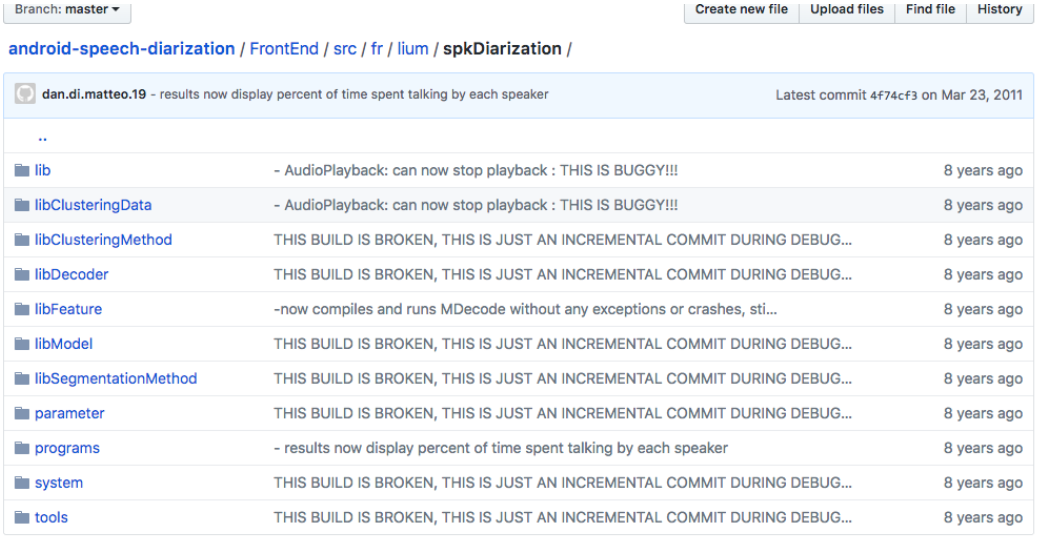
Developers Guide

In this developers guide we touch on two of the main classes recordingActivity and summaryActivity. In this guide we touch on the methods which may be used most for making future updates on the app.
Note for the algorithm Lium Library, potentially the library version we are using is broken as the tutorial github has broken builds


Branch: master ▾ Create new file Upload files Find file History

android-speech-diarization / FrontEnd / src / fr / lium / spkDiarization /

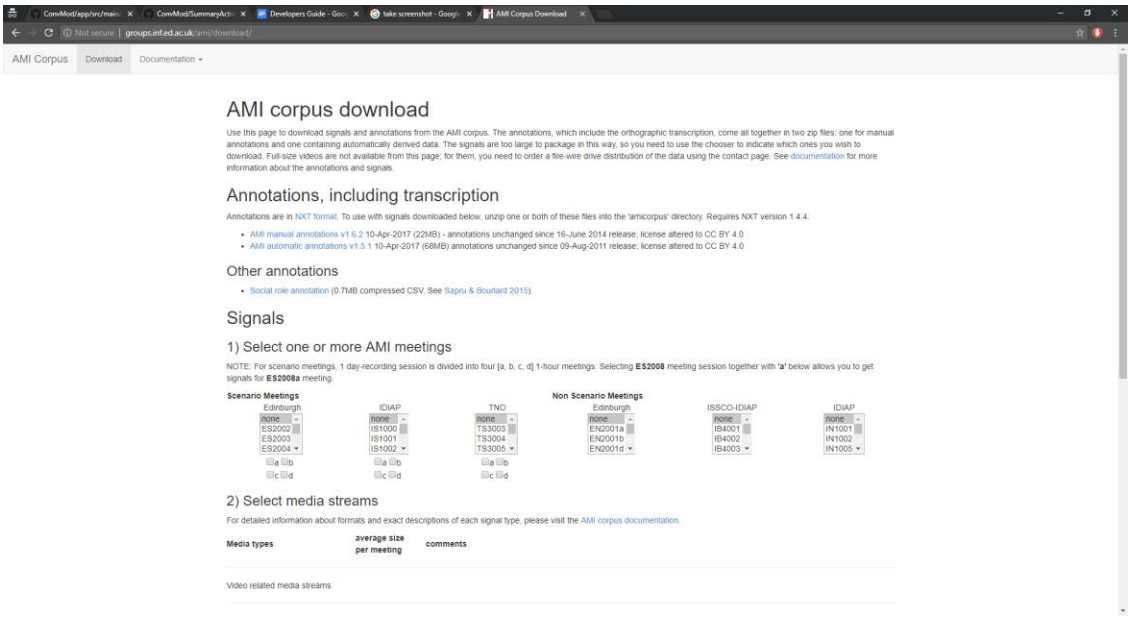
dan.di.matteo.19 - results now display percent of time spent talking by each speaker Latest commit 4f74cf3 on Mar 23, 2011

..		
lib	- AudioPlayback: can now stop playback : THIS IS BUGGY!!!	8 years ago
libClusteringData	- AudioPlayback: can now stop playback : THIS IS BUGGY!!!	8 years ago
libClusteringMethod	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago
libDecoder	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago
libFeature	-now compiles and runs MDecode without any exceptions or crashes, sti...	8 years ago
libModel	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago
libSegmentationMethod	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago
parameter	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago
programs	- results now display percent of time spent talking by each speaker	8 years ago
system	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago
tools	THIS BUILD IS BROKEN, THIS IS JUST AN INCREMENTAL COMMIT DURING DEBUG...	8 years ago

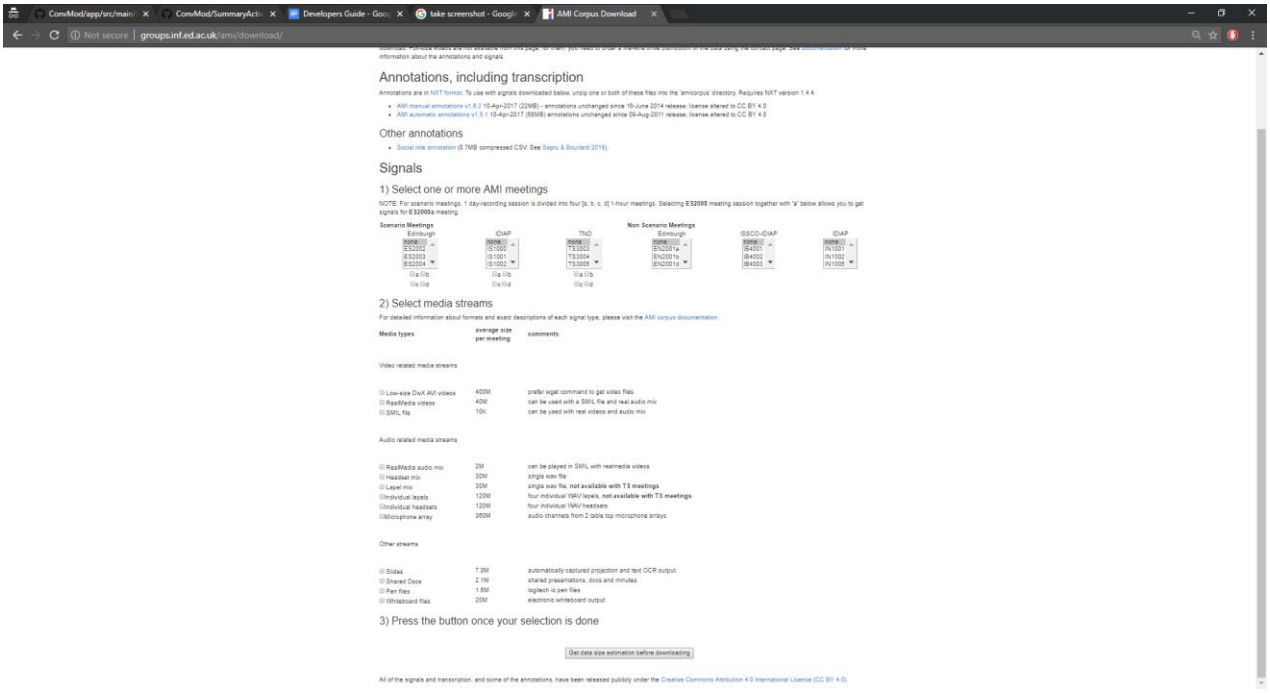
For testing data using prerecorded data vs using the app for as a recording tool

For testing data-

- In the recording activity class there should be an initial variable that says testingMode and parseData. Change these to true. There is also a variable called testingFile. Add the wav file name you want to test- I think right now its set to a file called ES2003a.Mix-Headset.wav.
- Note for any files you want to add, just add them to the assets folder in app/src/main/assets
- Now in the Summary activity class find the parseAnnotations method and find a line of code that says if(assets.contains and as the parameter change whatever you wav file name is BUT DO NOT ADD THE .WAV or whatever extension. Just the name like ES2003aMix-Headest NOT ES2003aMix-Headset.wav. This is because it will also search the xml files with same reg ex with this name and parse through the marked up annotations files which were marked up to describe the actual time spoken.
- Please note that the style of marked up annotations come from http://groups.inf.ed.ac.uk/ami/download/ this site. Click on one of the annotation links either manual or automatic



- Now select one of the meetings and choose headset mix. Then click download. Please note the app has some optimization issues where a certain length of audio or size may not work. I believe the audio file size that is working is 25 minutes. Anything over may or may not work for diarization. Also Parse annotations only works for this sites



annotations. The annotations of other corpuses may not work in parse annotations method.

- Now to run the app just click finish on the app screen. there is no need to record anything. Once diarization is finished it should take you to summary activity page BUT IT

WILL BE GLITCHY and not really show much except the piano roll. Instead you have to go to your adb debugger and type in
    - ./adb logcat diarization:* *:S
- To view the results. The method responsible for data testing is dataComparison()

For regular app usage-

All you need to do is set testingMode to false and parseData to false. And run the app.

Class recordingActivity:

Initial variables-

- Boolean testingMode:
    - determines whether you want to diarize newly recorded audio or use a prerecorded corpus
- Boolean parseData:
    - checks whether the corpus file comes with data files to parse
- String testingFileName:
    - select the name of the sound file you wish to diarize if testing mode is true. If parseData is true, this filename will also be used to select the files to diarize

Tutorial:

Initial variables :

- Boolean tutorialMode:
    - when the user selects tutorial from either the menu or the dialogue box, the variable becomes true. This calls the startTutorial() method which creates the first popup bubble
- Int tutorialNumber:
    - When first initialized, the tutorialNumber is set as 1. However, when tutorial is run, the number changes to allow the user to only click on a certain button during the interactive tutorial. See startTutorial(), pauseTutorial(), and clickRecordTutorial()

Tutorial Methods:

- void TutorialManager(Boolean firstRecord, SharedPrefrences.Editor editor):
    - Determines whether or not it is the first time using the app. If so set up the tutorial dialogue box to see if one should run the tutorial.

- Void startTutorial():
    - If user runs the tutorial this initializes the tutorial and starts changing the tutorialNumber variable as one follows along. It displays the tooltips which are

speech bubbles which place themselves above the buttons the user is to click during the tutorial.

- Boolean clickRecordTutorial(), pauseTutorial(), resetTutorial(), finishTutorial():
    - These methods are placed into the respective button method such as clickRecord(), pause(), etc and decide whether the user is allowed to click on that button during the tutorial based on the corresponding tutorialNumber. If not it tells the outer method if not ready to click (it does this by being true).

Methods which interact with activity_record xml interface objects:

- Void clickRecord():
    - Corresponds to the button_Record xml ID. Starts recording if clicked

- Void clickPause():
    - Corresponds to button_pause xml ID object. If clicked, stops the app from recording.

- Void finish():
    - Corresponds to button_finish xml ID object. If clicked, calls the diarize() method and then creates intents used to pass 4 variable states to summary Activity which will decide the behavior of summaryActivity. The variables sent are tutorialMode, testingMode, parseData, and testingFileName.

Algorithm:

- Void Diarize():
    - Method which houses the algorithm for diarization. Calls copyAssets() which copies any files from the assets folder into the internal app storage. This is used when in testingmode If one wants to access a wav file or data parsing file to diarize.
    - The method then decides if testingMode is on. If it is it gets the testingFileName and extracts those audio features. If testingmode is off, it gets the sound features from AudioEventProcessor.RECORDER_RAW_FILENAME (the in app recorded file).
    - The method then sends the features to a .mfc file using outstream to the filepath getFilesDir()+"/"+AudioEventProcessor.RECORDER_FILENAME_NO_EXTENSION+ ".mfc". With the mfc file created it can be inputted into the Lium Library.

- To see how the library should be called view link below. http://www-lium.univ-lemans.fr/diarization/doku.php/single-show_diarization

- Note: in our app, we were unable to run the first MDecode, the sFilter and the SSplitSeg. The tools which are called using the lium frameworks methods are housed in fr.lium/spkdiarization/tools

- Though each method has parameters which differ from one another slightly we describe the basic parameters here.
    - fInputMask- this is always basePathName+".mfc" as it is the features input mask
    - The sInput mask is the segmentation file which was the soutput mask of the previous method.
    - The soutputMask is the name of the segmentation file to be output. All diarization methods save their sOutputMask file to the internal storage stored under getFilesDir()+"/"+AudioEventProcessor.RECORDER_FILENAME_NO_EXTENSION+"x .seg"
    - These files are accessed in summaryActivity

```
997     }
998     try{
999         SAdjSeg.main(new String[] { "--trace","--help",
1000                "--fInputDesc=sphinx,1:1:0:0:0:0,13,0:0:0",
1001                "--fInputMask=" + basePathName + ".mfc",
1002                "--sInputMask=" + basePathName + ".d.seg",
1003                "--sOutputMask=" + basePathName + ".adj.seg", AudioEventProcessor.RECORDER_RAW_FILENAME});
1004    }
1005    catch (Exception e) {
1006        // TODO Auto-generated catch block
1007        e.printStackTrace();
1008    }
1009    try{
1010        SAdjSeg.main(new String[] { "--trace", "--help", "--sGender", "--sByCluster",
1011                "--fInputDesc=sphinx,1:3:2:0:0:0,13,1:1:0",
1012                "--fInputMask=" + basePathName + ".mfc",
1013                "--sInputMask=" +basePathName + ".adj.seg",
1014                "--sOutputMask=" + basePathName + ".g.seg",
1015                "--tInputMask=" + genderModel.getAbsolutePath(),
1016                AudioEventProcessor.RECORDER_RAW_FILENAME});
1017    }
1018    catch (Exception e) {
```

Class summaryActivity:

This activity is called after diarization has finished.
When the activity starts, the methods called in sequence from the onCreate then the onResume method. The xml which corresponds to this activity is the activity_summary.xml file

Intent variables-

- Passed in from recordingActivity is testingMode, parseData, testingFileName, and tutorialMode.

Accessing segmentation files-

- ArrayList<Speaker> speakerSegmentFiles():
- While we opened all the segmentation files the only one we really need is the last sOutput file from the the diarize method in recordingActivity. We then pass the filename to a file input stream and return speakerArray =new SpeakersBuilder().parseSegStream(inputStreamName).build(); this speaker array will then be used if testingmode is false the pie chart and piano graph .

Working with SpeakerArray-

- The speaker array contains properties about every speaker including the time they started speaking, and the time each talking streak lasts.
- If testing mode is false, the speakerArray will be sent to the piano roll creator and pie chart creator
- If testingMode is passed in as true:
    - To save memory and time no graphs are displayed on the interface. The dataComparison() method will be run and results will be displayed by going to the folder where your adb debugger is downloaded and running the command
      ./adb logcat results:* *:S
      after the summaryActivity activity finishes loading
      This will display the results in the logcat.

Methods used for testingMode:

- Void Datacomparison()
    - In this method we parse the corpus xml files and compare it with the speaker.get start times. As both the corpus xml files and the speaker segmentation files give start times and end time of a period when a person speaks, we do a riemann sum of .1 second intervals and compare for a subset between what the algorithm detected per person and what the actual speaking time was. The time wrong is then calculated and divided over the total time of the conversation file to determine the diarization error percentage.

Methods run exclusive to testingMode being false:

<u>Tutorial:</u>
Void startTutorial():

- The tutorial in this activity is self contained in a single method called startTutorial(). This method gives a quick description of all the graphs and buttons on the interface.

Methods which relate to the interface.

- Void createPieGraph(Speaker speaker)
    - This method corresponds to the mikephilpiechart xml framework object. In this method, it displays the percent time spoken per speaker.

- Void createPianoRollBar(Speaker speaker):
    - This method corresponds to the piano graph xml id object. This method gets the start times for each speaker and calculates the length relative to the width of the screen and draws the piano roll

- From library of android waveform visualizer: method call
    - getSupportFragmentManager().beginTransaction()
      .add(R.id.*soundWaveContainer*, new CustomWaveformFragment()).commit();
      calls the sound wave visualizer and corresponds to the xml id soundwavecontainer

Other Methods one may want to modify

- Other major classes which are part of the app are housed in com/loyola.talkertracer/model/audiorecord which contains classes used for building speakers, timer management and other in app objects.
- For modifying the Lium Library, the tools are housed in fr.lium/spkDiarization/tools/