Gailbot Documentation

About

Gailbot is an automated transcription software named after [Gail Jefferson](https://en.wikipedia.org/wiki/Gail_Jefferson), inventor of the transcription system used for [Conversation Analysis](https://en.wikipedia.org/wiki/Conversation_analysis).

It is designed to be primarily used for generating Conversation Analysis transcripts that can then be improved manually.

**\*\***Please see the [**liability notice**](#Liability)**.**

Overview

Gailbot either takes in recorded speech or records a new dialogue and uses a Speech to Text API to generate a time-aligned transcript. It uses a system of modules to detect conversation speech rate, analyze laughter, and to annotate structural features of conversation (pauses, gaps, overlaps etc.)

Gailbot renders the final transcript in JSON, CSV, and in time-aligned transcription formats including [CHAT](https://talkbank.org/manuals/CHAT.html), [TalkBank XML](https://talkbank.org/software/xsddoc/), and the Jeffersonian [CAlite](https://github.com/saulalbert/CABNC/wiki/CHAT-CA-lite) format developed at the Human Interaction Lab.

Status / Contribute

**VERSION: 0.3.0**

This version of Gailbot has been tested on OSX Mojave 10.14.5.

This is an alpha version.

Users are encouraged to provide feedback, details regarding bugs, and development ideas.

Send an email to: [hilab-dev@elist.tufts.edu](mailto:hilab-dev@elist.tufts.edu)

Installation

**Pre-requisites**

In order to use Gailbot, you should have some familiarity with using the terminal to install and run software.

You should also be aware that Gailbot uses [IBM Watson's STT API](https://cloud.ibm.com/apidocs/speech-to-text) that includes transcription charges and requires that the user create an IBM Bluemix account.

**Installation Steps**

1. Install [Python 3.7](https://www.python.org/downloads/release/python-373/)
2. Install the [Homebrew](https://brew.sh/) packaging software:

mkdir homebrew && curl -L https://github.com/Homebrew/brew/tarball/master | tar xz --strip 1 -C homebrew

1. Install the pip3 package installer:

Brew install python3

1. Download and install the [CLAN editor](http://dali.talkbank.org/clan/) and [CAfont](http://dali.talkbank.org/clan/CAfont.otf) from [Talkbank](https://talkbank.org/software/).
2. Navigate to the Gailbot directory and use the 'requirements.txt' file to install all libraries:

Pip3 install -r requirements.txt

1. Create an account with IBM so you can use Watson's speech-to-text service
   1. You can sign up for a trial account [here](http://console.bluemix.net/catalog/services/speech-to-text).
   2. **NOTE:** Your IBM Bluemix username and password are required to establish a connection with Watson's Speech to Text service. Once you have registered you can find your credentials [here](ttps://console.us-east.bluemix.net/developer/watson/existing-services).
   3. Find transcription pricing [here](https://www.ibm.com/cloud/watson-speech-to-text/pricing).

Usage

Gailbot can be executed using the following command from inside the Gailbot directory:

Python3 gailbot-3.py -username [IBM Bluemix Username] -password [IBM Bluemix Password]

Gailbot can be used to perform three primary functions:

1. Transcribe an existing video or audio file.
2. Record and transcribe a new variable length conversation.
3. Re-apply post-processing algorithms to existing Gailbot transcript files and data without the need to re-transcribe the entire conversation using the (metered) Speech to Text service.

**Supported media formats**

Gailbot supports a number of audio and video file formats:

**AUDIO FORMATS**

|  |  |
| --- | --- |
| **Audio Format** | **Extension** |
| Audio/alaw | alaw |
| Ausio/basic | basic |
| Audio/flac | flac |
| Audio/g729 | g729 |
| Audio/l16 | pcm |
| Audio/mp3 | mp3 |
| Audio/mpeg | mpeg |
| Audio/mulaw | ulaw |
| Audio/ogg | opus |
| Audio/wav | wav |
| Audio/webm | webm |

**VIDEO FORMATS**

|  |  |  |
| --- | --- | --- |
| **Video Format** | **Extension** | **Min. Required channels** |
| Material exchange format | mxf | 2 |
| Quicktime file format | mov | 1 |
| MPEG-4 | mp4 | 1 |
| Windows media file | wmv | 1 |
| Flash video format | flv | 1 |
| Audio video interleave | avi | 1 |
| Shockwave flash | swf | 1 |
| Apple MPEG-4 | m4v | 1 |

**NOTE:** All video formats have a required minimum number of audio channels that the media file must have in order to be processed. This allows source separation to occur for multiple speakers.

**Transcribing existing conversations**

Gailbot allows users to transcribe all video / audio files that are supported while allowing a large level of customization. This allows users to tweak custom thresholds for different statistical models, CHAT files, and machine learning algorithms using a user-friendly command line interface.

A simple breakdown of the process is provided below. A more-detailed technical description is provided later on.

**STEP 1**: Select all files.

* There are multiple ways to add supported files to Gailbot:
  + Add files as an individual file using the name of the file.

Sample1.mp3

* + Add two pair files part of the same conversation using the **‘-pair’** flag. Note that this flag can be used multiple times in the same instance.
  + **NOTE:** Pair files are files that are conversations recorded with separate audio channels for each speaker.

-pair [file1] [file2]

* + Add multiple files as separate files by placing them in a unique directory and using the **‘-dir’** flag as follows:

-dir [Directory Name]

**STEP 2:** Select [post-processing modules](#postProcessing) to be applied

* Gailbot allows the user to select the individual post-processing modules that will be applied to all files added as part of a single instance.
* By default, all post-processing modules are selected.
* This feature enables isolated testing of phenomenon that may be identified using only a single module or a specific combination of modules. Additionally, it allows user to a straightforward way to add custom developed post-processing modules to Gailbot.
* **NOTE:** Gailbot automatically calls customization menus for all post-processing modules added after the selection process.

**STEP 3:** Customize [post-processing modules](#postProcessing)

* Once all post-processing modules have been selected, the user can customize these modules using their in-built and separate customization menus.
* **NOTE:** The user will only be able to customize modules that are selected to be applied.

**STEP 4:** Finalize and transcribe

* With all files added, post-processing modules selected, and their custom values defined, the user can finalize and send their transcription request to [Watson’s STT API](https://cloud.ibm.com/apidocs/speech-to-text).
* All added files, their directories, content type, speaker names, and number of speakers are displayed here.
* **NOTE:** Files that have the same output directory are part of a **file pair.**
* Here, the user can add [**custom language and acoustic models**](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-languageCreate).
* The user can choose to opt out of Watson’s default [data recording mechanism](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-summary#summary-x-watson-authorization-token) that allows IBM’s Watson to improve its own transcription model. By default, the user has **opted out** of this service.
* Additionally, the user can select the method to interact with Watson: [Access or Watson authentication tokens](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-summary#summary-watson-token). Note that this feature does not alter the functionality of Gailbot in any way.
* Finally, the user can [change the weight](#Glossary) attributed to the custom model if used in the transcription process.

**Recording and Transcribing new conversations**

Gailbot allows users to record and transcribe new conversations on the go.

This allows users to test out Gailbot capabilities or added custom models for conversations that they are currently having and reduced the need to have a high-quality audio recording system for corpus generation.

Gailbot is able to achieve this using the Speech-to-text service’s [dialogue model](#Glossary).

The process for using this feature is the same as the process for transcribing an existing conversation defined above. However, an additional recording menu is presented that allows the user to specify different features of the recorded conversation including bit rate, channels etc.

**NOTE:** The recorded conversation is saved in the current directory as **‘Recorded.wav’**.

**Re-applying post-processing modules**

In some cases, the user may have already run a media file through Gailbot but have the need to reproduce the transcript after modifying certain post-processing variables. This feature allows quick reproduction of files by only applying post-processing modules without sending requests to the Watson transcription service.

**NOTE:** This feature requires the original unchanged result directory produced by Gailbot.

This following is a list of files needed in the generated directory to use this feature:

|  |  |
| --- | --- |
| **File** | **Status** |
| Individual audio file | **REQUIRED** |
| Combined audio file | **REQUIRED** |
| JSON files | **REQUIRED** |
| CSV files | **NOT REQUIRED** |
| CHAT / CA files | **NOT REQUIRED** |

**\*\*NOTE:** In order to use this feature with [**pair files**](#Glossary)that are part of the same conversation, the user has to go through the process of adding files **twice**, once for each of the individual file part of the conversation. This is because Gailbot uses directory name to identify files that are part of the same conversation.

**Narrowband and Broadband files**

A characteristic of all media files is the [**bit/sample-rate**](#Glossary)i.e. the number of bits processed per unit of time during the file generation process.

By default, Gailbot uses one of a selection of [base speech models](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-models#modelsList) to transcribe media files. These base models are classified as **narrowband** or **broadband** depending on the sampling rate of the audio file:

* Narrowband models
  + Gailbot models used for audio files sampled at 8KHz, typically used for phone calls.
* Broadband models
  + Gailbot models used for audio files sampled at 16KHz or greater, more usual for high quality video and audio recordings.

Note that in order for Gailbot to function, users must select the correct **type** of base model for the files being processed. Gailbot runs checks on media files and will only send data to the Speech-to-text API if the sampling rate of the file matches the narrowband or broadband base model selected.

**\*\*NOTE:** Users can check the bit rate of a media file by right clicking on it and viewing its properties.

Base models can be modified using the [**Pre-request menu**](#Glossary).

**\*\*NOTE:** The base model in this case is the base model for both the **custom language model** and the **custom acoustic model** (defined below)**.** A custom acoustic model must be trained in cases where it does not exist for a specific base model.

Special Features and Post-processing

Gailbot’s post-processing modules take the verbatim transcript data produced by the Speech-to-text API and apply user-defined heuristics, statistical models, machine learning models and neural networks to add a range of structural features of conversation to the final transcription files. This extensible set of features currently includes turn-taking, silences, laughter, speech rate, and overlaps.

Gailbot’s post-processing interface is designed to provide a user-friendly command line interface to manage all post-processing modules and is scalable to allow users to add custom modules.

Additionally, Gailbot provides a number of special features that enhance user-experience and provide additional functionality.

**Custom language and acoustic models**

As described in the ‘Narrowband and Broadband files’ section, Gailbot uses different base models to process transcription requests. However, users can train and use [custom language and acoustic models](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-languageCreate) to enhance Gailbot’s transcription capabilities on top of the [base transcription model](#Glossary).

The custom models provide the following functionality:

|  |  |  |  |
| --- | --- | --- | --- |
| **Functionality** | **Details** | **Language Model** | **Acoustic Model** |
| Enhances transcription quality. | Extracts and adds new custom words to existing dictionary. | **YES** | **NO** |
| Enhances transcription quality | Filters background audio based on training data | **NO** | **YES** |
| Uses base model | Trained on top of a base model | **YES** | **YES** |
| Re-trainable | Existing models can be retrained using new data | **YES** | **YES** |
| Resettable | Can be reset to remove all custom data | **YES** | **YES** |
| Upgradable | Base model can be upgraded after creation | **YES** | **YES** |
| Lists dictionary words | Shows all words learned using training data | **YES** | **NO** |
| Lists audio sources | Shows audio files used to train model | **NO** | **YES** |
| Adjustable weight | Weight applied to custom model vs. base model is adjustable | **YES** | **NO** |
| Required | Necessary to use custom model to process request | **NO** | **NO** |
| Model type same as content type | The model type (broadband or narrowband) must be the same as the content | **YES** | **YES** |

**TRAINING**

|  |  |
| --- | --- |
| **Custom Model** | **Training Details** |
| Language Model | * Trained using a [custom corpus file](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-languageCreate#addCorpus) or using [individual words](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-languageCreate#addWords). |
| Acoustic Model | * Trained using a [‘.wav’ audio file](https://cloud.ibm.com/docs/services/speech-to-text?topic=speech-to-text-manageAudio#listAudio) that is longer than 10 mins and less than 20 hours in length. |

**Multi-Language Transcription**

By default, Gailbot uses the ‘en-US-Broadband’ base transcription model to process requests.

In addition to this base model, Gailbot provides base models for different languages defined below.

Users can change the base model by changing the base model defined in the **custom language model interface.**

|  |  |
| --- | --- |
| **Model Name** | **Language** |
| Pt-Br-NarrowbandModel / pt-Br-BroadbandModel | Portuguese |
| Ko-Kr-NarrowbandModel / ko-Kr-BroadbandModel | Korean |
| fr-FR-NarrowbandModel / fr-FR-BroadbandModel | French |
| En-US-NarrowbandModel / en-US-BroadbandModel | US English |
| Zh-CN-NarrowbandModel / zh-CN-BroadbandModel | Mandarin |
| Ja-JP-NarrowbandModel / ja-JP-BroadbandModel | Japanese |
| En-GB-NarrowbandModel / en-GB-BroadbandModel | British English |
| En-ES-NarrowbandModel / en-ES-BroadbandModel | Spanish |
| Ar-AR-NarrowbandModel / ar-AR-BroadbandModel | Arabic |
| De-DE-NarrowbandModel / de-DE-BroadbandModel | German |

**\*\*NOTE:** Some post-processing modules may not function or function improperly when used with languages other than English.

**Speech Rate Detection**

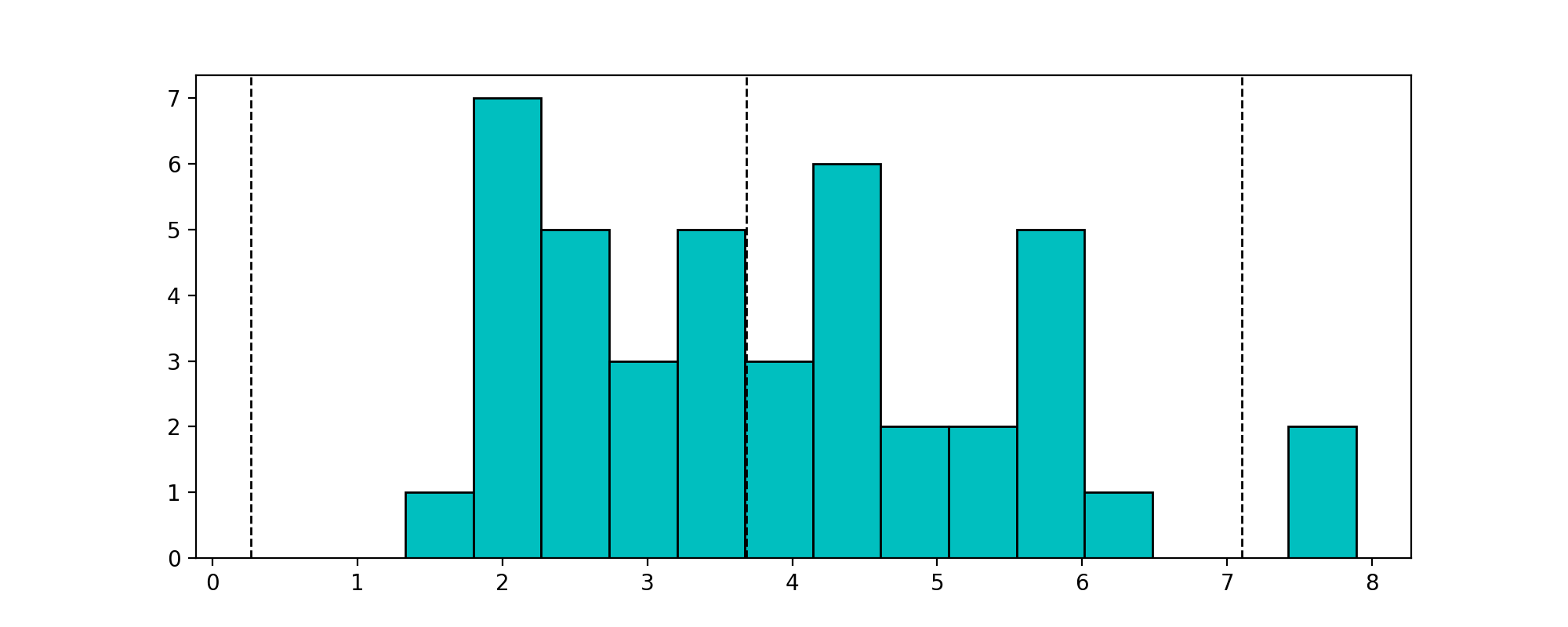
Gailbot uses a statistical model developed at the [Tufts Human Interaction Lab](https://sites.tufts.edu/hilab/) that classifies a [turn construction unit (TCU)](https://en.wikipedia.org/wiki/Turn_construction_unit) as either a fast turn or slow turn depending on the average speech rate of each individual speaker.

This allows Gailbot to mark sound stretches, as well as areas of faster and slower-than-surrounding speech using markers defined as part of the CHAT and [CAlite format](https://github.com/saulalbert/CABNC/wiki/CHAT-CA-lite)s.

Below is a brief overview of the speech rate detection model:

* The speech rate is the number of syllables in a TCU divided by the total time of the TCU.
* The number of syllables is calculated using a syllable dictionary for known and machine learning algorithm for unknown words.
* The median syllable rate for each speaker is separately calculated.
* The [median absolute deviation](https://en.wikipedia.org/wiki/Median_absolute_deviation) is calculated for each speaker.
* Any TCU having a syllable rate greater than Median + (2 \* Median absolute deviation) is classified as a fast TCU while any TCU having a syllable rate less than Median – (2 \* Median absolute deviation) is classified as a slow TCU.

The model can be visualized for a sample conversation as follows:



Syllable rate per turn

No of Turns

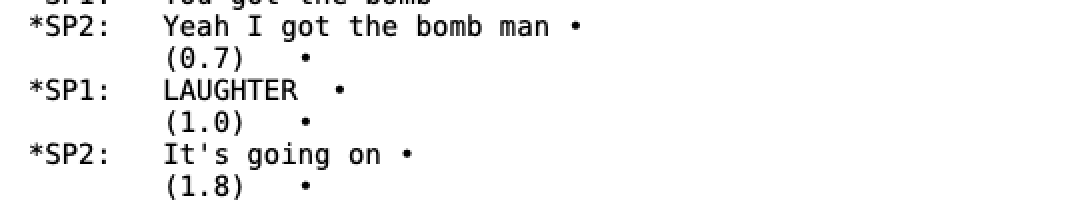
No. of Turns vs. Syllable Rate per turn graph

**Laughter Analysis**

Laughter in natural conversation occurs at frequencies higher than the average frequency of sounds in the audio signal. We therefore use a [Butterworth low-pass filter](https://en.wikipedia.org/wiki/Butterworth_filter) to remove lower frequency sounds from the audio signal. Gailbot then uses a Tensorflow model to identify parts of the audio signal that contain laughter and returns the start and end time for each instance that it identifies. This instance is then marked in the Gailbot transcript.

Gailbot uses a [Tensorflow](https://www.tensorflow.org/) deep neural network and the [librosa](https://librosa.github.io/librosa/tutorial.html" \l "overview) audio processing library to detect and classify laughter in its transcripts. The Tensorflow model is trained to detect laughter using supervised learning of laughter instances extracted from the [switchboard corpus](https://catalog.ldc.upenn.edu/LDC97S62), and is inspired by [JRGillick’s laughter detection module](https://github.com/jrgillick/laughter-detection) (Ryokai et al. 2018)

An instance of detected laughter is added to the Gailbot transcript as follows:



**Beat and absolute timing**

CA transcription as described by [Hepburn & Bolden (2012)](https://onlinelibrary.wiley.com/doi/10.1002/9781118325001.ch4) uses two methods of measuring silences: absolute time and ‘beat time’.

* **Absolute time:** measures the clock time between the end of one utterance and the start of the next in tenths of a second.
* **Beat time:** Relative measurements of the timings of gaps and pauses relative to the surrounding speech tempo, since it influences the perception of the length of silences in conversation.

The user can choose whether Gailbot transcribes absolute or ‘beat’ timings:

* Absolute mode: All pauses and gaps are transcribed in absolute time.
* Beat mode: All pauses and gaps are transcribed in ‘beats’, based on a model that emulates [Jefferson’s (1988)](http://liso-archives.liso.ucsb.edu/Jefferson/standard_maximum_silence.pdf) method of counting beats based on the tempo of surrounding speech.

**NOTE:** The type of Gailbot mode is added as a comment at the start of the transcript.

The model used to calculate beat timing was developed at the Tufts Human Interaction Lab and is summarized as follows:

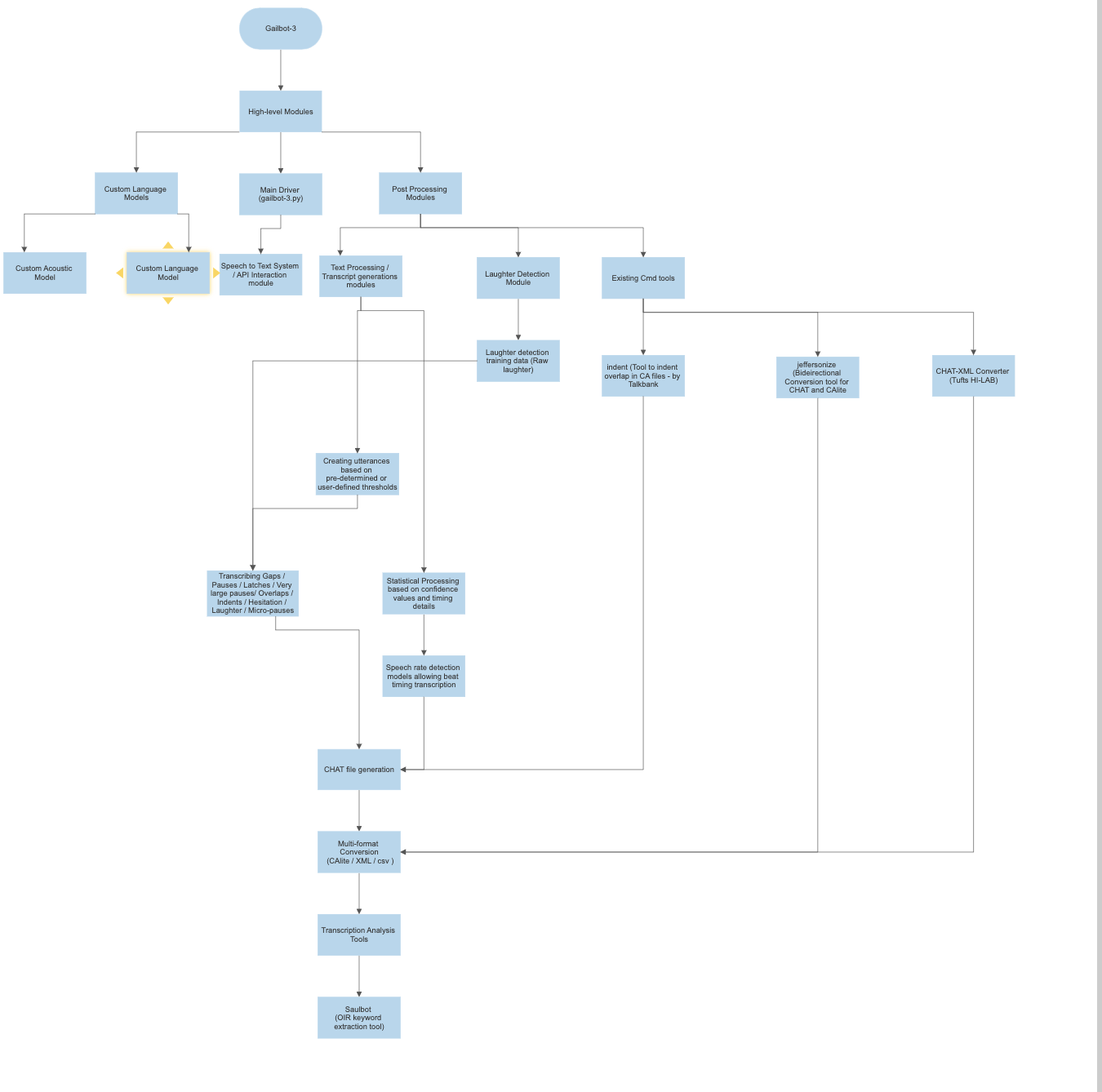
* The syllable rate is the number of syllables in a TCU divided by the total time of the TCU.
* The median syllable rate is found from the syllable rate for all turns.
* The beat time is the pause or gap time in seconds multiplied by the median syllable rate per second.
* The beat time in seconds is the above value divided by 4.
* The beat time in seconds is added to the transcript.

Developer Documentation

This section provides a complete overview of Gailbot’s design and its internal component.

**Design Flow**

**\*\*NOTE:** Please zoom in to view all the individual modules in the design



**Internal Modules**

Gailbot consists of several internal modules that are as follows:

**Speech to Text module:**

* This module is used to interact with the Watson Speech to Text API and send transcription requests.
* It uses the [autobahn library](https://buildmedia.readthedocs.org/media/pdf/autobahn/latest/autobahn.pdf) that provides a Websocket interface that can be used to send and receive requests from an external API over a broadband connection. The twisted autobahn interface is used by creating a Websocket interface factory that is run once per instance of Gailbot. Further, it is used to create a Websocket client interface that is created per transcription request sent to the API. Note that there are multiple requests sent to the API in a single factory instance.
* Secondly, this module uses a multi-threading reactor approach that is used to process multiple file requests using multiple threads simultaneously. This greatly increases the efficiency of Gailbot by allowing it to transcribe ‘n’ number of files in the least amount of time possible. Note that each transcription request takes real-time, or time equal to the length of the media file, to process.
* The multi-threading approach is implemented using python [queues](https://docs.python.org/2/library/queue.html) that provide a **‘task\_done()’** method that is used to mark a specific task in a queue as completed. Gailbot only proceeds once all media files in the queue have been processed. Since all files are being processed in a separate shell, this approach allows Gailbot to keep track of all requests.
* Finally, this script uses all documented methods to interact with the [Watson STT API](https://cloud.ibm.com/apidocs/speech-to-text#introduction)

**Main driver module:**

* As the name suggests, the main driver module is used to provide error control mechanisms, manage interfaces, and provide the base algorithm to completely run Gailbot.
* In terms of general functionality and error handling, this module performs the following functions:
  + Prevents users from inputting missing files through file and directory checking methods.
  + Verifies format of input media files to ensure it is supported
  + Authenticates user IBM Bluemix credentials before transcription request.
  + Organizes individual files, file pairs, and directory inputs.
  + Extracts audio from video files, links extracted audio to input, and overlays individual speaker audio files to be able to play the final CHAT transcript.
  + Adds all input audio files as a single request to be processed in a single connection to the Watson API.
  + Generates an independent output directory with all required files present.
  + Determines and sets the content type of the audio.
* Additionally, this module allows the user to interact with the following menu interfaces:
  + Main menu 🡪 To choose initial transcription option.
  + Recording menu 🡪 To set audio recording variables for new conversations.
  + Pre-request menu 🡪 Displays final file selection and transcription variables before processing request
  + Custom language menu 🡪 Allows user to manage IBM Watson’s custom and base language models and select them as part of the transcription request.
  + Custom acoustic menu 🡪 Allows user to manage IBM Watson’s custom acoustic models.
  + Post-processing menu 🡪 Allows selection of all post-processing modules to be applied as part of the transcription request.
  + CHAT Header menu 🡪 Displays changeable header variables for the CHAT file specifically.
  + CHAT values menu 🡪 Menu to change transcription thresholds for the CHAT file including pause / gap length, laughter length, and model weightage etc.
* In short, the main driver module ties together different components of Gailbot and is the central module that the user interacts with.

**Post-processing module**

* This module is used to manage all the post-processing modules to be applied. It does this by generating a list of modules that have to be applied and calling the driver function for each module sequentially.
* By default, the list includes all modules currently provided with Gailbot including Syllable rate detection, laughter, and CHAT file generation. However, it also provides an interface that allows the user to individually select the modules to be applied and subsequently change this internal list.
* Finally, the module provides a complete command line interface for module management featuring color coded text and selection lists to prevent invalid inputs.

Future Development

There are plans to incorporate a number of features into Gailbot in the future.

Due to Gailbot’s flexible designed, these will almost always consist of post-processing modules to enhance the transcription process.

Following are some of the development plans:

**Optional Speech to Text API**

Currently, Gailbot uses IBM Watson’s Speech to Text API to generate text, which is the developer’s choice for the API to be used. However, there is a wide selection of API’s available that have distinct features suitable for different tasks a user might want to achieve.

Ideally, the user will be able to set up a configuration file with different variables including one for the STT API to be used. This will automatically select the appropriate STT module and use the select service’s API.

**SPEECH TO TEXT API SUMMARY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Company** | **API Name** | **Special Feature** | **Languages** | **Quality** |
| Amazon | Transcribe | STT with multiple speaker recognition | English / Spanish | **GOOD** |
| Google cloud | Speech API | Context dependent STT with noise robustness | 120 | **INTERMEDIATTE** |
| IBM Watson | Speech to Text | Custom models, speaker labels, alternatives, and smart formatting | 11 | **GOOD** |
| Microsoft Azure | Bing Speech API | Text normalization, integration with [Azure LUIS](https://azure.microsoft.com/en-us/services/cognitive-services/language-understanding-intelligent-service/), and speech scenarios | 10 + 29 dialects | **GOOD** |

**Volume and pitch analysis**

One of Gailbot’s main developmental goals is the production of transcripts that capture structural features of conversation including prosody, intonation, and volume.

Volume analysis is an unsolved natural language processing problem that requires a complex statistical model to identify and classify parts of an audio signal higher and lower than specific thresholds.

Pitch analysis is also an unsolved NLP problem. This is because it is difficult to map pitch changes in the audio signal to the word being identified at that point in the text transcript. Additionally, there is no statistical model that defines a high pitch, low pitch, and a relation to the average pitch.

As of yet, we have been unable to determine an algorithm that will be able to identify volume levels and pitch and transcribe them for human voice. Most existing analysis tools are optimized for music.

**Challenges in volume and pitch analysis**

|  |  |  |
| --- | --- | --- |
| **Analysis type** | **Challenge** | **Potential Solutions** |
| Volume analysis | * Statistical model needed * Difficult to calculate average volume using audio signal. * No large data set available for supervised learning to form a machine learning model. * Lack of speaker diarisation prevents analysis for the dialogue model. | * Develop a model using audio signal intensity for a time series. * Use lowpass / high pass filters to remove additional data before analysis |
| Pitch analysis | * No model to classify high and low pitch. * Difficult to map pitch of an audio sample to the exact word. | * Construct a model to find the average pitch and use thresholds for classification. |

Acknowledgements and References

**Acknowledgements**

Gailbot is a project of the [Human Interaction Lab at Tufts](https://sites.tufts.edu/hilab/) University

Following is a list of all members of the development team:

* [Muhammad Umair](https://github.com/mumair01) (Tufts University)
* [Julia Mertens](https://twitter.com/therealjmertens) (Tufts University)
* [Saul Albert](https://www.lboro.ac.uk/subjects/communication-media/staff/saul-albert/) (Loughborough University)
* [Jan. P de Ruiter](https://twitter.com/JPdeRuiter?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor) (Tufts University)
* [Lena Warnke](https://twitter.com/LenaWrnk) (Tufts University)

**References**

Hepburn, A., & Bolden, G. B. (2012). The Conversation Analytic Approach to Transcription. In J. Sidnell & T. Stivers (Eds.), *The Handbook of Conversation Analysis* (pp. 57–76). Oxford: John Wiley & Sons.

Jefferson, G. (1988). Preliminary notes on a possible metric which provides for a’standard maximum’silence of approximately one second in conversation. In D. Roger & P. Bull (Eds.), *Conversation: An interdisciplinary perspective.* (pp. 166–196). Clevedon, UK: Multilingual Matters.

Kimiko Ryokai, Elena Durán López, Noura Howell, Jon Gillick, and David Bamman (2018), "Capturing, Representing, and Interacting with Laughter," CHI 2018

Glossary

This section contains definitions for some of the technical terms found in this document:

**Pair files:**

* Media files recorded with separate audio channels for each speaker.

**Post-processing modules:**

* Modules that enhance the text returned by the Speech to Text service.

**IBM Watson STT**

* IBM’s Speech to Text API that is used to convert sound/speech to text.

**Request**

* Data sent to the Speech to Text API.

**Customization Menus**

* Menus that allow users to change the transcription request itself or other variables used by Gailbot.

**Customization Weight**

* The weightage given to a trained custom language model being used in the transcription process as compared to the base language model. Value ranges from 0 to 1.

**Dialogue Model**

* IBM’s transcription model that identifies multiple speakers within a single media file

**Bit/Sampling rate**

* The number of bits encoded per second.

**Pre-Request Menu**

* Gailbot menu that is displayed before data is sent to the Speech to Text service.

**Base Transcription Model**

* A stock language model that is used in the transcription request and is provided by the Speech to Text API without any additional training.

Liability Notice

**Gailbot is a tool to be used to generate specialized transcripts. However, it is not responsible for the quality of any output produced. Generated transcripts are meant to be a first pass in the transcription process and are designed to be improved incrementally. They are not meant to replace the manual transcription process and can be improved upon. Gailbot uses IBM Watson’s Speech to Text API to generate text which required an IBM Bluemix account. The development team is not liable for any third-party transaction between the user and any external service used by Gailbot.**

**Additionally, the development team does not guarantee the accuracy or correctness of any statistical model used in Gailbot. These models have been developed in good faith and we hope that they are accurate. Users should always verify results.**

**We are committed to continue and develop Gailbot. Any feedback can be provided at:** [**hilab-dev@elist.tufts.edu**](mailto:hilab-dev@elist.tufts.edu)**.**

**However, we do not guarantee any response to emails/requests sent to us. We do not guarantee any bug-fixes or future updates.**

**Gailbot is an open-source product. We encourage its use in research. However, by using Gailbot, users agree to cite Gailbot and the Tufts Human Interaction Lab in any publications or results found as a direct or indirect result of using Gailbot.**