AuToBI Manual v.1.2.2

Andrew Rosenberg

October 1, 2012

1 Introduction

AuToBI is a tool for predicting prosodic event locations and types.

The prosodic events that AuToBI identifies are those defined by the ToBI standard for annotating Standard American English prosody. These are specifically, accented words and intonationally significant phrase boundaries. After these events are detected AuToBI classifies these into the categories defined under the ToBI standard.

2 Set up

The AuToBI system requires the AuToBI package, and trained classification models.

2.1 Obtaining AuToBI

A jar file containing the latest version of AuToBI can be found at

`http://eniac.cs.qc.cuny.edu/autobi/AuToBI.jar`

The AuToBI source is written in java and hosted on Github:

`http://github.com/AndrewRosenberg/AuToBI`

2.2 Configuring Paths

If compiling from source, the java classpath must include the necessary libraries. The libraries required for version 1.3 are listed below. These are also available on the github repository

Library Name	URL
Weka 3.7.5	`http://www.cs.waikato.ac.nz/ml/weka/`
Jakarta 2.1	`http://jakarta.apache.org/oro/`
log4j 1.2.15	`http://logging.apache.org/log4j/1.2/`
JNT FFT	`http://math.nist.gov/~BMiller/java/#FFT`
Commons Math 2.1	`http://commons.apache.org/math/`

2.3 Downloading Models

Trained models for each of the six prosodic event classification and detection tasks can be downloaded from the AuToBI website.

`http://eniac.cs.qc.cuny.edu/andrew/autobi/index.html#models`

Models trained on the Boston Directions Corpus, Boston Radio News Corpus and Columbia Games Corpus are available as of the preparation of this document.

3 Using AuToBI

To generate hypothesized prosodic events using AuToBI, a number of parameters are set through the command line. These parameters describe the input files including word segmentations, audio files, normalization parameters, model files and filenames for requested output files.

AuToBI outputs hypotheses in an interval tier of a Praat .TextGrid file with the input words in a separate, aligned tier. Future releases are likely to support other output formats. If the input TextGrid includes ToBI annotations in a Tier called "tones" (or another name as specified using the `-tones_tier_name` command line parameter) AuToBI will evaluate its performance on each detection and classification task against these annotations.

Note that while AuToBI will run all six detection and classification tasks in one execution, if fewer models are specified through the command line, only those tasks will associated models will be executed.

Sample AuToBI command line:

```
java -jar AuToBI.jar \
  -input_file=<filename> \
  -wav_file=<wav file> \
  -normalization_parameters=<parameters filename> \
  -pitch_accent_detector=<model_filename> \
  -pitch_accent_classifier=<model_filename> \
  -intonational_phrase_boundary_detector=<intonational phrase detector filename> \
  -intermediate_phrase_boundary_detector=<intermediate phrase detector filename> \
  -phrase_accent_classifier=<model_filename> \
  -boundary_tone_classifier=<model_filename> \
  -out_file=<filename>
```

3.1 Input file types

3.1.1 Word boundaries

AuToBI operates by predicting prosodic events at the word level. This requires word boundary annotations to be delivered to the system. All of the trained models have been trained using manually annotated word boundaries. Performance using automatically generated word boundaries has not been evaluated and may be worse than published results using manual word boundaries. Word boundaries are read by AuToBI in Praat's .TextGrid format. Word boundaries must be kept on an interval tier named "words" by default. Pauses or silences should have empty labels. If there are existing ToBI annotations for the words, AuToBI expects these to be on tiers named "tones" and "breaks" by default. If these are present, AuToBI will evaluate its performance when generating predictions.

Silent words are used to mark pauses. These pauses are used in AuToBI's processing, and n predictions are made on silent words. By default, words that match this regular expression string are considered "silent": `(#|>brth|brth|}sil|endsil|sil|_|_\*_|\*_|_\*)` A different regular expression designating silent regions can be set through the command line.

Some TextGrids include non-ascii characters. In this case a unicode character encoding is used to store the TextGrid files. AuToBI can support these files, but must be explicitly told what character

Flag	Description
US-ASCII	Seven-bit ASCII, a.k.a. ISO646-US, a.k.a. the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, a.k.a. ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark

Table 1: Supported Character Encodings and Flags

encoding is used. The `-charset` flag is used to send this information. Valid charset parameters are identical to those supported by Java's Charset (`java.nio.charset.Charset`). These are described in Table 1.

Command line parameter:

```
-input_file=<filename>
```

Optional command line parameters:

```
-words_tier_name=<words_tier>
   -tones_tier_name=<tones_tier>
   -breaks_tier_name=<breaks_tier>
   -silence_regex=<regular expression>
   -charset=<character set description>
```

3.1.2 Wave file

The signal processing routines included in AuToBI currently only support 16kHz wave formatted files. The SoX application is recommended for converting audio files to this format:
`http://sox.sourceforge.net/`.

Command line parameter:

```
-wav_file=<filename>
```

3.1.3 Normalization Parameters

To best normalize for speaker differences AuToBI uses z-score normalization of pitch and intensity based on audio data drawn from the speaker. AuToBI expects that each audio file contains speech from a single speaker. If there is more speech material available for a given speaker, this information can improve the robustness of the normalization. AuToBI includes a utility function to generate normalization parameters given a set of wave files. The utility is a class called SpeakerNormalizationParameterGenerator.

AuToBI Command line parameter:

```
-normalization_parameters=<parameters filename>
```

Command line for generation of normalization parameters

```
java -cp AuToBI.jar \
  edu.cuny.qc.speech.AuToBI.SpeakerNormalizationParameterGenerator \
  -wav_files=<wav files> -output_file=<filename> -speaker_id=<unique id>
```

3.1.4 Output of raw features

The features used in classification tasks can be output as an Arff formatted file. This file format is used by Weka. This allows users to either experiment with other classification scenarios, or perform other analysis of raw acoustic features.

AuToBI Command line parameter:
`-arff_file=<Arff formatted output filename>`

4 Training new AuToBI models

The six prosodic event detection and classification tasks are addressed using classification models trained with supervised training. In order to train new models, gold standard ToBI annotations must be delivered to the system.

Note: Support for model adaptation and active learning using unlabeled speech data are planned future features.

4.1 Input format

Currently, AuToBI requires ToBI annotations to be included in a TextGrid file along with word segmentations. Minimally, a words, tones and breaks tier are required. The words tier must be an interval tier. The tones and breaks tiers must be point tiers. Warnings, or errors will be raised if the tone and break annotations to not comply with the ToBI standard. By default these tiers must be named “words”, “tones” and “breaks”. Optional command line parameters can be used to indicate non-standard tier names.

Input wave files must be located in the same directory as the TextGrid input files and must be named with an identical filestem, but with the file extension ‘wav’. For example, if the following two input files are used:

```
/home/andrew/training/speaker1.TextGrid
/home/andrew/training/speaker2.TextGrid
```

The following two wave files are required:

```
/home/andrew/training/speaker1.wav
/home/andrew/training/speaker2.wav
```

Command line parameter:
`-training_filenames=<filenames>`

Optional command line parameters:

```
-words_tier_name=<words_tier>
-tones_tier_name=<tones_tier>
-breaks_tier_name=<breaks_tier>
```

4.2 Training models

Each classifier, corresponding to a single AuToBI task, is trained using the class, `edu.cuny.qc.speech.AuToBI.AuT`
The classifier or classifiers that are trained are specified using command line parameters. The parameter and value specify which classifier is trained and the filename where the trained classifier will be stored. These command lines are identical to those used when using AuToBI to generate hypotheses. Multiple command lines can be provided and multiple classifiers will be trained.

```
  -pitch_accent_detector=<model filename> \
  -pitch_accent_classifier=<model filename> \
  -intonational_phrase_boundary_detector=<model filename> \
  -intermediate_phrase_boundary_detector=<model filename> \
  -phrase_accent_classifier=<model filename> \
  -boundary_tone_classifier=<model filename> \
```

In the current version, the features used for each task and classifier algorithm used for each task are fixed without modification to the AuToBI source code. Future improvements may allow selection of classification algorithm and exclusion of features through command line parameters and/or configuration files.

As in the model evaluation scenario, external speaker normalization parameters can be delivered to the system. If none are present, speaker normalization is performed by aggregating all of the available speech from a given speaker. Each corpus reader must specify how speaker id information is annotated. If speaker ID information is not specified, AuToBI has the default behavior of treating each file as being spoken by a single, unique speaker.

Command line parameter:
`-model_file=<filename>`

Optional command line parameters:
`-speaker_normalization_files=<speaker normalization filenames>`

Sample Model Training command line:

```
java -cp AuToBI.jar \
  edu.cuny.qc.speech.AuToBI.AuToBITrainer \
  -training_filenames=<filenames> \
  -normalization_parameters=<parameter filenames> \
  -pitch_accent_detector=<filename>
```

5 Contact and Support

AuToBI is actively being developed. If you have any questions about its use, requested features, or would like to contribute to its development, feel free to email `andrew@cs.qc.cuny.edu`.