Reading App Builder

Using aeneas for Audio-Text Synchronization

SIL

Reading App Builder: Using aeneas for Audio-Text Synchronization

Last updated: 19 June 2023

Contents

1. Introduction 5

1.1. What is aeneas? 5

1.2. Will it work with our language? 6

1.3. How long does it take? 6

1.4. How do we run aeneas? 6

2. Windows Installation 6

3. Linux Installation 9

4. Using aeneas within Reading App Builder 9

4.1. Running the Audio Synchronization wizard 9

4.2. Troubleshooting 9

4.3. Viewing and testing the timing files created by aeneas 10

5. Fine Tune Timings 11

5.1. Fine tune timing files using the Fine Tune Timings viewer 11

5.2. Changing audio speed when doing fine tuning 13

5.3. Keyboard shortcuts when doing fine tuning 13

5.4. Sending Fine Tuning files to others 14

5.5. Fine tune timing files using Audacity 14

6. Adding voices to eSpeak 15

Acknowledgements

Thank you very much to Alberto Pettarin, Italy, for all his work developing **aeneas** and for making it freely available for use in synchronising text and audio around the world.

Website: https://github.com/readbeyond/aeneas/#aeneas

These instructions were written using aeneas 1.7.2, released 09 mars 2017.

Thank you to Firat Özdemir for inventing the **Fine Tuning** feature, a modified version of which is built into Reading App Builder.

Website: https://github.com/ozdefir/finetuneas

Thank you to Daniel Bair for his work on the Windows installer for **aeneas**.

Website: https://github.com/sillsdev/aeneas-installer

1. Introduction

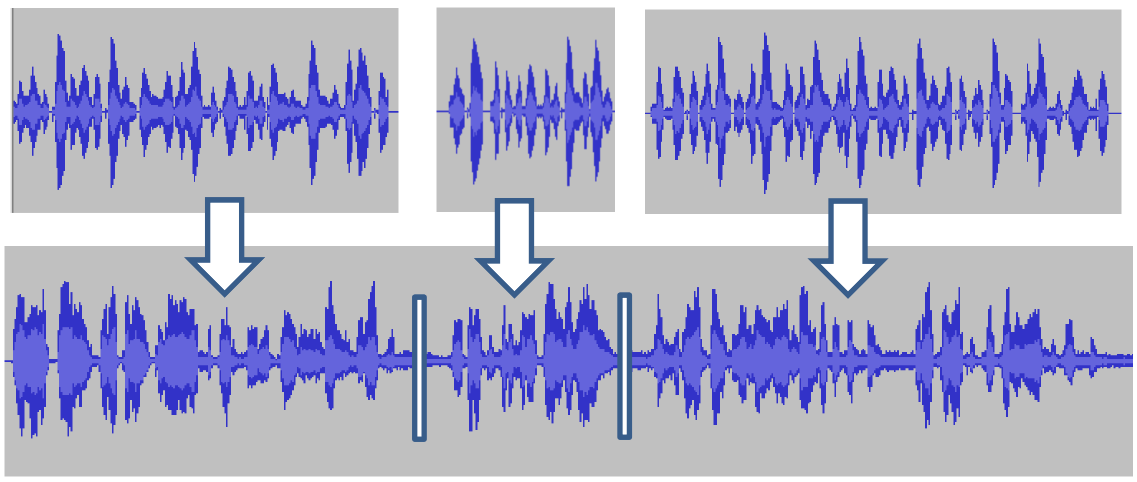
1.1. What is aeneas?

aeneas is a tool designed to "automagically synchronize audio and text".[1] It takes an audio file and a list of phrases as inputs. Its output is a synchronization timing file, i.e. the same kind of labels file you would create manually in Audacity.

The way **aeneas** works is to synthesize each phrase using a text-to-speech engine, and then compare the synthesized audio files against the input audio file to find the phrase breaks. To do this, a lot of clever mathematical computation goes on behind the scenes:

> Using the Sakoe-Chiba Band Dynamic Time Warping (DTW) algorithm to align the Mel-frequency cepstral coefficients (MFCCs) representation of the given (real) audio wave and the audio wave obtained by synthesizing the text fragments with a TTS engine, eventually mapping the computed alignment back onto the (real) time domain.[2]

Or to illustrate it more simply, the synthesized phrases…



…are mapped against the input audio file to find phrase breaks.

Best results are obtained:

- if you have a clearly spoken recording;
- if any background music is low enough in volume not to confuse the system;
- if the synthesized phrases correspond well enough to how the language is pronounced.

[1] https://github.com/readbeyond/aeneas/#aeneas

[2] https://github.com/readbeyond/aeneas/#one-sentence-explanation-pro-edition

1.2. Will it work with our language?

aeneas is producing impressively accurate timing files for languages around the world. We would recommend you try it out and see how well it works for your text and audio files.

What about special characters or tone marks? Since the text-to-speech engine copes best with simple A-Z Roman scripts, the **Synchronize using aeneas** wizard in Reading App Builder has a **Character Replacement** page, designed to replace special characters with simple ones (ɓ to b, ɛ to e, etc.). It also strips out accents and tone marks. The comparison algorithms are tolerant enough to match up the phrases even though the pronunciation differs.

If you have a non-Roman script, you will need to define a set of character replacements, transliterating your non-Roman script into a simple A-Z Roman script (अ to a, क to ka, उ to u, etc.). This approach has already been used successfully to synchronize text and audio in Hebrew. If you create a set of additional character replacements that work well for your script and language, please let us know so that we can add them to the default set of replacements.

You can find a selection of Romanization tables here:
http://www.loc.gov/catdir/cpso/romansource.html

1.3. How long does it take?

For a typical phrase-by-phrase synchronisation, running **aeneas** in **Windows** takes 1-2 minutes to process an hour of audio. Actual times will depend on the speed of your computer.

1.4. How do we run aeneas?

You need to Install aeneas on your computer and use the **Synchronize using aeneas wizard** in RAB to launch the synchronization process.

See section 2 for instructions on how to install **aeneas** on Windows, and section 3 on how to install **aeneas** on Linux. This is especially easy for Ubuntu users since **aeneas** is included automatically in the SAB installation.

2. Windows Installation

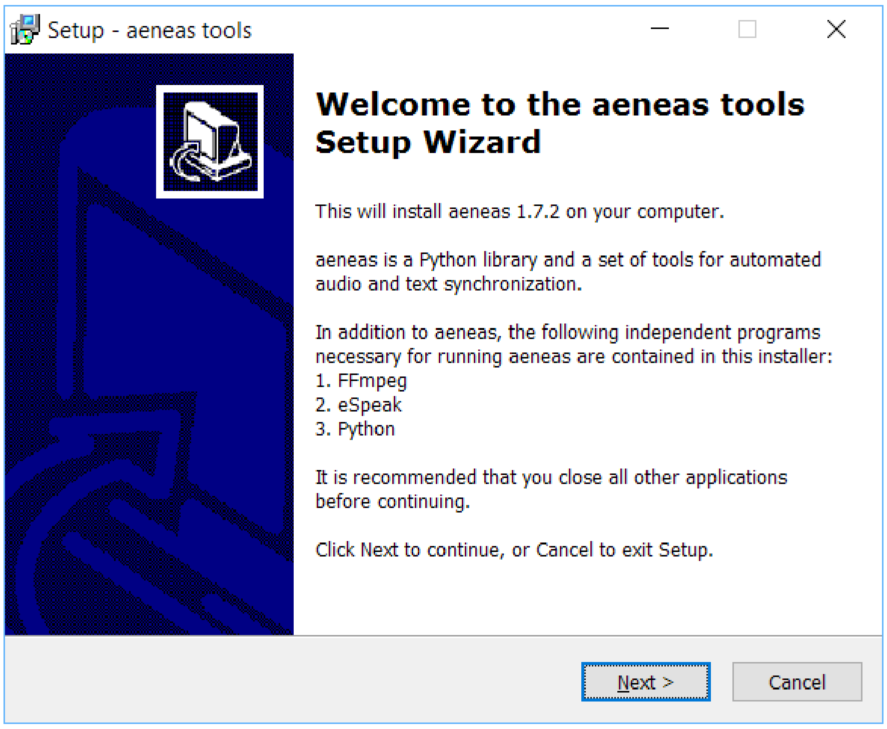
To set up **aeneas** on Windows, there are several programs and modules to download and install: FFmpeg, eSpeak, Python and aeneas. These can all be installed with a single setup program.

1. Go to the **Download** page on the Reading App Builder website http://software.sil.org/readingappbuilder/download/ and download the latest

aeneas setup program for Windows. You will find it under the heading **Audio Synchronization Tools**.

The filename will be something like **aeneas-windows-setup-1.7.2.exe**.

2. Double-click the file you have downloaded to start the installation wizard.



Welcome to the aeneas tools Setup Wizard

This will install aeneas 1.7.2 on your computer.

aeneas is a Python library and a set of tools for automated audio and text synchronization.

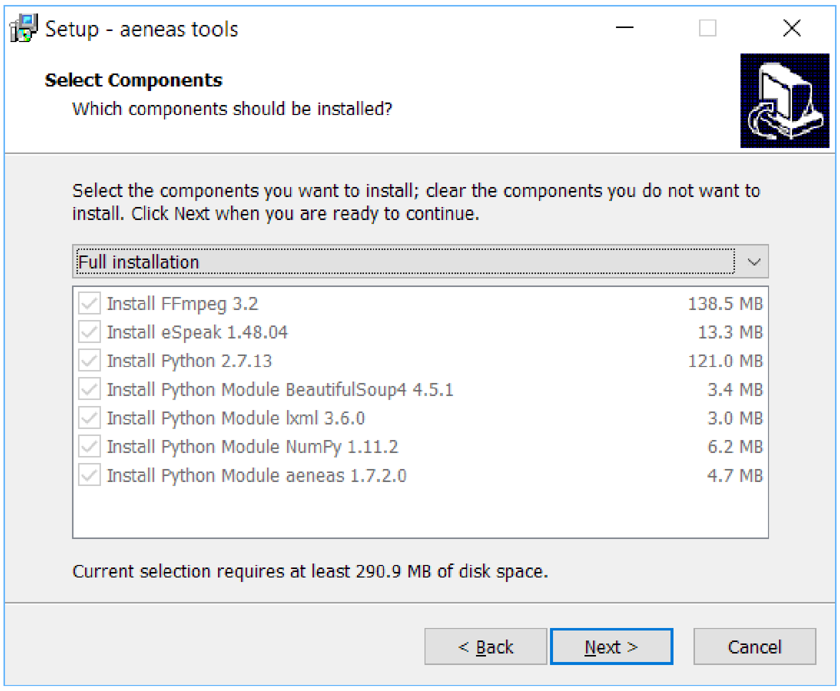
In addition to aeneas, the following independent programs necessary for running aeneas are contained in this installer:
1. FFmpeg
2. eSpeak
3. Python

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

3. Follow the instructions in the wizard. You can accept all the defaults.

4. On the **Select Components** page, you need the **Full Installation** which should be selected by default.



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

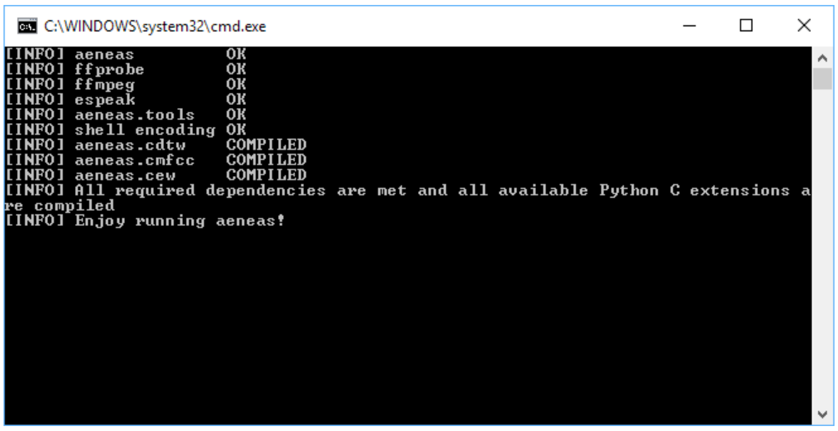
✓ Install FFmpeg 3.2	138.5 MB
✓ Install eSpeak 1.48.04	13.3 MB
✓ Install Python 2.7.13	121.0 MB
✓ Install Python Module BeautifulSoup4 4.5.1	3.4 MB
✓ Install Python Module lxml 3.6.0	3.0 MB
✓ Install Python Module NumPy 1.11.2	6.2 MB
✓ Install Python Module aeneas 1.7.2.0	4.7 MB

Current selection requires at least 290.9 MB of disk space.

5. On the **Ready to Install** page, click **Install**.

   You will see the wizard running several different installers: for FFmpeg, eSpeak, Python. It will also install three Python modules.

   Before the wizard completes, you should see a command box appear for a few seconds which verifies the aeneas installation.



```
[INFO] aeneas         OK
[INFO] ffprobe        OK
[INFO] ffmpeg         OK
[INFO] espeak         OK
[INFO] aeneas.tools   OK
[INFO] shell encoding OK
[INFO] aeneas.cdtw    COMPILED
[INFO] aeneas.cmfcc   COMPILED
[INFO] aeneas.cew     COMPILED
[INFO] All required dependencies are met and all available Python C extensions a
re compiled
[INFO] Enjoy running aeneas!
```

__Important__: If Reading App Builder was open while you installed aeneas, close it and start it again to ensure that it picks up the changes to your Path settings.

3. Linux Installation

If you are using Ubuntu, the **aeneas** software is a required dependency of Reading App Builder. This means that aeneas is automatically installed or updated along with Reading App Builder and you do not have to install it manually.

4. Using aeneas within Reading App Builder

4.1. Running the Audio Synchronization wizard

You can run the **Audio Synchronization using aeneas** wizard from four places within Reading App Builder:

1. The **Audio Files** tab for a book. Select one or more rows in the table, right-click and select **Synchronize using aeneas**.
2. The **Audio Synchronization** tab for a book. Click on the button **Synchronize using aeneas**.
3. The **Books** tab on the Books page. Right-click on a book and select **Synchronize using aeneas**.
4. The Apps tree on the left of the screen. Right-click on a book and select **Synchronize using aeneas**.

Follow the instructions in the wizard to configure the synchronization.

For Windows users, if the wizard asks, "Where have you installed aeneas?", please follow the installation instructions in section 2 of this document.

After the final page of the wizard, the synchronization process will start in a separate command Window. It runs a batch file (or bash script) which calls an **aeneas** task for each chapter of each book.

As timing files are generated you will see their filenames being added to the table on the **Audio Files** page.

4.2. Troubleshooting

If the synchronization fails, please ensure that aeneas has been installed correctly. You can check the installation by selecting **Tools ➢ Check aeneas Installation…** from the Reading App Builder main menu.

A common problem on Windows is that **eSpeak** stops working after a while. To correct this, run the aeneas Windows setup program again.

To verify that eSpeak has been installed correctly and is in your path, open a new Windows command prompt and type:

```
espeak hello
```

If all is well, you should hear the word “hello” pronounced.

4.3. Viewing and testing the timing files created by aeneas

To view a generated timing file, select a row on the **Audio Files** page, right-click, and select **View Timing File**.

To test the generated timing files with the text and audio, select a row in the **Audio Files** page, right-click and select **Export to HTML**.

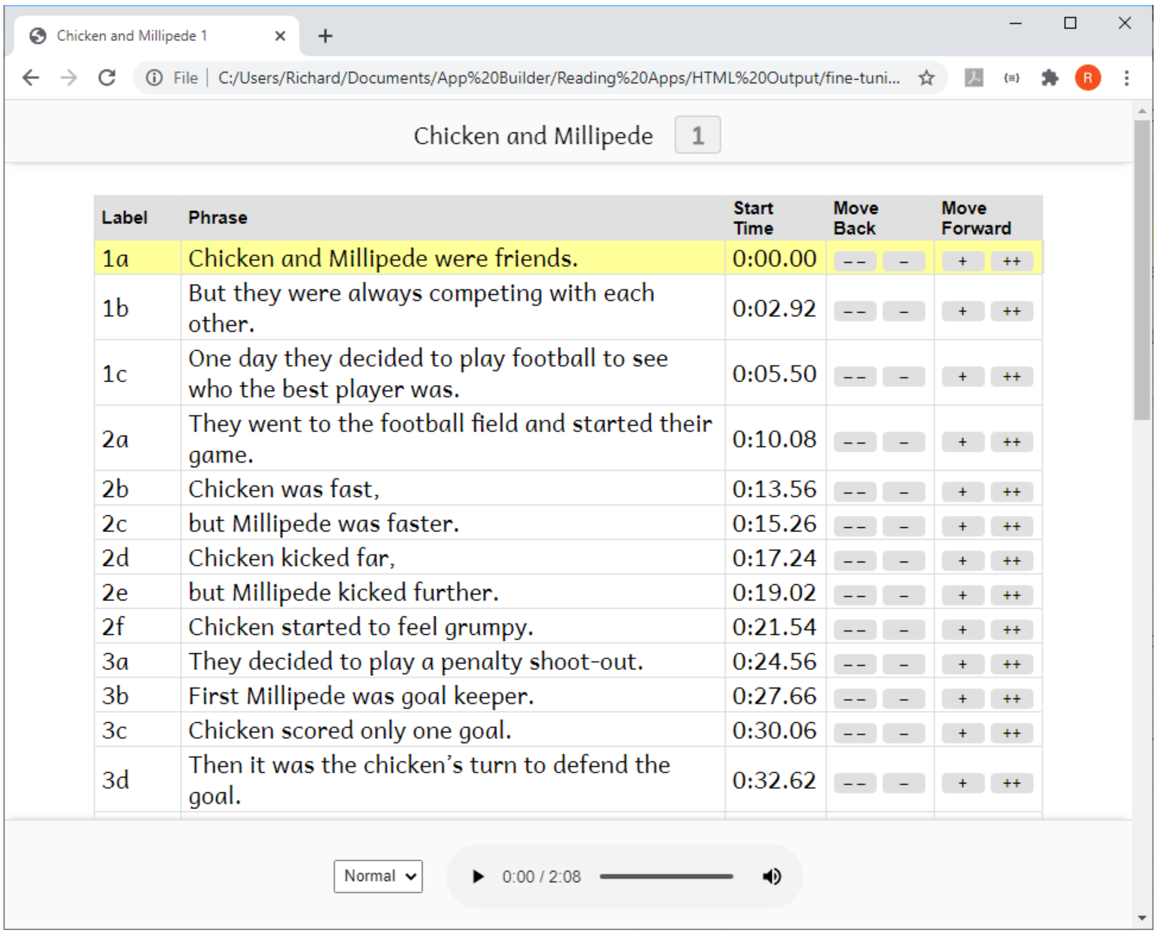
5. Fine Tune Timings

5.1. Fine tune timing files using the Fine Tune Timings viewer

To fine tune a generated timing file:

1. Select a row on the **Audio Files** page, right-click, and select **Fine Tune Timings**.

   A page should open in your default browser, looking something like this:


Chicken and Millipede 1
File | C:/Users/Richard/Documents/App%20Builder/Reading%20Apps/HTML%20Output/fine-tuni...
Chicken and Millipede 1
Normal
0:00 / 2:08

Label	Phrase	Start Time	Move Back		Move Forward	
1a	Chicken and Millipede were friends.	0:00.00	--	-	+	++
1b	But they were always competing with each other.	0:02.92	--	-	+	++
1c	One day they decided to play football to see who the best player was.	0:05.50	--	-	+	++
2a	They went to the football field and started their game.	0:10.08	--	-	+	++
2b	Chicken was fast,	0:13.56	--	-	+	++
2c	but Millipede was faster.	0:15.26	--	-	+	++
2d	Chicken kicked far,	0:17.24	--	-	+	++
2e	but Millipede kicked further.	0:19.02	--	-	+	++
2f	Chicken started to feel grumpy.	0:21.54	--	-	+	++
3a	They decided to play a penalty shoot-out.	0:24.56	--	-	+	++
3b	First Millipede was goal keeper.	0:27.66	--	-	+	++
3c	Chicken scored only one goal.	0:30.06	--	-	+	++
3d	Then it was the chicken's turn to defend the goal.	0:32.62	--	-	+	++

2. Click on the first row of the table to start the audio playing.

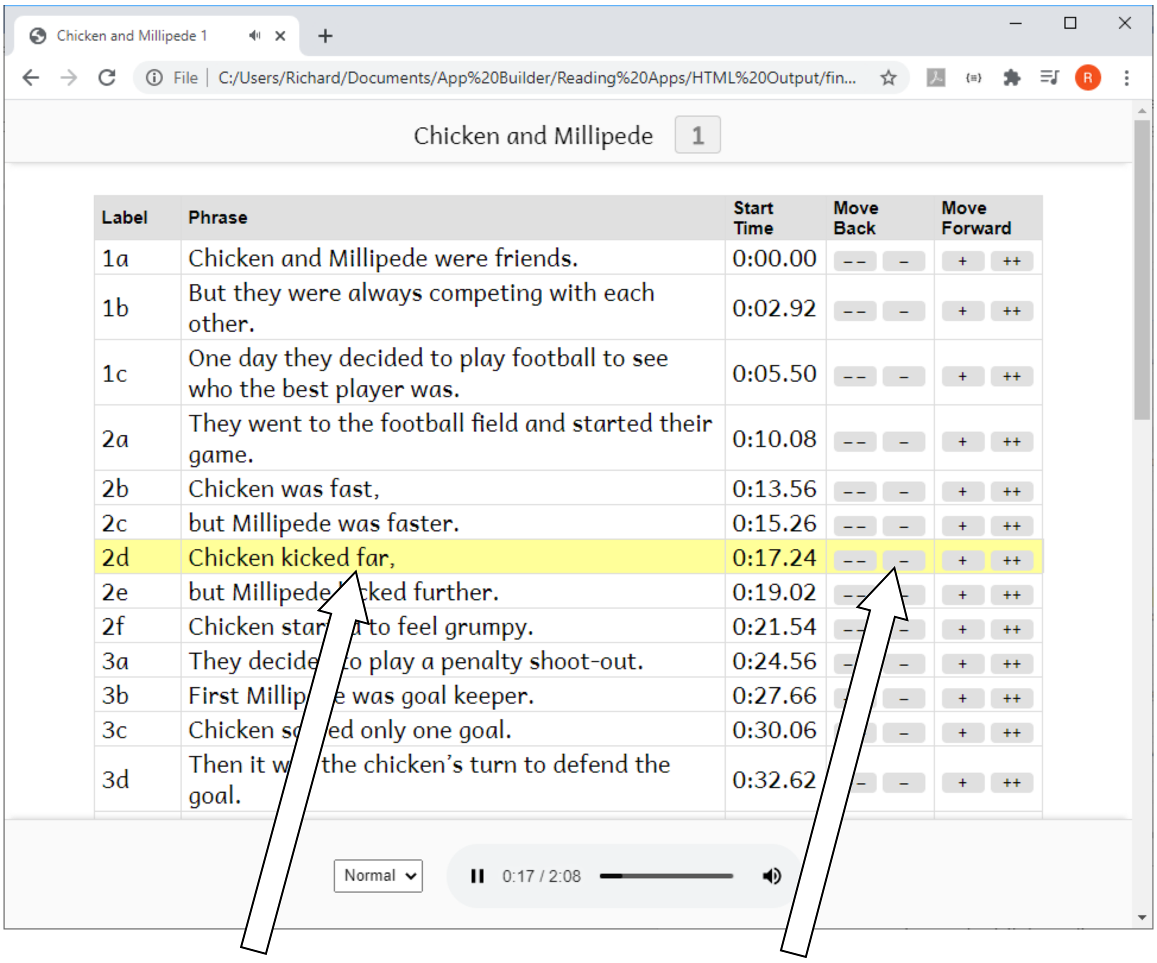
3. As soon as you hear the first word or two from the selected row, click on the next row.

4. Continue clicking on each row to listen for enough time to determine whether the timing break has been placed in the right place, i.e. not too early and not too late.

   You can also use the **N** keyboard shortcut to move to the next row.

5. If you hear that the timing for a row needs to be fine-tuned, click the plus (+) button to move the start time forward by 0.1 seconds, the double-plus (++) to move

forward by 1 second, the minus (–) button to move the start time backwards by 0.1 seconds, or the double-minus (– –) to move back 1 second.


Chicken and Millipede 1
File | C:/Users/Richard/Documents/App%20Builder/Reading%20Apps/HTML%20Output/fin...
Normal
0:17 / 2:08

Chicken and Millipede 1

Label	Phrase	Start Time	Move Back		Move Forward	
1a	Chicken and Millipede were friends.	0:00.00	– –	–	+	++
1b	But they were always competing with each other.	0:02.92	– –	–	+	++
1c	One day they decided to play football to see who the best player was.	0:05.50	– –	–	+	++
2a	They went to the football field and started their game.	0:10.08	– –	–	+	++
2b	Chicken was fast,	0:13.56	– –	–	+	++
2c	but Millipede was faster.	0:15.26	– –	–	+	++
2d	Chicken kicked far,	0:17.24	– –	–	+	++
2e	but Millipede kicked further.	0:19.02	– –	–	+	++
2f	Chicken started to feel grumpy.	0:21.54	– –	–	+	++
3a	They decided to play a penalty shoot-out.	0:24.56	– –	–	+	++
3b	First Millipede was goal keeper.	0:27.66	– –	–	+	++
3c	Chicken scored only one goal.	0:30.06	– –	–	+	++
3d	Then it was the chicken’s turn to defend the goal.	0:32.62	– –	–	+	++

Click on each row for enough time to check if the audio starts at the right place.

Adjust the start time of the currently selected timing using the plus and minus buttons.

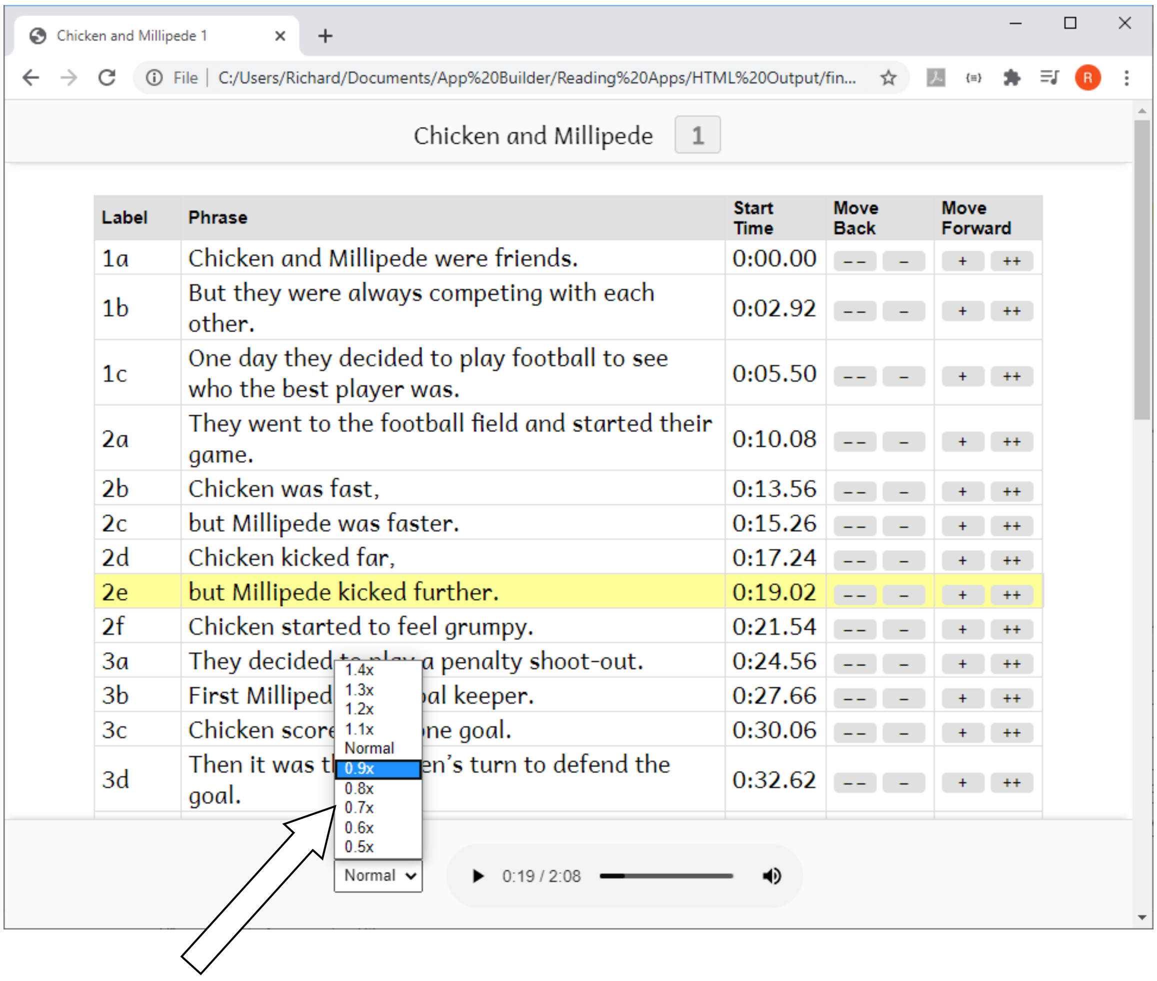
6. When you are happy with the current row, continue clicking on the subsequent rows in the same way, fine tuning those that need adjusting.

7. When you have reached the last row of the table, save your changes by clicking on the button **Save Changes to Timing File** at the bottom of the page (or by using the keyboard shortcut **T**).

Important
For security reasons, your browser does not have permission to save the modified timing file in its original location. It will be saved to your default Downloads folder. Reading App Builder can watch this folder for modified files and move them back to your project data folder. Specify the Downloads folder to watch in **Settings ➢ Files**. Otherwise, you will need to move the modified files back to your project manually, e.g. using the **Add Timing Files** button.

5.2. Changing audio speed when doing fine tuning

If the audio is playing too quickly for you to fine tune accurately, you can slow down the playback speed using the speed selector at the bottom of the screen.



Chicken and Millipede 1
C:/Users/Richard/Documents/App%20Builder/Reading%20Apps/HTML%20Output/fin...

Chicken and Millipede 1

Label	Phrase	Start Time	Move Back	Move Forward
1a	Chicken and Millipede were friends.	0:00.00	-- -	+ ++
1b	But they were always competing with each other.	0:02.92	-- -	+ ++
1c	One day they decided to play football to see who the best player was.	0:05.50	-- -	+ ++
2a	They went to the football field and started their game.	0:10.08	-- -	+ ++
2b	Chicken was fast,	0:13.56	-- -	+ ++
2c	but Millipede was faster.	0:15.26	-- -	+ ++
2d	Chicken kicked far,	0:17.24	-- -	+ ++
2e	but Millipede kicked further.	0:19.02	-- -	+ ++
2f	Chicken started to feel grumpy.	0:21.54	-- -	+ ++
3a	They decided a penalty shoot-out.	0:24.56	-- -	+ ++
3b	First Milliped al keeper.	0:27.66	-- -	+ ++
3c	Chicken score ne goal.	0:30.06	-- -	+ ++
3d	Then it was t en's turn to defend the goal.	0:32.62	-- -	+ ++

Change the playback speed here

5.3. Keyboard shortcuts when doing fine tuning

The following keyboard shortcuts are available in the Fine Tuning interface:

Key	Action
Spacebar	Start/Pause audio
T	Save timing file
N	Start playing from the next row
P	Start playing from the previous row
B	Set the phrase start time to be at the current audio position

A (or Q)	Move the start time of the current row back 1 sec
S	Move the start time of the current row back 0.1 secs
D	Move the start time of the current row forward 0.1 secs
F	Move the start time of the current row forward 1 sec

5.4. Sending Fine Tuning files to others

If you want to do fine tuning well, it can take a lot of time of concentrated listening. You might have friends and colleagues who are willing to help with this. It is possible to send them the Fine Tuning files to work on without them needing to install Reading App Builder.

To do this:

1. Select one or more books, right-click, and select **Fine Tune Timings**.
2. In the Fine Tuning wizard, choose **The files will be sent to others to help with the fine tuning**.
3. On the next page, **Audio Files**, choose to copy the audio files (if you have a good internet connection) or choose not to copy them (if your internet connection is limited and you will tell your friends where they can download the audio files).
4. On the next page, **Zip Files**, choose to zip the fine tuning files together to send to your friends.
5. On the next page, **Output Folder**, choose where you would like the files to be created, then click **Create Files**.
6. Send the files to your friends, ask them to unzip them on their computer, and explain to them how they can work on the timings. They should send the modified timing files back to you.

5.5. Fine tune timing files using Audacity

Instead of using the Fine Tune Timings viewer, you can make corrections in Audacity. To do this:

1. Open the audio MP3 file in Audacity.
2. Import the timing file, by selecting **File** ➢ **Import** ➢ **Labels…**
3. Zoom into the waveform.
4. Adjust the position of any labels that are not in the right place.
5. Export the modified labels track, using **File** > **Export Labels…**

6. Test the synchronisation again within Reading App Builder using the **Export to HTML** utility (see above) to see if the timing problems have been resolved.

6. Adding voices to eSpeak

By using character replacements in the **Synchronize using aeneas** wizard, you will find that you can get good synchronization results for many languages using the existing eSpeak voices.

Adding a new voice in eSpeak is not easy, but if you would like to try, please follow the instructions here: http://espeak.sourceforge.net/add_language.html