Experiments in generating audiobook files with epub2tts-edge

John C Nash

2025-01-09

Initial version 2024-07-29, revised 2024-10-19, 2025-01-09

Background

Audiobooks have become a popular way to hear rather than read the content of books, in addition to being an important resource for the visually impaired. Unfortunately, creating good-quality audiobook files is not easy. Written content has many features of typography, layout, illustration and idiosynchratic expression that do not carry over to the spoken word. Moreover, most readers, even professional actors, have difficulty maintaining their enunciation and providing expressiveness over the generally extended duration of a book. Furthermore, the services of professional actors and readers are expensive relative to the resources of the visually impaired and amateur authors.

The development of large language models for converting text to speech (i.e. TTS) has reached the stage where it could be considered a possibility for carrying out the task of generating audiobooks from textual form, for example, epub files. This article is a set of notes on some experiments with freely available software to do precisely this.

Popularly such software is called "artificial intelligence", but "large language models" more correctly describes the programs, since they use models of how language is pronounced and expressed to create the sounds of a human voice in a particular accent and style. In particular, the software used in developing this report comes from the Github project **epub2tts-edge** of Christopher Aedo. See https://github.com/aedocw/epub2tts-edge. Aedo has done a great service in this and the earlier **epub2tts** project which is based on the Coqui AI TTS tools (https://github.com/coqui-ai/TTS). Moreover, he has responded quickly and helpfully to queries.

There are undoubtedly other tools that I could have used, but experimentation is very time consuming, and preliminary experiments showed promise, first with **epub2tts**, then somewhat better results with **epub2tts-edge**. This led to more, and more ambitious, experiments, as well as some investigations into ways to improve outputs or make them more usable to potential listeners.

Source works

I have generally put my creative writing on the UK writers' site *https://obooko.com*. This currently is about a dozen novels and two collections of memoir/short story material. A feature of several of my works is that they involve more than one language. My first novel, **Thursday Afternoon**, for example, has a scene where a young Canadian airforceman is working with some British soldiers and a Flemish woman to find and clear a landmine in 1944 Flanders. The Flemish woman communicates with the Canadian in French, but falls back into Flemish on occasions. There are thus three languages mixed in a couple of pages of text, which is a challenge to the software. Moreover, I have yet to discover how to inform the software of the language to be read, though I feel certain there must be a way to do this. For example, I am surprised that simple tags cannot be put into the text to be read giving the language which follows. Similarly, some indication where emphasis is desired would be useful.

Installation of the software

Setup of **epub2tts-edge** followed instructions in the github repositories to install the software for the two projects. Note that this may involve a considerable download of code and binaries depending on the existing software installation. I worked with Linux Mint version 22 (Wilma). One limitation for me is a lack of CUDA-capable graphical processing units (GPUs), so performance of the tools on my computers was somewhat lacklustre. Typically rendering of an epub to an m4b or mp3 took over an hour.

Note that the `epub2tts-edge` software currently calls for Python support using at least version 3.11 or later. The audio rendering also needs an active internet connection.

The software was apparently developed for a Linux environment. While there is the possibility in Windows of running the Windows Subsystem for Linux, I have chosen to use Oracle VirtualBox to install Linux Mint 22 (Wilma) and run the software within a virtual machine (VM). This has the advantage of isolating the text to speech work from other ongoing activities. Initially, my working platform was an earlier Linux Mint with Python 3.09. I did not wish to disturb some stable facilities of my working environment, though I have since upgraded the host to LM22. My writing colleague and coordinator of the Stittsville Creative Writers Group, R J Partington, installed LM22 under Windows. Another colleague in the group, Allan McCarville wanted to install on a Macintosh, but we had difficulty getting that set up, and it seems likely he will repurpose an older PC to work with audiobooks.

Standard usage of the software

We assume we have followed instructions in the Github repository to install the software.

epub2tts-edge is intended to be applied first to an **epub** file. Such files are described in https://www.w3.org/publishing/epub3/ and https://en.wikipedia.org/wiki/EPUB; they are packaged collections of HTML text and metadata. Suppose that we have a file **mybook.epub**. Then we first convert this file to a properly structured plain text file **mybook.txt** by opening a terminal (command line screen) and issuing the command

```
        epub2tts-edge mybook.epub
```

This takes only a few seconds to execute. This (usually) creates a file mybook.txt and extracts a cover image mybook.png from the epub file. The text file begins with two lines of the form

```
Title: Dodging the Potholes
Author: John C. Nash
```

It also has some lines that commence with a hash symbol (#) that give “chapters” or “sections” and the text behind the hash is read out in the output audio as a chapter heading. However, the algorithm by which these lines are created is not clear to me, but they can be manually inserted into the file or edited with a plain text editor. For example, generated ones like

```
# Part 24
```

can be edited to a more desirable phrase like

```
# Part 24 -- At the party
```

Text following the hash is read as the title of the section. Below we discuss how to break up the work and have separate audio files, which is important in saving time and effort for correction of individual parts of the overall novel or story.

Failure of the epub to text process

We have observed some cases where `epub2tts-edge` fails to carry out the conversion to text and extraction of the cover image. There are at least two other approaches we have used, and some others we have investigated:

1) **calibre** program `ebook-convert`

This tool is part of the **calibre** ebook management software. We simply use the command

```
ebook-convert myfile.epub myfile.txt
```

Generally `myfile.txt` will be more or less correctly formatted, though we have noticed that the initial "Title:" and "Author:" may be combined in one line. This can, of course, be corrected with a text editor. This tools should be the first alternative to `epub2tts-edge` when that program fails to convert the epub successfully.

2) **pandoc**

```
pandoc -f epub -t plain -o myfile.txt myfile.epub
```

will create a text version of the epub. Unfortunately, it often breaks paragraphs into multiple lines which cause undesirable pauses in the eventual audio file(s). (See https://unix.stackexchange.com/questions/647686/convert-epub-to-txt-and-preserve-original-formatting). Moreover, the section delimiters ("# (some text)") are often missing. `pandoc` also leaves two line endings between "paragraphs". It should be noted that epub2tts-edge then pauses for each line ending, so these single line endings need to be removed. However, there are some parts of the epub such as the title and cover pages where we would like to retain the single line endings (or double them to ensure separation of important fields that should be read separately). Note that omitting the "-t plain" option leaves some tags within the resulting text file that could cause unwanted spoken material.

3) Some word processing programs like **Word** or **LibreOffice** may be able to save a text file, but likely the user will need to manually edit the output.

CAUTION: Plain text often gets corrupted (or perhaps "enhanced") by word processing software. It is advisable to use an editor for plain text. In Linux, I like **L3afpad** and **gedit**. In Windows, the rather ancient **Notepad** is still a reliable standby. Below we discuss some tweaks to the text file that are generally needed to give a polished audiobook. For example, each paragraph should be a single line terminated with a line ending.

4) I could not get the software **epub2txt2** (https://github.com/kevinboone/epub2txt2) to compile (and hence was unable to install it).

5) **epub2txt** is a Python program (https://github.com/ffreemt/epub2txt) that can be easily installed with the command `pip install epub2txt` if Python is properly set up. This seems to get all the material, but gives no spaces between what would be "paragraphs". Moreover, verbatim material has line endings, so some attention to that detail may be relevant.

Conversion to audio file(s)

The much longer process that **epub2tts-edge** carries out is converting the text file to an audio file of type m4b (https://www.lifewire.com/m4b-file-2621958). For a novel, this may take of the order of an hour if the work is in a single computer text file. Breaking up the text file into parts is therefore helpful for repairing glitches in individual sections, and my experience is that these glitches are inevitable. We want to avoid having to re-run the process for the whole book rather than one section.

The second stage run of **epub2tts-edge** generally requires us to specify a "voice" or "speaker". If no speaker is given, a neutral English-only speaker with a quite moderate American accent is used, which can be specified by the identifier **en-US-AndrewNeural**. Moreover, for multilingual texts we need to use one of a few special multilingual voices. For example, the specifier **en-US-AndrewMultilingualNeural** will use the same voice as the default **for English**, but process multiple languages using **different** speaker voices for the other languages, which is disconcerting for the listener. Note that it seems that different "speaker" specifications will handle the same text differently and some words pronounced correctly by one voice will be mangled by another. We have yet to find out how to get a seemingly common voice for different languages.

The second stage command can also specify a cover image which can be in jpg or png format. The command is

```
    epub2tts-edge --speaker speakerID --cover mybook.png  mybook.txt
```

where `speakerID` is a valid speaker identifier and `mybook.png` (or some other png or jpg image file) is a cover image available in the same directory as `mybook.txt`. The output will be a file `mybook-speakerID.m4b`. This can be a very large file of several hundred megabytes for a full novel, and take 8-16 hours to play. The m4b format allows bookmarks so the user can pause and restart. On the other hand, older devices may not be able to play these files. They are, however, convertible to other audio file formats, in particular mp3.

As mentioned, paragraphs should not have line breaks. Such line breaks in the text file `mybook.txt` cause quite long pauses in the voice reading. Therefore line breaks that are non-pausing should be removed, giving full paragraphs on a single "line".

Some important general problems

Omitted text

WARNING: I have discovered that text in an epub file that is in a monospaced (i.e., typewriter-like) font is somehow eliminated completely from the file `mybook.txt` and must be re-inserted manually. This is a nuisance and can cause grief. For example, in my novels, I use such fonts for "letters" and other correspondence or quotations such as the text on a tombstone. I have had some email correspondence with Christopher Aedo about this, but so far we have not found the source of the glitch. (Note 2024-11-01: This may have been fixed, at least in part, but has not been tested.)

As mentioned, it is feasible to replace the first stage run of epub2tts-edge with a different translator from epub to text. Some other choices were noted above.

I have not tested to see if possibly material in some other fonts may be omitted.

Lost features

I have not been able to get strikeout, bold or italic to be expressed in the audio output.

Emphasis, by way of capitalization, is also lost, as can be seen by processing the file **cap4emphasis.txt** (the text below) through epub2tts-edge. Note line endings have been removed.

```
# Capitals for emphasis

Putting words in capitals to emphasize them will only get them pronounced letter by letter in some cases

"I said," often has awkward pauses or mis-emphasis.
```

Difficulties with language change

When the language of the text changes, particularly with dialogue, there can be unwelcome results:

- Sometimes the language model, which appears to be linked into the speaker voice chosen, either fails to recognize the language change, so continues in the current language setting inappropriately, or else will continue after the "new" language in that language when it is not correct.
- The voice of the speaker may change drastically in the new language, which is disconcerting for the reader.

I have not (at 2024-1-9) yet found work-arounds for these issues.

Workarounds

Some glitches in pronunciation can be overcome by use of phonetic text. Here are a number of examples, all of which are small files that can be tested using

```
  epub2tts-edge [file].txt
```

where [file] is replaced by the appropriate identifier. Note that in some cases the long lines needed by epub2tts-edge give display problems for the Rmarkdown editor in R-studio (https://www.rstudio.com/categories/rstudio-ide/) and the resulting pdf output in this document.

0pauses.txt

```
0pauses.txt

Various things cause pauses.

Line endings.

Like
these
ones.

A period will give a break. Just like that. But shorter than a line ending.

And John C. Nash sounds incorrect compared to John C Nash with no period
behind the middle initial.

But dashes -- seem to be ignored.

What about a semicolon; does that work? More or less. Perhaps dashes are same.
```

titles.txt

```
# titles.txt

Abbreviations may or may not be expanded.

Lt. Cmdr. Try full words "Lieutenant Commander" or phonetically Leftenant Commander

Putting M I 9 together gets MI9.  Better separated.

Ms. (not sounding like Miz). Try Miz.

Mrs. could be better MISSUS or Missus. Note that epub2tts-edge seems to get this to work OK.

We can also try versus and vs.

Monsieur Lefebvre not M. Lefebvre

POWs  ("s" not pronounced sometimes). Try P O W's.
```

names.txt

Proper names, especially those originating outside English, cause mispronunciation.

```
# names.txt

Vaudreuil try  vohdroy
A / C or Ae slash C for air conditioning or alternating current. Spell it out.
Arnprior Arnpryor Arnprye-ore
Belisle  Belleel
```

```
Jean Godin use zjonn gohdann
Lebreton Lebretton
Levis try Levee
Long Sault  (pronunciation)  Long Soo
saltpetre  saltpeter
Paule try Pohl

Almonte use Almont
arboretum
Aylmer
Boucherville probably OK
Dieppe
Domenico Nacci or Nachi
Erminia
escapees
escheat
Frieda
Greenock -- seems OK
I-81
Jocelyn
kinesiology and phys. ed. or phonetic fiz ed
Leica
Maniac
Model A Ford
Myrna pronounced strangely by female voice default of epub2tts
Ontario
Rideau pronounced Rydoh in epub2tts
Rimouski
Tremblay mispronounced  try T R O M B L A Y  Tromblay
V I M Y or Vimy pronounced Veemy or Vy me? By using V I M M Y we get Vimmy.
```

homonymns.txt

```
# homonyms.txt

"read" sometimes reed, not red

bough of a tree, bow of a ship, a ribbon in a bow.

bowed down

learned vs learnt

row -- pronounced as rou not roe

lead pronounced as "leed" (led for weight)
```

numberpunctuation.txt

```
# numberpunctuation.txt

Numbers, units and punctuation give difficulties.

2/3 versus two thirds
```

```
.2 or point 2 is just 2 with pause. Not Point 2

Brackets (), [], and {} are not handled well

web addresses are often mangled

Lower case M M may be millimeters. M M M is mmm.
35mm vs 35 mm
No. 2  not number 2
cwt. versus hundredweight
```

other.txt

```
# other.txt

en-US-AndrewNeural speaker handles these correctly
escapees
steno for stenographer
`unmaking` not pronounced correctly in epub2tts
```

Dialect

Sometimes we want to write dialogue with the speaker using local pronunciation. Particular cases may be the dropping of the letter H by some working-class British speakers. This can give problems, and need phonetic replacement.

```
# dialect.txt

I'll 'ave the recording of Ave Maria.

I'll avv the recording of Ave Maria.
```

Vocalizations

Vocalizations like "oh", "er", "hmmph", "hmmm" sometimes get mangled, though epub2tts-edge with the en-US-AndrewNeural voice seems to work reasonably well.

```
# vocalizations.txt

epub2tts-edge does not seem to know how to properly enunciate some vocalizations common in human speech

The surprise expressive "oh" is often mis-said. "Oh, I really must fix that."

Disdain, as with "hmmph". Hesitation as with -- er -- and -- hmmm -- or realization with "ah" are poten
```

Audio post-processing

It is sometimes necessary to post-process the audio files.

Conversion to mp3

Since many older devices cannot play m4b files, but are intended for mp3 files, we may need to convert. Unfortunately, most such devices do NOT allow the user to save the position of a paused play. Therefore it is useful to have an audiobook in mp3 form in multiple smaller files. One way to do this is to build the audiobook in pieces, and this is the recommended approach. Nevertheless, **epub2tts-edge** produces files as m4b files, and these need conversion. I do this with **ffmpeg** and batch process them with a small (Linux / bash) command-line script.

```
#!/bin/bash
#
# convm4b
mkdir newfiles
for f in *.m4b; do
ffmpeg -i "$f" -codec:v copy -codec:a libmp3lame -q:a 2 newfiles/"${f%.m4b}.mp3"
done
```

Open question What are Windows / Mac equivalents?

Breaking a large mp3 file into pieces

I have also had to break up a large file (converted from the m4b of an entire novel). For this I used the **audacity** software which is available on most common platforms. This is a graphical user interface program allowing for sections of an audio file to be cut and pasted and exported in various formats, including mp3.

Concatenating audio files

Sometimes the mp3 files are really too small, so the user needs to keep loading them. In that case we want to concatenate several files. **ffmpeg** can do this using a text file containing a list of the files to be processed in the order they appear. An example command is

```
ffmpeg -f concat -i mylist.txt -c copy output.wav
```

which will produce a file of type wav.

File mylist.txt must have lines of the form

```
file 'myfile.type'
```

where type is one of acceptable types to ffmpeg, which does not include m4b unfortunately, even though we used ffmpeg above to do conversions.

Another tool is **mp3wrap**, which uses a command

```
mp3wrap [options]  OUTPUTFILE  f1.mp3  f2.mp3  [f3.mp3]
```

Unfortunately, there seems to be a limit to the number of files that can be included, but the number does not appear to be documented. One also needs to be careful to ensure the ordering of the files.

I was unable to get the program **mp3cat** from https://github.com/dmulholl/mp3cat to work at all.

m4b-tool

https://github.com/sandreas/m4b-tool claims to do many of the operations we might wish to do to render audiobooks in convenient sizes etc. Unfortunately, I found

- the software was NOT easy to install.
- I needed to get the `docker` software, then try to install m4b-tool instance,
- I needed to create an alias to run the software (which disappeared over reboots, so I needed to explicitly save to `.bashrc`).
- I needed to add myself to the docker group before the tool would run.
- Once installed, it did NOT work in all directories e.g., the shared directory of the Linux Mint 22 virtual machine in which I run all audiobook work to keep software intact.
- Once I did get it to start running, I got sporadic errors depending on where I started the software relative to input and output locations.
- It seems the software is fragile concerning cover images.

- I did manage to get the software to “sort of” work, with reported errors, though the output was satisfactory.

Changing speed or tempo of audio

audacity allows the speed or tempo (pitch may change with one of these) to be altered. For some texts and some “speakers” (i.e., voices) a slowdown of approximately 8% made the material easier to comprehend.

Special Tools

To process the main source text for the second-stage of epub2tts-edge, I have written several small scripts to perform particular tasks. I have written these in **R** or in the Linux **bash** script, but those choices were purely for convenience. The tasks are relatively straightforward, but tedious to carry out without the automation.

Section numbering

Most of my books do not have regular chapters, but do have sections. These are typically, but not always, delimited by “* * *“. I use a text editor and replace these (and any other suitable breakpoints) with

```
# Part ??
```

An R script numpart.R (see Appendices) is then used (after editing filename specifications within the script) to auto-replace the “??” with suitable numbers. Note that we want these numbers to all be the same length for processing to audio files, so leading zeros are strongly recommended in the numbering, and the script prepares such numbers.

The section dividers provided by epub2tts-edge can also be edited to provide useful output. I have used lines like

```
# The title of this book
```

or

```
# Capsule comment
```

with the latter for the “teaser” comment for the book.

Section splitting

It is useful to have individual files of text for each section of a book to convert to audiobook form. Such files are smaller and more quickly processed, so errors or glitches can be repaired and the files re-processed. File **splitnovel.R** in the Appendices does this.

Notes:

- we may wish to consolidate several sections after splitting so the audio playback files are of a reasonable size. This can be done with script **combpart.R** in the Appendices. Titles for the consolidated sections could be titled withing the text source for conversion by adding title lines in the form

```
# Module A of My Excellent Book
```

- there are issues with sections that are named versus ‘Part 043’ style, in that mixing these can be awkward. A suggestion is to use a style

```
# Part 043 -- The plot thickens
```

A recipe

1) Extracting the cover and the text

```
epub2tts-edge mybook.epub
```

should give the cover image in file `mybook.png` and "text" in file `mybook.txt`.

The program **epub-cover** can be used to extract the cover image. This is part of the Ruby package epub-parser from https://gitlab.com/KitaitiMakoto/epub-parser. The syntax

```
epub-cover mybook.epub
```

will output whatever cover file is embedded in the epub, which may have a name somewhat different from the epub title, and may be a JPEG file. Note that one may choose to use epub2tts-edge to get mybook.png and then ignore the text file produced in case it is missing material that was in monospaced font.

Note that the `ebook-edit` of the **Calibre** ebook software or `sigil` can both export the cover image.

2) Add Title and Author

The first lines of `mybook.txt`, which can be copied to a file `mybookheader.txt`, should be something like

```
Title: My fabulous book
Author: Me A. Writer
```

Moreover, the software will try to create "parts" denoted with the hash (#) symbol followed by a title, but this will depend to a large extent on the epub content structure.

3) Check for monospaced material and insert

Unfortunately, as we have noted, **epub2tts-edge**, using the Python library **Beautiful Soup** (according to Christopher Aedo), fails to extract any material in a monospaced font.

ebook-convert does the job reasonably well, but we may need to ensure the Title/Author header of the type `mybookheader.txt` mentioned above is present. Moreover, some pauses in the address and date on letters may need to be inserted as punctuation or line endings.

The issue of material which may be dropped by epub to text conversion is, I believe, likely to remain for some time. This means that users/authors still need to check the output of the processes in the epub to audio conversion.

4) Consolidate paragraphs

- Run the program **novelparas.R** to ensure text paragraphs are consolidated for the audio output. Even if the text has been extracted with epub2tts-edge, there may be manually inserted monospace text that requires paragraph consolidation to remove unwanted line endings.

5) Edit for section divisions and their titles

Whatever divisions are present (I divide sections with `* * *`) can be edited to the form `# Part ??`, or such divisions can be inserted. Blank lines above and below the division are recommended. This can easily be done with most text editors.

6) Section numbering

Run **numpart.R** to get sections numbered. This replaces the `??` in `# Part ??` with appropriately sequenced numbers.

7) Standard cleanup of the text file

Besides the Author/Title header and the monospaced material, it is useful to check the text (before splitting by sections) to fix some issues.

- **Front matter** often has symbols, such as copyright, or Internet addresses for web pages or email. These may need attention to get them read in a comprehensible way by the artificial voice.

- Remove periods after initials in author name(s) to avoid unnecessary pause.
- Remove the copyright symbol (©), if necessary expanding to the word "copyright".
- Put spaces in email addresses e.g. `j.smith@isp.org` becomes `j . s m i t h @ i s p . o r g`. Similarly expand abbreviations and postal codes so they are not read as if they are words.
- Expand province or state abbreviations.
- Some paragraphs as text will not offend the book reader, but when in audio form will not have the right pauses, so need splitting or combining for better flow when read out loud.
- Check that there is a line ending for the final line of the mybook.txt file. This seemingly trivial change bit me in one conversion effort.
- Add section dividers for the title page(s), the capsule comment and any preamble. Optionally expand or diminish the section divisions.

8) Phonetic corrections

Proper names, dialect, homonyms and similar issues for text to speech should be corrected. To do this, I wrote a small program **wordsub.R** as a prototype for this operation. This program needs a set of source / replacement choices. We shall suppose this is in the form of lines

```
old (tab)  new
```

This assumes 1-1 choices. We will manually deal with situations where there is more than one choice using a text editor. **wordsub.R** uses a named file **phoneticsub.txt** which has entries as described above. To save computing time, it is worthwhile having a "master" file, call is **phoneticsub.full.txt** with all possible replacements. This file can be copied to **phoneticsub.txt** and unnecessary entry lines simply removed.

9) Section splitting

Run **splitnovel.R** to divide the source text into individual sections. This is helpful later when or if we discover that the conversion of text to speech has made an error or glitch. We can then attempt to correct the issue for just the single (small?) section. Once the text is adjusted appropriately, we can then re-work the section, if necessary inserting it into a consolidated module where it belongs.

10) (Optional) Text section consolidation

To provide suitably sized audio files, we could consolidate the text sections into modules using **combpart.R**.

However, it is likely more human-efficient to create small audio files then put these together with the help of a suitable program such as `joinmp3.R`.

11) Conversion to speech

The script **runedge.R** is intended to process the text modules (or sections) and produce files of type m4b. Note that this script is currently set up to be first edited to provide the right filename root for the modules and the correct cover image.

This stage of the process takes quite a long time – typically of the order of an hour.

12) Conversion to mp3

Unfortunately, the tools for consolidation of sets of small m4b files into larger blocks are not, as far as we have so far found at least, reliable. There is a tool `m4b-tool` from https://github.com/sandreas/m4b-tool, but we could not get a satisfactory result.

The `ffmpeg` program does offer concatenation of mp3 files, and also will convert m4b files to mp3 (though apparently not concatenate m4b files). We set this operation up as a script `convm4b.sh` which is given in an appendix. (Note that the scripts may be still under development.)

13) Consolidation of audio blocks

We want our finalized audiobooks to be in parts small enough for users to navigate, that is, to move to particular parts. Our experience is that this means blocks of around one to two hours, or between five and ten blocks for a book of reasonable length. With files of type m4b users can generally, with appropriate player software or devices, move quickly between sections. Still, we believe division of audiobooks into modestly-sized blocks of audio time is sensible.

The program `joinmp3.R` calculates which small mp3 files to combine, then combines them, and then does some other operations as described in subsequent steps.

14) Convert consolidated mp3 blocks to m4b format

`joinmp3.R` also will convert the consolidated mp3 audio files to m4b format. It uses `ffmpeg` to do this.

15) Add a cover page to the files

`joinmp3.R` will also add cover images to each of the large block files, both mp3 and m4b.

The program **AtomicParsley** (https://github.com/wez/atomicparsley) can be used to insert a different cover image into m4b (and some other audio) files. One syntax is

```
cp myfile.m4b tmp.m4b
AtomicParsley tmp.m4b --artwork REMOVE_ALL
AtomicParsley tmp.m4b --artwork cover.jpg -o tmpx.m4b
mv tmpx.m4b myfile.m4b
rm tmp.m4b
```

However, we carefully avoid having an image in the mp3 blocks, so can simply add the "–artwork" and also author metadata (with the "–artist" tag).

Atomic Parsley does not, unfortunately, add images to mp3 files, and we do this (in `joinmp3.R`) with `ffmpeg`. It is important to note that this NOT be performed before conversion of the mp3 files to m4b. We spent the better part of an afternoon trying different methods to insert the cover image and achieve the appropriate mp3 to m4b conversion before a careful test established this.

There are some other tools useful for removing or inserting images, of which the `eyed3` package is useful (with command `eyeD3`).

16) (Optionally) Adjust audio file characteristics

The program **audacity** is useful to change speed, tempo or pitch of audio files. Some text-to-speech models create output that is very quick. Slower and lower-pitched output may be helpful to listeners. We found an approximate 8 percent slowing of playback (with concurrent pitch reduction) was helpful with some of the voices.

Open needs

- The loss of material in the initial conversion from epub to text is a serious issue. A fix for this first stage of epub2tts-edge so that monospaced (or other problematic) material is always included. An alternative would be tools to work with other epub to text translation tools such as **pandoc** or **epub2txt** mentioned above that also establish the paragraph structure without line endings. At the moment a more or less plain text editor is used for this work.

- Until and unless the large language models can cope with various proper names, homographs and dialect, tools to streamline the replacement of selected words with phonetic substitutes will be welcome. It would be helpful to have a GUI editor that highlights target words and offers replacements. Such tools already exist for spelling checkers, so this is not an idle possibility.
- epub2tts-edge needs an active internet connection to function. There are a number of situations where it would be important to be able to prepare audio output from text WITHOUT the need for a connection to the Microsoft service. Moreover, it is by no means certain that this service will be continued, or continued on the same basis as at present.

Conclusions

This article underlines the reality that we are still in the early stages of large-scale text to speech, at least in the open-source world.

- We see a large number of small but important "glitches" in the text-to-speech engines. While not mentioned much in the article, the Coqui AI TTS was tried, but found wanting.
- Workarounds have been found for most of the glitches discovered. These have, however, requires quite a lot of work.
- Pronunciation and emphasis issues still are found. These are not major, but may still sound jarring to the listener.
- The lack of a way to tell the TTS engine what language a segment of text should be used is a serious fault and causes a number of issues. It would also be useful to be able to specify emphasized or softer speech, or to specify the "speaker" voice for novels with more than one narrator, such as https://www.obooko.com/free-historical-fiction-books/dodging-the-potholes .
- Some books require multiple voices. This is relatively easy to arrange if the text blocks are broken up, but tags to allow the "speaker" to be changed would be helpful.
- The **edge** TTS engine require the computer processing text to be internet- connected. This is a serious obstacle for some applications. It also suggests that users may be vulnerable to being held hostage by Microsoft should they suddenly impose a charging model on the parts of edge-TTS now open.

Appendix 1: epub2tts-edge voices

There is a list of voices at the end of this document. This was obtained using a feature of the `edge-tts` project from https://github.com/rany2/edge-tts. This can be installed under Python 3.12 in the Linux Mint 22 Wilma operating system with the terminal command

```
pipx install edge-tts
```

Then the command

```
edge-tts --list-voices > edgevoices.txt
```

will create a text file of documented voices.

How to get audio samples

The program **get-edge-tts-samples.py** shows how to get samples of audio for different edge-tts speakers.

```
#!/usr/bin/env python3

"""
Basic example of edge_tts usage.
"""
```

```
import asyncio
import edge_tts

NAMES = ['en-US-AnaNeural', 'en-US-AndrewNeural', 'en-US-AriaNeural', 'en-US-AvaNeural', \
 'en-US-BrianNeural', 'en-US-ChristopherNeural', 'en-US-EmmaNeural', 'en-US-EricNeural', \
 'en-US-GuyNeural', 'en-US-JennyNeural', 'en-US-MichelleNeural', 'en-US-RogerNeural', \
 'en-US-SteffanNeural', 'de-DE-FlorianMultilingualNeural', 'de-DE-SeraphinaMultilingualNeural',\
 'en-US-AndrewMultilingualNeural', 'en-US-AvaMultilingualNeural', 'en-US-BrianMultilingualNeural',\
 'en-US-EmmaMultilingualNeural', 'fr-FR-RemyMultilingualNeural', 'fr-FR-VivienneMultilingualNeural']

TEXT = "This is a longer string that I am sending to text to speech, and using the python module directl

async def amain() -> None:
    """Main function"""
    for name in NAMES:
        output = f"sample-{name}.wav"
        print(f"Saving {output}")
        communicate = edge_tts.Communicate(TEXT, name)
        await communicate.save(output)


if __name__ == "__main__":
    loop = asyncio.get_event_loop_policy().get_event_loop()
    try:
        loop.run_until_complete(amain())
    finally:
        loop.close()
```

There are a number of lists of edge-tts voices. One is at https://github.com/rany2/edge-tts/blob/master/RE ADME.md

Appendix A: Possible troubles

The following text is believed to be likely to give problems so that audio output will need to be checked. Different voices appear to have different language models, so some will “work” while others will not.

```
# Oddbods.txt

Nene -- This is an early Rolls Royce jet engine using a centrifugal complressor of the
         Whittle style. It was used in the De Havilland Vampire and a few other early
         jet aircraft.  Better to use Neen to get the right pronunciation.
Sterling currency expression 17/6 is better written 17 and 6.
ha'penny  as hah penny, not haypenny
"wan" --> W A N
Marthe -- more like Martha than Mart
Feltham  "ham"
French mis-pronounced. "687 25th Avenue"
Paule, Lise (more like Liss)
souper
"Sacrément!"
dommage
compris
cintres
"C’est un très grand réfrigérateur."
tarte au sucre
```

```
erable
Houle
Decarie
Ste Catherine
Outrement
Ultramontes
Universit'e
maudit
Anges
Sinterklaas
Bolduc
Pieds nus dans l'aube
Sault Ste Marie,   Regina
Metis
Mon Dieu
V-1 may be better as V 1
```

Appendix B: Software tools associated with this article

convm4b.sh – convert m4b files to mp3

```
convm4b.sh #!/bin/bash
convm4b.sh #
convm4b.sh # convm4b
convm4b.sh mkdir newfiles
convm4b.sh for f in *.m4b; do ffmpeg -i "$f" -codec:v copy -codec:a libmp3lame -q:a 2 newfiles/"${f%.m4
```

combpart.R – combine individual book sections

The purpose of this script is to create texts that will have a reasonable audio file duration, particularly when in mp3 form, which is more difficult in most cases to position or resume playing. The number of parts to combine is set at `np <- 10` in the present version.

```
combpart.R Warning in readLines("../baseabsw/combpart.R"): incomplete final line found on
combpart.R '../baseabsw/combpart.R'
```

```
combpart.R # combpart.R
combpart.R # Combine parts of text source for audiobooks
combpart.R ifnroot <- readline("input file root:")
combpart.R np <- 10 # can change to number of parts to combine
combpart.R
combpart.R pat<-paste("^",ifnroot,"...\\.txt",sep='')
combpart.R ll <- list.files(pattern=pat)
combpart.R ll <- sort(ll)
combpart.R print(ll)
combpart.R tmp<-readline("cont.")
combpart.R nl <- length(ll)
combpart.R
combpart.R f0<-ll[1]
combpart.R con<-file(f0, open="r")
combpart.R tt<-readLines(con)
combpart.R close(con)
combpart.R hd<-tt[1:3]
combpart.R
combpart.R sufx<-c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N",
combpart.R           "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z")
combpart.R
combpart.R fcount<-1
combpart.R ofn<- paste(ifnroot,"cc",sufx[fcount],".txt",sep='')
combpart.R cat("Initial ofn:", ofn,"\n")
combpart.R ocon<-file(ofn, open="wt")
combpart.R writeLines(hd, ocon) # puts in header
combpart.R
combpart.R
combpart.R for (i in 1:nl) {
combpart.R   cat("Input file ", ll[i],"\n")
combpart.R   con<-file(ll[i], open="r")
combpart.R   tt<-readLines(con)
combpart.R   nt<-length(tt)
combpart.R   otxt<-tt[4:nt]
combpart.R   close(con)
combpart.R   writeLines(otxt, ocon)
combpart.R   if (i >= nl) { # finished
combpart.R      flush(ocon)
combpart.R      close(ocon)
combpart.R      break
combpart.R   } else { # check if need new output file
combpart.R     if ( (i %% np) == 0 ) { # New output file?
combpart.R       flush(ocon)
combpart.R       close(ocon)
combpart.R       fcount <- fcount + 1 # next output file
combpart.R       ofn<- paste(ifnroot,"cc",sufx[fcount],".txt",sep='')
combpart.R       cat("New ofn:", ofn,"\n")
combpart.R       ocon<-file(ofn, open="wt")
combpart.R       writeLines(hd, ocon) # puts in header
combpart.R     }
combpart.R   }
combpart.R } # end loop over files
combpart.R cat("Done!\n")
```

novelparas.R – remove single line endings to get paragraphs

epub2tts-edge will pause on line endings. Paragraphs are therefore assumed to be terminated by a line ending.

```
novelparas.R # novelparas.R
novelparas.R library(stringr)
novelparas.R # Consolidate paragraph lines in novel audiobook source texts
novelparas.R ifn<-readline("Input file name(text):")
novelparas.R con<-file(ifn, open="r")
novelparas.R ofn<-readline("Output file nam (txt):")
novelparas.R ocon<-file(ofn, open="w")
novelparas.R y<-readLines(con)
novelparas.R close(con)
novelparas.R ny<-length(y)
novelparas.R bb<-which(y=="")
novelparas.R nbb<-length(bb)
novelparas.R nbb<-nbb+1
novelparas.R ny<-length(y)+1
novelparas.R bb[nbb] <- ny # beyond end of file
novelparas.R inpos<-0
novelparas.R lb <- 1 # initial first line in para
novelparas.R for (ib in 1:nbb){ # loop over paras
novelparas.R   tpara <- ""
novelparas.R   ub <- bb[ib]-1 # last line in para
novelparas.R   for (ll in lb:ub){
novelparas.R      inpos<-ll
novelparas.R      lyn<-y[inpos]
novelparas.R      tpara <- paste(tpara,lyn," ",sep='')
novelparas.R      # May want to remove extra white space afterwards
novelparas.R   }
novelparas.R   # should get rid of terminating and extra whitespace in tpara
novelparas.R   tpara<-str_squish(tpara)
novelparas.R #   allp[ib]<-tpara
novelparas.R    writeLines(tpara, ocon)
novelparas.R    writeLines("", ocon) # Just in case
novelparas.R    lb <- ub + 1 # to reset for next para
novelparas.R } # end loop over paras
novelparas.R flush(ocon)
novelparas.R close(ocon)
```

numpart.R – number sections that have generic divider naming

This script is intended to convert dividers `# Part ??` into numbered dividers. It requires generic dividers of the form `* * *` to be converted to `# Part ??` with a text editor.

```
numpart.R # numpart.R
numpart.R # converts book that has '*  *  *' sections changed to '# Part ??'
numpart.R
numpart.R ifn <- readline("Input filename:")
numpart.R pn <- 0
numpart.R pat <- "# Part ??"
numpart.R
numpart.R con=file(ifn, open="r")
numpart.R ofn <- readline("Output file (text):")
numpart.R ofcon <- file(ofn, open="w")
numpart.R
numpart.R while ( TRUE ) {
numpart.R   line = readLines(con, n = 1)
numpart.R   if (length(line)==0) {
numpart.R     break
numpart.R   }
numpart.R   if ( substr(line, 1, 9) == pat) {
numpart.R      pn <- pn + 1
numpart.R      pnc <- as.character(pn)
numpart.R      while(nchar(pnc) < 3) {pnc<-paste("0",pnc,sep='')}
numpart.R # convert number to text with leading zeros so section names sort.
numpart.R # Suggest 3 character form, as books with more than 999 sections too long.
numpart.R      oline <- paste("# Part ",pnc,sep='')
numpart.R      writeLines(oline, ofcon)
numpart.R   } else { writeLines(line, ofcon) }
numpart.R }
numpart.R close(con)
numpart.R flush(ofcon)
numpart.R close(ofcon)
```

splitnovel.R – divide book text file into individual sections with headers

```
splitnovel.R # splitnovel.R
splitnovel.R # Split novel into sections
splitnovel.R ## ?? possibly not finishedifn <- readline("input file:")
splitnovel.R ifn <- readline("Input file name:")
splitnovel.R
splitnovel.R ofnroot <- readline("Output file root:")
splitnovel.R
splitnovel.R pat <- "# Part "
splitnovel.R npat<-nchar(pat)
splitnovel.R con=file(ifn,open="r")
splitnovel.R
splitnovel.R tt <- readLines(con)
splitnovel.R ntt<- length(tt)
splitnovel.R close(con)
splitnovel.R hd<-tt[c(1,3,4)]
splitnovel.R
splitnovel.R plines <- which(substr(tt, 1, npat)==pat)
splitnovel.R npl <- length(plines)
splitnovel.R plines<-c(plines, ntt+1) # add an upper bound at end
splitnovel.R
splitnovel.R lb <- 5 # initial start line
splitnovel.R sn <- 0
splitnovel.R snc <- as.character(sn)
splitnovel.R while(nchar(snc) < 3) {snc<-paste("0",snc,sep='')}
splitnovel.R if (snc != "000") stop("Bad number")
splitnovel.R ofn <- paste(ofnroot,snc,".txt",sep='')
splitnovel.R ofcon <- file(ofn, open="w")
splitnovel.R ub <- plines[1] - 1
splitnovel.R tfile<-tt[lb:ub]
splitnovel.R writeLines(hd, ofcon)
splitnovel.R writeLines(tfile, ofcon)
splitnovel.R writeLines("", ofcon)
splitnovel.R flush(ofcon); close(ofcon)
splitnovel.R
splitnovel.R lb<-ub+1
splitnovel.R for (ii in 1:npl){
splitnovel.R    ub<-plines[ii+1] - 1
splitnovel.R    cat(lb," ", ub,"\n")
splitnovel.R    sn<-sn+1
splitnovel.R    snc <- as.character(sn)
splitnovel.R    while(nchar(snc) < 3) {snc<-paste("0",snc,sep='')}
splitnovel.R    tfile<-tt[lb:ub]
splitnovel.R    ofn <- paste(ofnroot,snc,".txt",sep='')
splitnovel.R    ofcon <- file(ofn, open="w")
splitnovel.R    writeLines(hd, ofcon)
splitnovel.R    writeLines(tfile, ofcon)
splitnovel.R    writeLines("", ofcon)
splitnovel.R    flush(ofcon); close(ofcon)
splitnovel.R    lb<-ub+1 # update
splitnovel.R }
```

joinmp3.R – consolidate, convert and add images

`joinmp3.R` works out which files to consolidate to yield a small number of rougly equal-sized mp3 files. These are converted to m4a files which are renamed to m4b files. Then cover images are added to both sets of files.

```
joinmp3.R # joinmp3.R
joinmp3.R outp<-"FN-" # output file pattern (?? make this easy to change)
joinmp3.R mlist<-list.files(path="./mp3", pattern="*.mp3")
joinmp3.R dmlist<-paste0("mp3/",mlist)
joinmp3.R nfile<-length(dmlist)
joinmp3.R lfile<-file.size(dmlist)
joinmp3.R tsize<-sum(lfile)
joinmp3.R nblock<-5
joinmp3.R psize<-tsize/nblock
joinmp3.R cusize<-cumsum(lfile)
joinmp3.R cusize
joinmp3.R cublk<-(1:nblock)*psize
joinmp3.R cublk
joinmp3.R sb<-rep(NA,nblock+1) # to give block splits
joinmp3.R sb[1]<-0
joinmp3.R sb[nblock+1]<-nfile
joinmp3.R for( ii in 1:(nblock-1)){
joinmp3.R   sb[ii+1]<-max(which(cublk[ii] > cusize))
joinmp3.R }
joinmp3.R bcA1<-as.numeric(charToRaw("A"))-1
joinmp3.R outlist<-c()
joinmp3.R for (ii in 1:nblock){
joinmp3.R    cch<-rawToChar(as.raw(ii+bcA1))
joinmp3.R    cat("Charex=",cch,"\n")
joinmp3.R    ofn<-paste0(outp,cch,".mp3")
joinmp3.R    # ?? put in askYN here??
joinmp3.R    if (file.exists(ofn)){file.remove(ofn)}
joinmp3.R    blist<-dmlist[(sb[ii]+1):(sb[ii+1])]
joinmp3.R    if (file.exists("tlist.txt")) {file.remove("tlist.txt")}
joinmp3.R    fileConn<-file("tlist.txt")
joinmp3.R    jstart<-sb[ii]+1
joinmp3.R    jend<-sb[ii+1]
joinmp3.R    for (jj in jstart:jend){
joinmp3.R       tt<-dmlist[jj]
joinmp3.R       tt<-paste0("file '",tt,"'")
joinmp3.R       blist[jj-jstart+1]<-tt
joinmp3.R    }
joinmp3.R    print(blist)
joinmp3.R    writeLines(blist, fileConn)
joinmp3.R    close(fileConn)
joinmp3.R    cmd <- paste0("ffmpeg -f concat -safe 0 -i tlist.txt -c copy ",ofn)
joinmp3.R    system(cmd)
joinmp3.R    outlist<-c(outlist,ofn)
joinmp3.R }
joinmp3.R # Make m4a -> m4b
joinmp3.R # ?? need to get covpage from profile, author too
joinmp3.R covpage<-"WhoIsPauleCovpage.jpg"
joinmp3.R author<-'John C Nash'
joinmp3.R print(outlist)
joinmp3.R for (ff in outlist){
```

```
joinmp3.R    fnroot<-tools::file_path_sans_ext(ff)
joinmp3.R    ffa<-paste0(fnroot,".m4a")
joinmp3.R    ffb<-paste0(fnroot,".m4b")
joinmp3.R    rc<-file.copy(ff, paste0(ff,".copy"))
joinmp3.R    cmd<-paste0("ffmpeg -i ",ff," -c:a aac -b:a 192k ",ffa)
joinmp3.R    system(cmd)
joinmp3.R    file.rename(ffa,ffb) # ?? retcode?
joinmp3.R    cmd<-paste0("AtomicParsley ",ffb," --output 'out' --artist ",author," --artwork ",covpage)
joinmp3.R    system(cmd)
joinmp3.R    rc<-file.copy("out", ffb, overwrite=TRUE)
joinmp3.R    cat('file.copy("out", ffb)  returns ',rc,"\n")
joinmp3.R    rc<-file.remove("out")
joinmp3.R    cat('file.remove("out")  returns ',rc,"\n")
joinmp3.R }
joinmp3.R # cover for mp3 -- seems that if this is done BEFORE conversion to m4a, that fails
joinmp3.R for (ff in outlist) {
joinmp3.R cmd<-paste0("ffmpeg -i ",ff," -i ",covpage," -map_metadata 0 -map 0 -map 1 -acodec copy 'out.
joinmp3.R system(cmd) # ?? should we check return code
joinmp3.R rc<-file.copy("out.mp3", ff, overwrite=TRUE)
joinmp3.R cat('file.copy("out.mp3", ff)  returns ',rc,"\n")
joinmp3.R rc<-file.remove("out.mp3")
joinmp3.R cat('file.remove("out.mp3")  returns ',rc,"\n")
joinmp3.R }
```

Appendix C: List of voices usable by epub2tts-edge

```
voices Name                                Gender    ContentCategories      VoicePersonalities
voices ---------------------------------  --------  ---------------------  ---------------------------
voices af-ZA-AdriNeural                    Female    General                Friendly, Positive
voices af-ZA-WillemNeural                  Male      General                Friendly, Positive
voices am-ET-AmehaNeural                   Male      General                Friendly, Positive
voices am-ET-MekdesNeural                  Female    General                Friendly, Positive
voices ar-AE-FatimaNeural                  Female    General                Friendly, Positive
voices ar-AE-HamdanNeural                  Male      General                Friendly, Positive
voices ar-BH-AliNeural                     Male      General                Friendly, Positive
voices ar-BH-LailaNeural                   Female    General                Friendly, Positive
voices ar-DZ-AminaNeural                   Female    General                Friendly, Positive
voices ar-DZ-IsmaelNeural                  Male      General                Friendly, Positive
voices ar-EG-SalmaNeural                   Female    General                Friendly, Positive
voices ar-EG-ShakirNeural                  Male      General                Friendly, Positive
voices ar-IQ-BasselNeural                  Male      General                Friendly, Positive
voices ar-IQ-RanaNeural                    Female    General                Friendly, Positive
voices ar-JO-SanaNeural                    Female    General                Friendly, Positive
voices ar-JO-TaimNeural                    Male      General                Friendly, Positive
voices ar-KW-FahedNeural                   Male      General                Friendly, Positive
voices ar-KW-NouraNeural                   Female    General                Friendly, Positive
voices ar-LB-LaylaNeural                   Female    General                Friendly, Positive
voices ar-LB-RamiNeural                    Male      General                Friendly, Positive
voices ar-LY-ImanNeural                    Female    General                Friendly, Positive
voices ar-LY-OmarNeural                    Male      General                Friendly, Positive
voices ar-MA-JamalNeural                   Male      General                Friendly, Positive
voices ar-MA-MounaNeural                   Female    General                Friendly, Positive
voices ar-OM-AbdullahNeural                Male      General                Friendly, Positive
voices ar-OM-AyshaNeural                   Female    General                Friendly, Positive
```

voices ar-QA-AmalNeural	Female	General	Friendly, Positive
voices ar-QA-MoazNeural	Male	General	Friendly, Positive
voices ar-SA-HamedNeural	Male	General	Friendly, Positive
voices ar-SA-ZariyahNeural	Female	General	Friendly, Positive
voices ar-SY-AmanyNeural	Female	General	Friendly, Positive
voices ar-SY-LaithNeural	Male	General	Friendly, Positive
voices ar-TN-HediNeural	Male	General	Friendly, Positive
voices ar-TN-ReemNeural	Female	General	Friendly, Positive
voices ar-YE-MaryamNeural	Female	General	Friendly, Positive
voices ar-YE-SalehNeural	Male	General	Friendly, Positive
voices az-AZ-BabekNeural	Male	General	Friendly, Positive
voices az-AZ-BanuNeural	Female	General	Friendly, Positive
voices bg-BG-BorislavNeural	Male	General	Friendly, Positive
voices bg-BG-KalinaNeural	Female	General	Friendly, Positive
voices bn-BD-NabanitaNeural	Female	General	Friendly, Positive
voices bn-BD-PradeepNeural	Male	General	Friendly, Positive
voices bn-IN-BashkarNeural	Male	General	Friendly, Positive
voices bn-IN-TanishaaNeural	Female	General	Friendly, Positive
voices bs-BA-GoranNeural	Male	General	Friendly, Positive
voices bs-BA-VesnaNeural	Female	General	Friendly, Positive
voices ca-ES-EnricNeural	Male	General	Friendly, Positive
voices ca-ES-JoanaNeural	Female	General	Friendly, Positive
voices cs-CZ-AntoninNeural	Male	General	Friendly, Positive
voices cs-CZ-VlastaNeural	Female	General	Friendly, Positive
voices cy-GB-AledNeural	Male	General	Friendly, Positive
voices cy-GB-NiaNeural	Female	General	Friendly, Positive
voices da-DK-ChristelNeural	Female	General	Friendly, Positive
voices da-DK-JeppeNeural	Male	General	Friendly, Positive
voices de-AT-IngridNeural	Female	General	Friendly, Positive
voices de-AT-JonasNeural	Male	General	Friendly, Positive
voices de-CH-JanNeural	Male	General	Friendly, Positive
voices de-CH-LeniNeural	Female	General	Friendly, Positive
voices de-DE-AmalaNeural	Female	General	Friendly, Positive
voices de-DE-ConradNeural	Male	General	Friendly, Positive
voices de-DE-FlorianMultilingualNeural	Male	General	Friendly, Positive
voices de-DE-KatjaNeural	Female	General	Friendly, Positive
voices de-DE-KillianNeural	Male	General	Friendly, Positive
voices de-DE-SeraphinaMultilingualNeural	Female	General	Friendly, Positive
voices el-GR-AthinaNeural	Female	General	Friendly, Positive
voices el-GR-NestorasNeural	Male	General	Friendly, Positive
voices en-AU-NatashaNeural	Female	General	Friendly, Positive
voices en-AU-WilliamNeural	Male	General	Friendly, Positive
voices en-CA-ClaraNeural	Female	General	Friendly, Positive
voices en-CA-LiamNeural	Male	General	Friendly, Positive
voices en-GB-LibbyNeural	Female	General	Friendly, Positive
voices en-GB-MaisieNeural	Female	General	Friendly, Positive
voices en-GB-RyanNeural	Male	General	Friendly, Positive
voices en-GB-SoniaNeural	Female	General	Friendly, Positive
voices en-GB-ThomasNeural	Male	General	Friendly, Positive
voices en-HK-SamNeural	Male	General	Friendly, Positive
voices en-HK-YanNeural	Female	General	Friendly, Positive
voices en-IE-ConnorNeural	Male	General	Friendly, Positive
voices en-IE-EmilyNeural	Female	General	Friendly, Positive
voices en-IN-NeerjaExpressiveNeural	Female	General	Friendly, Positive

```
voices en-IN-NeerjaNeural                 Female    General                Friendly, Positive
voices en-IN-PrabhatNeural                Male      General                Friendly, Positive
voices en-KE-AsiliaNeural                 Female    General                Friendly, Positive
voices en-KE-ChilembaNeural               Male      General                Friendly, Positive
voices en-NG-AbeoNeural                   Male      General                Friendly, Positive
voices en-NG-EzinneNeural                 Female    General                Friendly, Positive
voices en-NZ-MitchellNeural               Male      General                Friendly, Positive
voices en-NZ-MollyNeural                  Female    General                Friendly, Positive
voices en-PH-JamesNeural                  Male      General                Friendly, Positive
voices en-PH-RosaNeural                   Female    General                Friendly, Positive
voices en-SG-LunaNeural                   Female    General                Friendly, Positive
voices en-SG-WayneNeural                  Male      General                Friendly, Positive
voices en-TZ-ElimuNeural                  Male      General                Friendly, Positive
voices en-TZ-ImaniNeural                  Female    General                Friendly, Positive
voices en-US-AnaNeural                    Female    Cartoon, Conversation  Cute
voices en-US-AndrewMultilingualNeural     Male      Conversation, Copilot  Warm, Confident, Authentic,
voices en-US-AndrewNeural                 Male      Conversation, Copilot  Warm, Confident, Authentic,
voices en-US-AriaNeural                   Female    News, Novel            Positive, Confident
voices en-US-AvaMultilingualNeural        Female    Conversation, Copilot  Expressive, Caring, Pleasant
voices en-US-AvaNeural                    Female    Conversation, Copilot  Expressive, Caring, Pleasant
voices en-US-BrianMultilingualNeural      Male      Conversation, Copilot  Approachable, Casual, Sincer
voices en-US-BrianNeural                  Male      Conversation, Copilot  Approachable, Casual, Sincer
voices en-US-ChristopherNeural            Male      News, Novel            Reliable, Authority
voices en-US-EmmaMultilingualNeural       Female    Conversation, Copilot  Cheerful, Clear, Conversatio
voices en-US-EmmaNeural                   Female    Conversation, Copilot  Cheerful, Clear, Conversatio
voices en-US-EricNeural                   Male      News, Novel            Rational
voices en-US-GuyNeural                    Male      News, Novel            Passion
voices en-US-JennyNeural                  Female    General                Friendly, Considerate, Comfo
voices en-US-MichelleNeural               Female    News, Novel            Friendly, Pleasant
voices en-US-RogerNeural                  Male      News, Novel            Lively
voices en-US-SteffanNeural                Male      News, Novel            Rational
voices en-ZA-LeahNeural                   Female    General                Friendly, Positive
voices en-ZA-LukeNeural                   Male      General                Friendly, Positive
voices es-AR-ElenaNeural                  Female    General                Friendly, Positive
voices es-AR-TomasNeural                  Male      General                Friendly, Positive
voices es-BO-MarceloNeural                Male      General                Friendly, Positive
voices es-BO-SofiaNeural                  Female    General                Friendly, Positive
voices es-CL-CatalinaNeural               Female    General                Friendly, Positive
voices es-CL-LorenzoNeural                Male      General                Friendly, Positive
voices es-CO-GonzaloNeural                Male      General                Friendly, Positive
voices es-CO-SalomeNeural                 Female    General                Friendly, Positive
voices es-CR-JuanNeural                   Male      General                Friendly, Positive
voices es-CR-MariaNeural                  Female    General                Friendly, Positive
voices es-CU-BelkysNeural                 Female    General                Friendly, Positive
voices es-CU-ManuelNeural                 Male      General                Friendly, Positive
voices es-DO-EmilioNeural                 Male      General                Friendly, Positive
voices es-DO-RamonaNeural                 Female    General                Friendly, Positive
voices es-EC-AndreaNeural                 Female    General                Friendly, Positive
voices es-EC-LuisNeural                   Male      General                Friendly, Positive
voices es-ES-AlvaroNeural                 Male      General                Friendly, Positive
voices es-ES-ElviraNeural                 Female    General                Friendly, Positive
voices es-ES-XimenaNeural                 Female    General                Friendly, Positive
voices es-GQ-JavierNeural                 Male      General                Friendly, Positive
voices es-GQ-TeresaNeural                 Female    General                Friendly, Positive
```

voices es-GT-AndresNeural	Male	General	Friendly, Positive
voices es-GT-MartaNeural	Female	General	Friendly, Positive
voices es-HN-CarlosNeural	Male	General	Friendly, Positive
voices es-HN-KarlaNeural	Female	General	Friendly, Positive
voices es-MX-DaliaNeural	Female	General	Friendly, Positive
voices es-MX-JorgeNeural	Male	General	Friendly, Positive
voices es-NI-FedericoNeural	Male	General	Friendly, Positive
voices es-NI-YolandaNeural	Female	General	Friendly, Positive
voices es-PA-MargaritaNeural	Female	General	Friendly, Positive
voices es-PA-RobertoNeural	Male	General	Friendly, Positive
voices es-PE-AlexNeural	Male	General	Friendly, Positive
voices es-PE-CamilaNeural	Female	General	Friendly, Positive
voices es-PR-KarinaNeural	Female	General	Friendly, Positive
voices es-PR-VictorNeural	Male	General	Friendly, Positive
voices es-PY-MarioNeural	Male	General	Friendly, Positive
voices es-PY-TaniaNeural	Female	General	Friendly, Positive
voices es-SV-LorenaNeural	Female	General	Friendly, Positive
voices es-SV-RodrigoNeural	Male	General	Friendly, Positive
voices es-US-AlonsoNeural	Male	General	Friendly, Positive
voices es-US-PalomaNeural	Female	General	Friendly, Positive
voices es-UY-MateoNeural	Male	General	Friendly, Positive
voices es-UY-ValentinaNeural	Female	General	Friendly, Positive
voices es-VE-PaolaNeural	Female	General	Friendly, Positive
voices es-VE-SebastianNeural	Male	General	Friendly, Positive
voices et-EE-AnuNeural	Female	General	Friendly, Positive
voices et-EE-KertNeural	Male	General	Friendly, Positive
voices fa-IR-DilaraNeural	Female	General	Friendly, Positive
voices fa-IR-FaridNeural	Male	General	Friendly, Positive
voices fi-FI-HarriNeural	Male	General	Friendly, Positive
voices fi-FI-NooraNeural	Female	General	Friendly, Positive
voices fil-PH-AngeloNeural	Male	General	Friendly, Positive
voices fil-PH-BlessicaNeural	Female	General	Friendly, Positive
voices fr-BE-CharlineNeural	Female	General	Friendly, Positive
voices fr-BE-GerardNeural	Male	General	Friendly, Positive
voices fr-CA-AntoineNeural	Male	General	Friendly, Positive
voices fr-CA-JeanNeural	Male	General	Friendly, Positive
voices fr-CA-SylvieNeural	Female	General	Friendly, Positive
voices fr-CA-ThierryNeural	Male	General	Friendly, Positive
voices fr-CH-ArianeNeural	Female	General	Friendly, Positive
voices fr-CH-FabriceNeural	Male	General	Friendly, Positive
voices fr-FR-DeniseNeural	Female	General	Friendly, Positive
voices fr-FR-EloiseNeural	Female	General	Friendly, Positive
voices fr-FR-HenriNeural	Male	General	Friendly, Positive
voices fr-FR-RemyMultilingualNeural	Male	General	Friendly, Positive
voices fr-FR-VivienneMultilingualNeural	Female	General	Friendly, Positive
voices ga-IE-ColmNeural	Male	General	Friendly, Positive
voices ga-IE-OrlaNeural	Female	General	Friendly, Positive
voices gl-ES-RoiNeural	Male	General	Friendly, Positive
voices gl-ES-SabelaNeural	Female	General	Friendly, Positive
voices gu-IN-DhwaniNeural	Female	General	Friendly, Positive
voices gu-IN-NiranjanNeural	Male	General	Friendly, Positive
voices he-IL-AvriNeural	Male	General	Friendly, Positive
voices he-IL-HilaNeural	Female	General	Friendly, Positive
voices hi-IN-MadhurNeural	Male	General	Friendly, Positive

voices	hi-IN-SwaraNeural	Female	General	Friendly, Positive
voices	hr-HR-GabrijelaNeural	Female	General	Friendly, Positive
voices	hr-HR-SreckoNeural	Male	General	Friendly, Positive
voices	hu-HU-NoemiNeural	Female	General	Friendly, Positive
voices	hu-HU-TamasNeural	Male	General	Friendly, Positive
voices	id-ID-ArdiNeural	Male	General	Friendly, Positive
voices	id-ID-GadisNeural	Female	General	Friendly, Positive
voices	is-IS-GudrunNeural	Female	General	Friendly, Positive
voices	is-IS-GunnarNeural	Male	General	Friendly, Positive
voices	it-IT-DiegoNeural	Male	General	Friendly, Positive
voices	it-IT-ElsaNeural	Female	General	Friendly, Positive
voices	it-IT-GiuseppeMultilingualNeural	Male	General	Friendly, Positive
voices	it-IT-IsabellaNeural	Female	General	Friendly, Positive
voices	iu-Cans-CA-SiqiniqNeural	Female	General	Friendly, Positive
voices	iu-Cans-CA-TaqqiqNeural	Male	General	Friendly, Positive
voices	iu-Latn-CA-SiqiniqNeural	Female	General	Friendly, Positive
voices	iu-Latn-CA-TaqqiqNeural	Male	General	Friendly, Positive
voices	ja-JP-KeitaNeural	Male	General	Friendly, Positive
voices	ja-JP-NanamiNeural	Female	General	Friendly, Positive
voices	jv-ID-DimasNeural	Male	General	Friendly, Positive
voices	jv-ID-SitiNeural	Female	General	Friendly, Positive
voices	ka-GE-EkaNeural	Female	General	Friendly, Positive
voices	ka-GE-GiorgiNeural	Male	General	Friendly, Positive
voices	kk-KZ-AigulNeural	Female	General	Friendly, Positive
voices	kk-KZ-DauletNeural	Male	General	Friendly, Positive
voices	km-KH-PisethNeural	Male	General	Friendly, Positive
voices	km-KH-SreymomNeural	Female	General	Friendly, Positive
voices	kn-IN-GaganNeural	Male	General	Friendly, Positive
voices	kn-IN-SapnaNeural	Female	General	Friendly, Positive
voices	ko-KR-HyunsuMultilingualNeural	Male	General	Friendly, Positive
voices	ko-KR-InJoonNeural	Male	General	Friendly, Positive
voices	ko-KR-SunHiNeural	Female	General	Friendly, Positive
voices	lo-LA-ChanthavongNeural	Male	General	Friendly, Positive
voices	lo-LA-KeomanyNeural	Female	General	Friendly, Positive
voices	lt-LT-LeonasNeural	Male	General	Friendly, Positive
voices	lt-LT-OnaNeural	Female	General	Friendly, Positive
voices	lv-LV-EveritaNeural	Female	General	Friendly, Positive
voices	lv-LV-NilsNeural	Male	General	Friendly, Positive
voices	mk-MK-AleksandarNeural	Male	General	Friendly, Positive
voices	mk-MK-MarijaNeural	Female	General	Friendly, Positive
voices	ml-IN-MidhunNeural	Male	General	Friendly, Positive
voices	ml-IN-SobhanaNeural	Female	General	Friendly, Positive
voices	mn-MN-BataaNeural	Male	General	Friendly, Positive
voices	mn-MN-YesuiNeural	Female	General	Friendly, Positive
voices	mr-IN-AarohiNeural	Female	General	Friendly, Positive
voices	mr-IN-ManoharNeural	Male	General	Friendly, Positive
voices	ms-MY-OsmanNeural	Male	General	Friendly, Positive
voices	ms-MY-YasminNeural	Female	General	Friendly, Positive
voices	mt-MT-GraceNeural	Female	General	Friendly, Positive
voices	mt-MT-JosephNeural	Male	General	Friendly, Positive
voices	my-MM-NilarNeural	Female	General	Friendly, Positive
voices	my-MM-ThihaNeural	Male	General	Friendly, Positive
voices	nb-NO-FinnNeural	Male	General	Friendly, Positive
voices	nb-NO-PernilleNeural	Female	General	Friendly, Positive

voices ne-NP-HemkalaNeural	Female	General	Friendly, Positive
voices ne-NP-SagarNeural	Male	General	Friendly, Positive
voices nl-BE-ArnaudNeural	Male	General	Friendly, Positive
voices nl-BE-DenaNeural	Female	General	Friendly, Positive
voices nl-NL-ColetteNeural	Female	General	Friendly, Positive
voices nl-NL-FennaNeural	Female	General	Friendly, Positive
voices nl-NL-MaartenNeural	Male	General	Friendly, Positive
voices pl-PL-MarekNeural	Male	General	Friendly, Positive
voices pl-PL-ZofiaNeural	Female	General	Friendly, Positive
voices ps-AF-GulNawazNeural	Male	General	Friendly, Positive
voices ps-AF-LatifaNeural	Female	General	Friendly, Positive
voices pt-BR-AntonioNeural	Male	General	Friendly, Positive
voices pt-BR-FranciscaNeural	Female	General	Friendly, Positive
voices pt-BR-ThalitaMultilingualNeural	Female	General	Friendly, Positive
voices pt-PT-DuarteNeural	Male	General	Friendly, Positive
voices pt-PT-RaquelNeural	Female	General	Friendly, Positive
voices ro-RO-AlinaNeural	Female	General	Friendly, Positive
voices ro-RO-EmilNeural	Male	General	Friendly, Positive
voices ru-RU-DmitryNeural	Male	General	Friendly, Positive
voices ru-RU-SvetlanaNeural	Female	General	Friendly, Positive
voices si-LK-SameeraNeural	Male	General	Friendly, Positive
voices si-LK-ThiliniNeural	Female	General	Friendly, Positive
voices sk-SK-LukasNeural	Male	General	Friendly, Positive
voices sk-SK-ViktoriaNeural	Female	General	Friendly, Positive
voices sl-SI-PetraNeural	Female	General	Friendly, Positive
voices sl-SI-RokNeural	Male	General	Friendly, Positive
voices so-SO-MuuseNeural	Male	General	Friendly, Positive
voices so-SO-UbaxNeural	Female	General	Friendly, Positive
voices sq-AL-AnilaNeural	Female	General	Friendly, Positive
voices sq-AL-IlirNeural	Male	General	Friendly, Positive
voices sr-RS-NicholasNeural	Male	General	Friendly, Positive
voices sr-RS-SophieNeural	Female	General	Friendly, Positive
voices su-ID-JajangNeural	Male	General	Friendly, Positive
voices su-ID-TutiNeural	Female	General	Friendly, Positive
voices sv-SE-MattiasNeural	Male	General	Friendly, Positive
voices sv-SE-SofieNeural	Female	General	Friendly, Positive
voices sw-KE-RafikiNeural	Male	General	Friendly, Positive
voices sw-KE-ZuriNeural	Female	General	Friendly, Positive
voices sw-TZ-DaudiNeural	Male	General	Friendly, Positive
voices sw-TZ-RehemaNeural	Female	General	Friendly, Positive
voices ta-IN-PallaviNeural	Female	General	Friendly, Positive
voices ta-IN-ValluvarNeural	Male	General	Friendly, Positive
voices ta-LK-KumarNeural	Male	General	Friendly, Positive
voices ta-LK-SaranyaNeural	Female	General	Friendly, Positive
voices ta-MY-KaniNeural	Female	General	Friendly, Positive
voices ta-MY-SuryaNeural	Male	General	Friendly, Positive
voices ta-SG-AnbuNeural	Male	General	Friendly, Positive
voices ta-SG-VenbaNeural	Female	General	Friendly, Positive
voices te-IN-MohanNeural	Male	General	Friendly, Positive
voices te-IN-ShrutiNeural	Female	General	Friendly, Positive
voices th-TH-NiwatNeural	Male	General	Friendly, Positive
voices th-TH-PremwadeeNeural	Female	General	Friendly, Positive
voices tr-TR-AhmetNeural	Male	General	Friendly, Positive
voices tr-TR-EmelNeural	Female	General	Friendly, Positive

voices uk-UA-OstapNeural	Male	General	Friendly, Positive
voices uk-UA-PolinaNeural	Female	General	Friendly, Positive
voices ur-IN-GulNeural	Female	General	Friendly, Positive
voices ur-IN-SalmanNeural	Male	General	Friendly, Positive
voices ur-PK-AsadNeural	Male	General	Friendly, Positive
voices ur-PK-UzmaNeural	Female	General	Friendly, Positive
voices uz-UZ-MadinaNeural	Female	General	Friendly, Positive
voices uz-UZ-SardorNeural	Male	General	Friendly, Positive
voices vi-VN-HoaiMyNeural	Female	General	Friendly, Positive
voices vi-VN-NamMinhNeural	Male	General	Friendly, Positive
voices zh-CN-XiaoxiaoNeural	Female	News, Novel	Warm
voices zh-CN-XiaoyiNeural	Female	Cartoon, Novel	Lively
voices zh-CN-YunjianNeural	Male	Sports, Novel	Passion
voices zh-CN-YunxiNeural	Male	Novel	Lively, Sunshine
voices zh-CN-YunxiaNeural	Male	Cartoon, Novel	Cute
voices zh-CN-YunyangNeural	Male	News	Professional, Reliable
voices zh-CN-liaoning-XiaobeiNeural	Female	Dialect	Humorous
voices zh-CN-shaanxi-XiaoniNeural	Female	Dialect	Bright
voices zh-HK-HiuGaaiNeural	Female	General	Friendly, Positive
voices zh-HK-HiuMaanNeural	Female	General	Friendly, Positive
voices zh-HK-WanLungNeural	Male	General	Friendly, Positive
voices zh-TW-HsiaoChenNeural	Female	General	Friendly, Positive
voices zh-TW-HsiaoYuNeural	Female	General	Friendly, Positive
voices zh-TW-YunJheNeural	Male	General	Friendly, Positive
voices zu-ZA-ThandoNeural	Female	General	Friendly, Positive
voices zu-ZA-ThembaNeural	Male	General	Friendly, Positive