a Python virtual environment and install required Python packages (Speech Event has been tested and proven to work using Python3.8):

```
# Create CSSR4Africa's virtual environments' parent directory if it
    ↪ doesn't exist
mkdir -p $HOME/workspace/pepper_rob_ws/src/cssr4africa_virtual_envs

# Change the current working directory to CSSR4Africa's virtual
    ↪ environments' parent directory
cd $HOME/workspace/pepper_rob_ws/src/cssr4africa_virtual_envs

# Create speech event's Python virtual environment
python3 -m venv cssr4africa_speech_event_env

# Source the created virtual environment
source cssr4africa_speech_event_env/bin/activate

# Install required Python dependencies
pip install -r $HOME/workspace/pepper_rob_ws/src/cssr4africa/
    ↪ cssr_system/speech_event/speech_event_requirements.txt
```

4. Build the ROS workspace (it's best to proceed with the next steps on a separate terminal, otherwise an error may arise when running 'catkin_make'):

```
# Change the current working directory to CSSR4Africa's ROS
    ↪ workspace
cd $HOME/workspace/pepper_rob_ws

# Build the CSSR4Africa project
catkin_make

# Source the CSSR4Africa ROS workspace
source devel/setup.bash
```

5. Make application files executable:

```
# Add executable permission flag to speech event's application file
chmod +x $HOME/workspace/pepper_rob_ws/src/cssr4africa/cssr_system/
    ↪ speech_event/src/speech_event_application.py

# Add executable permission flag to speech event's test application
    ↪ file
chmod +x $HOME/workspace/pepper_rob_ws/src/cssr4africa/unit_tests/
    ↪ speech_event_test/src/speech_event_test_application.py

# Add executable permission flag to speech event's driver file
chmod +x $HOME/workspace/pepper_rob_ws/src/cssr4africa/unit_tests/
    ↪ speech_event_test/src/speech_event_driver.py
```

## 7.2 Usage

To run a Speech Event ROS node, a Sound Detection ROS node needs to also be running (or a Speech Event driver node, which is described in a subsequent subsection).

1. Run a Speech Event ROS node: `rosrun speech_event speech_event_application.`
   `↪ py`

2. View text transcriptions on the terminal (optional): `rostopic echo /speechEvent/text`

If running the Speech Event ROS node in verbose mode (by setting the configuration option `verboseMode` to `true`), a graphical application that displays the text transcriptions will open on the screen.

## 7.3 Driver ROS Node

A driver ROS node that mimics the Sound Detection ROS node is also provided. It capturing audio from a personal computer's microphone instead of from Pepper, and publishes the audio signal on the `/soundDetection/signal` ROS topic in the same way that the Sound Detection ROS node does. To bring up this driver ROS node: rosrun speech_event speech_event_driver.py.

## References

[1] Cristina Romero-González, Jesus Martínez-Gómez, and Ismael García-Varea. Spoken language understanding for social robotics. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 152–157, April 2020.

[2] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Interspeech 2020*, pages 5036–5040. ISCA, October 2020.

[3] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Interspeech 2019*, pages 2613–2617. ISCA, September 2019.

## Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Clifford Onyonka, Carnegie Mellon University Africa.
David Vernon, Carnegie Mellon University Africa.
Richard Muhirwa, Carnegie Mellon University Africa.

## Document History

**Version 1.0**

First draft.
Clifford Onyonka.
20 February 2025.

**Version 1.1**

Added a ROS service that will be used by the Behaviour Controller to set the language (Kinyarwanda or English) that Speech Event will operate in.
Clifford Onyonka.
13 March 2025.

**Version 1.2**

Updated configuration file keys to camel case from snake case
Added the ‘confidence’, ‘speechPausePeriod’, and ‘maxUtteranceLength’ keys to the Speech Event ROS node’s configuration file
Added a ROS service that will be used by the Behaviour Controller to enable or disable Speech Event’s transcription process
Updated the testing section to use use a test ROS node to run tests
Updated the user manual section to detail usage of Speech Event using a Python virtual environment and steps for downloading ASR models from Hugging Face
Clifford Onyonka.
11 June 2025.

**Version 1.3**

Fixed typing errors.
Use listings for code blocks.
Clifford Onyonka.
30 July 2025.