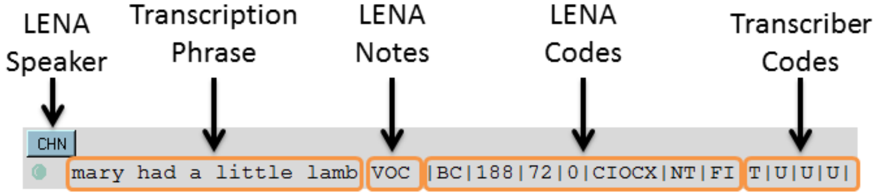
Unofficial BLL Programmer Guide

Data Files:

We mainly work with two types of data files:

1. TRS files – These are xml files that are exported from LENA. The files contain segment information (time boundaries, who LENA thinks is speaking, and more). The lab Transcribers open and edit these files in a program called Transcriber on the LENA machine. They look at the segments from 1hr to 1hr and 15 mins. They add in several types of information:
    a. What the person said (the "transcription phrase")
    b. Four codes that indicate who is speaking, who they are speaking to, whether or not the segment is a complete sentence, and the type of the segment (question or declarative, singing, reading). See the **transcriber manual** on Google Drive (account is info@babylanguagelab.org, check with lab member for password) for a full description of the transcription coding system.
    c. Information about linkage. Sometimes LENA will split speech incorrectly. For example, it there may be long segments that should really be split apart into two or more pieces. Or, there may be segments that were split apart that should really be together. Transcribers indicate this using a dot (.) and the special code (one of the four mentioned above). See the transcriber manual for more information.
2. ADEX files - These are regular CSV spreadsheets that are exported from a program called ADEX. In general, each line represents one segment. The columns list all kinds of information, including some data that you won't find in the TRS files.

Transcription terminology used in the codebase:



```
mary had a little lamb VOC |BC|188|72|0|CIOCX|NT|FI T|U|U|U|
```

- From a programming perspective, a *segment* refers to the whole image. Segments may have one or more *utterances*. These correspond to the "bullet points" in image above. This segment has only one utterance.
- For an XML-based definition of segments and utterances, see the documentation for the Utterance and Segment classes, or their code, which is located in the bll_app/data_structs/ directory.
- For a linguistic definition of segments and utterances, see the transcriber manual.

LENA Speaker Codes

Code	Description
CHF	Target Child Far
CHN	Target Child Near
CXF	Other Child Far
CXN	Other Child Near
FAF	Female Adult Far
FAN	Female Adult Near
MAF	Male Adult Far
MAN	Male Adult Near
NOF	Noise Far
NON	Noise Near
OLF	Overlapping Vocals Far
OLN	Overlapping Vocals Near
SIL	Silence
TVF	TV/Electronic Media Far
TVN	TV/Electronic Media Near

LENA Annotations/Notes Codes

Code	Description
VOC	Vocalization
SIL	Silence
FAN	Female Adult Near
VFX	Vegetative/Fixed-Signal Sounds
CRY	Child Crying

For a list of the transcriber codes and their descriptions, see the transcriber manual.

Special Transcriber Markings to watch out for:

XXX – unidentifiable speech (transcriber was not sure what they said)

BBL – babble (child is babbling rather than using distinct words)

Database:

The scripts read and store information using a simple SQLite database located at “bll_app/databases/bll_db.db”. A diagram of this database is available on the doxygen documentation page.

Classes that inherit from data_structs.base_objects.DBObject (like Segment, Utterance) have a corresponding database table. These objects have methods db_insert, db_delete, and db_select. These methods can be used to insert, remove, or reconstruct an instance of an object from the database table. This gives a basic mechanism for persistent storage, which occasionally comes in handy.

Constants that have to with LENA or Transcriber codes are also stored in the database. Most of these constants are automatically selected and read into a static object called DBConstants when an application starts up. Applications can use this class without having to select constants (or cache them) each time that they need to use them. See "bll_app/db/bll_database.py" for more details.

The database is set up so that SQL updates can be performed by dropping a SQL file into the directory "bll_app/db/sql/updates/". The files are named like "update-<number>.sql", where <number> is an integer that is incremented with each update. Just make sure you give your file the next highest number in the sequence. The system checks the files in the update directory each time a script is started (provided that script accesses the database). If it finds files with higher numbers than the last update file that was run, it executes them.

This means that when you're testing and you mess up your local copy of the database, if you want, you can actually delete your local copy (the .db file) and as soon as you start a script, a fresh copy will be recreated from scratch by running through all of the update scripts.

Launching scripts:

There's a special script in the bll_app/ root directory called launcher.py. This script sets up the environment so that the script files can access all of the different modules contained in the bll_app/ subfolders (parsers, db, data_structs, etc.). This can be a lot nicer than trying to mess with the Python sys.path list.

Each script has an associated file in the bll_app/app/ directory. App files should be named like "<name of script>_app". The app file should contain a class with the same name, but in camelCase. This class should override the method start(), which is inherited from the base App class. The start method is called when your application is invoked, so you can do whatever you need to start it up there.

App files should also contain a constructor. This constructor typically calls the base App class's constructor and passes it a parameter that indicates if the script will require a GUI, or if it's a command line script. If it requires a GUI, the base App's constructor will initialize GTK+ for you.

To create a new script, you just need to create an app file for it. You can call it from the command line (in the bll_app/ root directory) using "> python launcher.py my_app", where my_app is the name of your app file.

In order to set up shortcuts on the LENA computer desktop, I've made a simple batch file (launcher.bat). This is an executable file that accepts one parameter – the name of the app file for the script you want to run. It just executes the above command for that file.

Existing Functionality:

The existing codebase provides a small API for some common types of processing. Here are some of the things it does:

Task	Classes
Reading a list of Segments from a TRS file.	TRSParser
Reading a list of Segments from an ITS file	ITSParser
Reading a list of Segments from a CSV file	CSV Parser
Read, split, or play clips from WAV files	WavParser
Given a list of segments, pull out all chains of utterances connected by I/C codes	FilterManager
Filter a list of segments by Transcriber code, time, LENA speaker code, the presence of overlapping vocals	SegFilter and subclasses, also FilterManager can help
Check a list of segments for transcription errors	ErrorCollector, Utterance, Code and subclasses
Run SQL queries on small databases	Database, MemDatabase
Access LENA speaker constants, transcription constants, properties of segments (if it's far or near speech, overlap, etc.)	BLLDatabase, DBConstants, CodeInfo
Import a csv file into an SQLite database, export an SQLite database to a csv file	CSVDatabase
Calculate statistics from a list of Segments (Count number of segments with a particular speaker, list all segments in spreadsheet format, calculate the average speaking rate, and more).These classes also provide a means to export the stats in the form of a csv spreadsheet.	OutputCalc and subclasses
Display a progress bar	ProgressDialog
Display a window that allows the user to create a filter object	AddFilterWindow
Manipulate time strings, count number of words in strings, and other utilities	BackendUtils
Create combo boxes, show dialog boxes for opening files or folders, display message windows, automatically open results spreadsheets in Excel	UIUtils
Manage a large group of UI controls	Form
Control Praat at run time (dynamically write and execute scripts)	PraatInterop

Github repository and page:

Documentation: http://babylanguagelab.github.io/bll_app/docs/doxygen/html/index.html

Code: https://github.com/babylanguagelab/bll_app

The password for these accounts will be set to the lab password by the end of the week.

Contact:

If you have any questions (or complaints ☺), feel free to contact me:

Wayne Franz
Email: franz.wayne@gmail.com

All the best!