# **Language Audio Tool** Stephen 高思

This is a tool designed for editing language-learning MP3 files to make them more useful. It might be useful for other, more general, audio file editing too. It's very rough-and-ready, but thought it might already be in a stage where it could be useful for other people.

LanguageAudioTool automatically chops files up into audio “sections” and then can perform any combination of the following actions:

* Duplicate sections
* Slow or speed up sections
* Save sections as separate files

For example, you could pass an audio file containing 10 dialogues with padding silences, and have it create a new file, in which played each dialogue is first played at 70% speed, then once at 90% speed, and with all padding silences removed.

**Contents**

* 1 Background
* 2 [U](https://www.slimjimmusic.com/chord-progressions/#What_Are_Chord_Progressions)sage
  + 2[.1 S](https://www.slimjimmusic.com/chord-progressions/#Noting_Chord_Progressions)tep 1: Choose files
  + 2.2 Step 2: Subdivide into sections
  + 2.3 Step 3: Choose actions
  + 2.4 Step 4: Choose output
* 3 Notes
* 4 Todo
* 5 Contact

## **1. Background**

Many language learning courses contain (or comprise of) a number of audio files for practicing listening. Quite a lot of these are in what I'll call "exam format": some audio played once at full, fluent-speaker speed, followed by a silence of 15 seconds or more, then the next audio section. This isn't very useful for listening practice, in my experience, as it’s pretty hard to learn new words or grammar patterns while trying to parse full-speed speech, particularly if it's played only once. And the long silence between sections is just time for the mind to wander.

Imagine a world in which you could take such an MP3 file, pass it to an app that automatically finds the audio sections, strips out the silences and performs whatever actions you like on each section (e.g. play twice, first at 70% speed, then at 90% speed). Well, that world now exists ;)

This project makes use of:

* NAudio library (<https://github.com/naudio/NAudio>)
* NAudio Lame MP3 encoder (<https://www.nuget.org/packages/NAudio.Lame/>)
* Soundtouch library (<https://gitlab.com/soundtouch/soundtouch>)
* Varispeed soundtouch extension (<https://github.com/naudio/varispeed-sample>)

## **2. Usage**

This app uses a wizard format. There are 4 dialogs of options to set, and then the app will perform the required actions.

*2.1 Step 1: Choose files*

The first page is pretty self-evident. You just need to create a list of MP3 files by either clicking Browse to select files in an explorer window, or by directly dragging and dropping into the listbox.

Although this app can perform batch actions on n files, I recommend you first try it out on a single file to get the idea of how it works.

*2.2 Step 2: Subdivide into sections*

Next, we need to decide how to subdivide each MP3 file. There are three available kinds of subdivision:

* Subdivide by silences: for files which have padding silences, LanguageAudioTool can auto-detect these silences and define an audio section as just being any section of audio separated by silence.
* Subdivide by time: If there is not padding silence, you may prefer to just split based on time. E.g. a 2-minute MP3 file could be split into 6 equal 20-second audio sections.
* Whole file is one audio section (basically don’t subdivide). You may prefer this option if the files are short.

Additionally, you will notice that there are options concerning the first audio section. I included this because often the very first audio section is a copyright notice, or a piece of music or whatever. Therefore, you can choose to ignore the first audio section, or just play it once (regardless of what actions you choose to perform for the “real” audio sections).   
But bear in mind, this option will affect all audio files, so don’t choose this option if only one of the files contains this kind of intro.

*2.3 Choose actions*

Here you choose the actions to perform for each audio section. It’s the most complex-looking page but it’s actually very simple.   
The main window is the list of actions to perform. The buttons below the main window are to do simple manipulation on the list, like change the order of actions or clear the list. Finally, the “add” buttons are to add the corresponding action. A typical set of actions might be like this:

Play audio section at 70% speed  
Play 2 seconds of silence  
Play audio section at 100% speed  
Play a 1 second beep

With this set of actions, each dialogue will be played twice, first slowly and then at normal speed. The 2 seconds of silence helps to delimit the two repeats, and the beep makes it clear a whole new dialogue is now going to be played.

*2.4 Choose output*

The final step just requires you to choose the output folder for the generated files. Note that this app will not allow you to save files in the same folder as the original files; this is just to ensure you don’t write over your original files. I do not want anyone to potentially lose data.

Also, you can choose to save all audio sections as separate files. This might be a bit of a confusing concept because this is an app for batch file processing.

So, imagine that the inputs are 3 files called AudioA.mp3, AudioB.mp3 and AudioC.mp3   
And, that each file contains 4 audio sections.

If you choose one output per input, what that means is actually that the output folder will contain 3 files: AudioA.mp3, AudioB.mp3 and AudioC.mp3, each one the result of performing the user-defined actions.

If you choose one output per audio section, then you’ll have 12 files in the output folder, called for example AudioB003.mp3, and each file is just the result of performing the user-defined actions *on one audio section*.

## **3. Todo**

i. Convert other audio formats than MP3

ii. Include audio player functionality, where you can listen to an mp3, and it can perform your desired actions in real time

iii. Auto volume scaling. Scale the volume of files such that their loudest point will be at 100% volume (often language audio files are too quiet)

Let me know of any other suggestions.

## **4. Notes**

Detecting silence proved to be a far harder task than expected. You might think that you would just set some threshold volume and if all the samples are below that level, it's silence.

But that doesn't work; even an audio file that has had noise removed often will have compression artefacts; even in a part of the audio that has no audible sound, and displays as a flat line on an amplitude graph, it may actually have a surprisingly high number of "noisy" samples. The algorithm therefore includes thresholds not just for volume level but also for number of samples that need to be above that threshold, per 0.2 seconds, to count as being an audio section. It then joins up adjacent sections that have the same type.

A consequence of this algorithm is that it might erroneously cut up to 0.2 seconds of audio at the beginning or end of a section. Or, more likely, have up to 0.2s of padding at the beginning or end. This 0.2 value can be changed in the code, but making it smaller may affect performance (or even accuracy at extreme small durations).

I've tested this algorithm on around 100 audio files for 4-5 different sources. However, it would not shock me if it failed to parse a particular file incorrectly, hence why I don't let it write over existing mp3 files. If you find an example of a file it fails at, then please email me.

## **5. Contact**

Stephen Holtom

[stephen.holtom@gmail.com](mailto:stephen.holtom@gmail.com)

Wechat: Holtom