voxpopuli

Release 0.3.7

Dec 05, 2022

Table of Content

Here's the small yet infinitely useful documentation for voxpopuli. Two paths lay ahead of you:

- If you just want to do text-to-speech (TTS), go over the installation tutorial and jump straight to the *Tutorial* .
- If you want to do a little bit more with voxpopuli (e.g., manipulate phonemes as they're called), you might want to look at *Understanding Espeak and Mbrola*. It's fairly easy to understand, but it will prevent some headscratching and useless google searches.

CHAPTER 1

Installation

1.1 On Linux (Ubuntu)

Install with pip as:

```
pip install voxpopuli
```

You have to have espeak and mbrola installed on your system via apt:

```
sudo apt install mbrola espeak
```

You'll also need some mbrola voices installed, which you can either get on their project page, and then uppack in */usr/share/mbrola/<lang><voiceid>/* or more simply by installing them from the ubuntu repo's. All the voices' packages are of the form *mbrola-<lang><voiceid>*. You can even more simply install all the voices available by running:

```
sudo apt install mbrola-*
```

In case the voices you need aren't all in Ubuntu repository, you can use this convenient little script that install voices diretcly from Mbrola's voice repo

```
# this installs all british english and french voices for instance
sudo python3 -m voxpopuli.voice_install en fr
```

Note: To work, this requires that voxpopuli is installed in your "root" python environment. You'll have to install it with `sudo pip3 install voxpopuli`.

1.2 On Windows and Mac

There are no installation instructions for these platforms currently, for the following reasons:

- On Windows, there used to be an installer distributed by Mbrola's team, but their website went down. Installing Mbrola would require that you compile and install it yourself. Any documentation on how to do that reliably on a recent version of Windows would be appreciated.
- On MacOS, you'd also have to compile mbrola by yourself. Moreover, espeak and mbrola are hardwired to look for voice databases in `/usr/share/mbrola`, which isn't editable on MacOS anymore. `voxpopuli` would require some reworking of its internals to be able to work on MacOS. (without too much troubles on the user side).

CHAPTER 2

An introduction to Voxpopuli

2.1 Tutorial

2.1.1 Picking a voice and making it say things

The most simple usage of this lib is just bare TTS, using a voice and a text. The rendered audio is returned in a .wav bytes object:

```
from voxpopuli import Voice
voice = Voice(lang="fr")
wav = voice.to_audio("salut c'est cool")
```

Evaluating *type(wav)* whould return *bytes*. You can then save the wav using the *wb* file option

```
with open("salut.wav", "wb") as wavfile:
    wavfile.write(wav)
```

If you wish to hear how it sounds right away, you'll have to make sure you installed pyaudio *via* pip, and then do:

```
voice.say("Salut c'est cool")
```

Ou can also, say, use scipy to get the pcm audio as a *ndarray*:

```
import scipy.io.wavfile import read, write
from io import BytesIO

rate, wave_array = read(BytesIO(wav))
reversed = wave_array[::-1] # reversing the sound file
write("tulas.wav", rate, reversed)
```

2.1.2 Getting different voices

You can set some parameters you can set on the voice, such as language or pitch

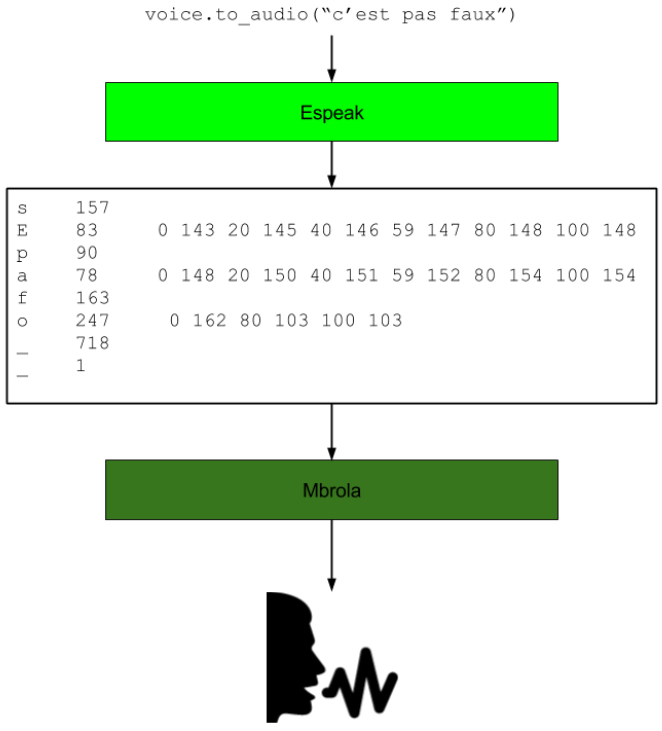
```
from voxpopuli import Voice
# really slow fice with high pitch
voice = Voice(lang="us", pitch=99, speed=40, voice_id=2)
voice.say("I'm high on helium")
```

The exhaustive list of parameters is:

- `lang`, a language code among those available (us, fr, en, es, ...). You can list them using `Voice.list_voice_ids()`.
- `voice_id`, an integer, used to select the voice id for a language. If not specified, the first voice id found for a given language is used.
- `pitch`, an integer between 0 and 99 (included)
- `speed`, an integer, in the words per minute. Default and regular speed is 160 words-per-minute.
- `volume`, float ratio applied to the output sample. Some languages have presets that our best specialists tested. Otherwise, defaults to 1.

2.1.3 Handling the phonemic form

To render a string of text to audio, the Voice object actually chains espeak's output to mbrola, who then renders it to audio. Espeak only renders the text to a list of phonemes (such as the one in the IPA), who then are to be processed by mbrola. For those who like pictures, here is a diagram of what happens when you run *voice.to_audio("Hello world")*



How this works is explained in more details in the section *Understanding Espeak and Mbrola*

Funny thing is, with voxpopuli, you can "intercept" that phoneme list as a simple object, modify it, and then pass it back to the voice to render it to audio. For instance, let's make a simple alteration that'll double the duration for each vowels in an english text.

```
from voxpopuli import Voice, BritishEnglishPhonemes

voice = Voice(lang="en")
# here's how you get the phonemes list
phoneme_list = voice.to_phonemes("Now go away or I will taunt you a second time.")
for phoneme in phoneme_list: # a PhonemeList instance works mostly like a regular␣
↪list object
    if phoneme.name in BritishEnglishPhonemes.VOWELS:
        phoneme.duration *= 3

# rendering and saving the sound, then saying it out loud:
voice.to_audio(phoneme_list, "modified.wav")
voice.say(phoneme_list)
```

Note:

- For French, Spanish, German and Italian, the phoneme codes used by espeak and mbrola are available as class attributes similar to the `BritishEnglishPhonemes` class as above.
- More info on the phonemes can be found here: SAMPA page

2.2 FAQ/Miscellaneous Examples

Here is a miscellaneous list of things that you might want to do with Voxpopuli. If something seems to be obviously missing from this list, don't hesitate and post an issue on the the package's github page.

2.2.1 Listing all avalaible voices

You can list all available voices for all languages on your current installation of voxpopuli this way:

```
>>> from voxpopuli import Voice
>>> print(Voice.list_voice_ids())
{'de': ['1', '3', '5', '8', '4', '7', '2', '6'],
 'en': ['1'],
 'fr': ['7', '2', '1', '3', '4', '6', '5'],
 'it': ['3', '1', '2', '4']}
```

You can also directly get all the available voices instances for a particular language:

```
>>> it_voices = Voices.get_voices_for_lang("it")
>>> it_voices[0].say("buongiorno ragazzi!")
```

2.2.2 Using Voxpopuli as a phonemizer

Although there are packages that are specialized in that regard (such as phonemizer), `voxpopuli` can be used as a simple phonemizer.

```
>>> phoneme_list = Voice(lang="es").to_phonemes("Hola mi amigos, donde esta la␣
↪biblioteca?")
>>> [pho.name for pho in phoneme_list]
```

(continues on next page)

(continued from previous page)

```
['o', 'l', 'a', 'm', 'i', 'a', 'm', 'i', 'g', 'o', 's', '_', '_', 'd',
 'o', 'n', 'd', 'e', 'e', 's', 't', 'a', 'l', 'a', 'b', 'i', 'b', 'l',
 'i', 'o', 't', 'e', 'k', 'a', '_', '_', '_', '_']
```

CHAPTER 3

Understanding Espeak and Mbrola

This page is a short primer on how Espeak and Mbrola work hand in hand to synthesize speech from text.

3.1 Basic Principles

First of all, you should understand that this package's method for synthesizing speech is divided in two main steps:

1. Translating text *in a particular language* into a list of phonemes (Espeak's job)
2. Using that list of phonemes and a voice database to synthesize audio (Mbrola's job)

Unsurprisingly, this is what a diagram describing that process looks like:

3.2 Espeak

Espeak is a pretty old program that used to be one of the only good open source option for basic text-to-speech synthesis (TTS). The synthesis quality is somewhat rudimentary (at least compared to modern deep-learning based solutions, or even Mbrola's speech synthesizer).

However, in order to be able to synthesize speech, espeak needs a phonetic representation of the text that is being fed. This is the part of espeak that is used in Voxpopuli. There are several notations for phonetics, the most commonly used in the linguistic/phonetic world is the International Phonetic Alphabet (IPA).

```
'hello' ->  /hl/
```

However, this notation, since it covers all of the world's languages, isn't very handy when dealing with a single language, nor it is practical to manipulate with a regular ASCII keyboard.

For this reason, a simplified and language-specific phonetic phonetic alphabet was developed: SAMPA. This is that alphabet that is used in our case. Indeed, there is specific set of SAMPA phonemes (all corresponding to an ASCII string of 1 to 3 characters) for each language it supports.

```
'happy' -> "h{pi
```

Moreover, in order to synthesize a string of phonemes, we also need

- its duration in time (usually computed in milliseconds)
- its pitch variations, to also account for the prosody of the synthesized utterance

Espeak also computes these information from the text, using sets of rules that are unique to each languages it supports.

In the end, this is what espeak outputs for the French sentence "c'est pas faux":

```
# This format is usually stored in .pho files
s   107
E   38      0 94 20 94 40 95 59 95 80 96 100 96
p   70
a   40      0 96 20 97 40 97 59 98 80 98 100 98
f   112
o   120     0 102 80 76 100 76
_   350
_   1
```

The first column is the phoneme name in the French SAMPA alphabet, the second one is the duration of each phoneme in ms, and the third (and what follows) are the pitch inflexions. These inflexions are usually only for vowels.

This data is then passed on to Mbrola.

Note: The duration of each phonemes depends on the words-per-minute settings. The pitch of each phonemes depends several factors: if the voice is set to resemble a man or a woman, or a fixed value set as a parameter for espeak.

3.3 Mbrola

Mbrola is also a pretty old program (written in the mid 90's), that only focuses on the audio synthesis part of TTS. From the start, it was made to rely on Espeak's phonemization capabilities to be able to work using only phonemes as an input.

Mbrola uses pre-processed voice databases to render voice. The databases is built using "resynthesized" recordings of diphones (two phonemes pronounced together, such as "ba" or "ish"). Each database contains recording from a single speaker. Thus, ideally, when using a database from synthesize some speech, it's the voice of that unique speaker that should be mimicked in the newly created audio.

Mbrola will then take a .pho file and a voice database as an input, and render some audio. The *pitch* and *duration* of phonemes can be controlled through Mbrola's parameters, as well as the *volume* of the rendered audio.

Note: No new voice database have been created since the beginning of the 2000's. What you'll find in Mbrola's voice repository is all there is.

CHAPTER 4

API reference

4.1 Synthesis Classes

class voxpopuli.**Phoneme**(*name: str*, *duration: int*, *pitch_mods: List[Tuple[int*, *int]] = None*)

Stores the phonetic data for a single phoneme:

- the name of the phoneme in SAMPA notation (depends on the language)
- its duration (in milliseconds)
- its pitch modifications (as a list of *(percentage, pitch)* tuples)

__init__(*name: str*, *duration: int*, *pitch_mods: List[Tuple[int*, *int]] = None*)

Initialize self. See help(type(self)) for accurate signature.

__str__()

Return str(self).

classmethod from_str(*pho_str*)

Instanciates a phoneme from a line of espeak's phoneme output.

set_from_pitches_list(*pitch_list: List[int]*)

Set pitches variations from a list of frequencies. The pitch variation are set to be equidistant from one another.

class voxpopuli.**PhonemeList**(*blocks: Union[voxpopuli.phonemes.Phoneme, Iterable[voxpopuli.phonemes.Phoneme]]*)

A list of phonemes. Can be printed into a .pho string formatted file

__add__(*other: voxpopuli.phonemes.PhonemeList*)

Concatenate two `PhonemeList`

__delitem__(*index: int*)

Remove a phoneme at index i in `PhonemeList`

__getitem__(*index: int*) → voxpopuli.phonemes.Phoneme

Get phoneme in `PhonemeList`

__init__(*blocks: Union[voxpopuli.phonemes.Phoneme, Iterable[voxpopuli.phonemes.Phoneme]]*)
Initialize self. See help(type(self)) for accurate signature.

__iter__() → Iterable[voxpopuli.phonemes.Phoneme]
Iterate over `PhonemeList`

__len__() → int
Number of phonemes in `PhonemeList`

__setitem__(*index: int*, *value: voxpopuli.phonemes.Phoneme*)
Set phoneme in `PhonemeList` at index i

__str__()
Return str(self).

append(*value: voxpopuli.phonemes.Phoneme*)
Append a phoneme to `PhonemeList`

classmethod from_pho_str(*pho_str_list: str*)
Build a `PhonemeList` from a string corresponding to a .pho file typically produced by Espeak.

insert(*index*, *value: voxpopuli.phonemes.Phoneme*)
Insert a phoneme at index i in `PhonemeList`

phonemes_str
Output the `PhonemeList` as a .pho compatible string.

class voxpopuli.**Voice**(*speed: int = 160*, *pitch: int = 50*, *lang: str = 'fr'*, *voice_id: int = None*, *volume: float = None*)

exception InvalidVoiceParameters

__init__(*speed: int = 160*, *pitch: int = 50*, *lang: str = 'fr'*, *voice_id: int = None*, *volume: float = None*)
All parameters are optional, but it's still advised that you pick a language, else it **will** default to French, which is a default to the most beautiful language on earth. Any invalid parameter will raise an *InvalidVoiceParameter* exception.

classmethod get_voices_for_lang(*lang: str*) → List[voxpopuli.main.Voice]
Get instances of all the available voices for a particular language

classmethod list_voice_ids() → Dict[str, List[T]]
Returns a dictionary listing available voice id's for each language

say(*speech: Union[voxpopuli.phonemes.PhonemeList, str]*)
Renders a string or a `PhonemeList` object to audio, then plays it using the PyAudio lib

to_audio(*speech: Union[voxpopuli.phonemes.PhonemeList, str], filename=None*) → bytes
Renders a str or a `PhonemeList` to a wave byte object. If a filename is specified, it saves the audio file to wave as well Throws a *InvalidVoiceParameters* if the voice isn't found

to_phonemes(*text: str*) → voxpopuli.phonemes.PhonemeList
Renders a str to a `` `PhonemeList `` object.

4.2 SAMPA Phoneme Sets

class voxpopuli.**FrenchPhonemes**

CONSONANTS = {'H', 'J', 'N', 'R', 'S', 'Z', 'b', 'd', 'f', 'g', 'j', 'k', 'l', 'm', 'n

```
    FRICATIVES = {'S', 'Z', 'f', 'j', 's', 'v', 'z'}
    INDETERMINATE_WOVELS = {'&/', 'A/', 'E/', 'O/', 'U~/'}
    LIQUIDS = {'H', 'R', 'j', 'l', 'w'}
    NASAL_CONSONANTS = {'J', 'N', 'm', 'n'}
    NASAL_WOVELS = {'9~', 'a~', 'e~', 'o~'}
    ORALS = {'2', '9', '@', 'A', 'E', 'O', 'a', 'e', 'i', 'o', 'u', 'y'}
    PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {'&/', '2', '9', '9~', '@', 'A', 'A/', 'E', 'E/', 'O', 'O/', 'U~/', 'a', 'a~'
    all = {'"', '%', '&/', "'", '2', '9', '9~', ':',  '@', 'A', 'A/', 'E', 'E/', 'H', 'J',
class voxpopuli.BritishEnglishPhonemes

    ADDITIONALS = {'?', 'x'}
    AFFRICATES = {'dZ', 'tS'}
    CENTRAL = {'@'}
    CHECKED = {'I', 'Q', 'U', 'V', 'e', '{'}
    CONSONANTS = {'D', 'N', 'S', 'T', 'Z', 'b', 'd', 'dZ', 'f', 'g', 'h', 'j', 'k', 'l', '
    FREE = {'3:',  '@U', 'A:', 'I@', 'O:', 'OI,', 'U@', 'aI', 'aU,', 'e@', 'eI', 'i:',  'u
    FRICATIVES = {'D', 'S', 'T', 'Z', 'f', 'h', 's', 'v', 'z'}
    GLIDES = {'j', 'w'}
    INDETERMINATE = {'i', 'u'}
    LIQUIDS = {'l', 'r'}
    NASALS = {'N', 'm', 'n'}
    PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    SONORANTS = {'N', 'j', 'l', 'm', 'n', 'r', 'w'}
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {'3:',  '@', '@U', 'A:', 'I', 'I@', 'O:', 'OI,', 'Q', 'U', 'U@', 'V', 'aI', '
    all = {'"', '%', "'", '3:',  ':',  '?',  '@', '@U', 'A:', 'D', 'I', 'I@', 'N', 'O:', '
class voxpopuli.AmericanEnglishPhonemes

    CONSONANTS = {'D', 'N', 'S', 'T', 'Z', 'b', 'd', 'dZ', 'f', 'g', 'h', 'j', 'k', 'l', '
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {'3`', '@', '@`', 'A', 'E', 'I', 'O', 'OI', 'U', 'V', 'aI', 'aU', 'e', 'i', '
    all = {'"', '%', "'", '3`', ':',  '@', '@`', 'A', 'D', 'E', 'I', 'N', 'O', 'OI', 'S',
class voxpopuli.SpanishPhonemes

    ACCENTS = {'"'}
```

```
    AFFRICATES = {'jj', 'tS'}
    CONSONANTS = {'B', 'D', 'G', 'J', 'L', 'T', 'b', 'd', 'f', 'g', 'jj', 'k', 'l', 'm', '
    FRICATIVES = {'B', 'D', 'G', 'T', 'f', 's', 'x'}
    LIQUIDS = {'L', 'l', 'r', 'rr'}
    NASAL = {'J', 'm', 'n'}
    PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {'a', 'e', 'i', 'o', 'u'}
    all = {'"', '%', "'", ':',  'B', 'D', 'G', 'J', 'L', 'T', '`', 'a', 'b', 'd', 'e', 'f'
class voxpopuli.GermanPhonemes

    AFFRICATES = {'dZ', 'pf', 'tS', 'ts'}
    CENTRING_DIPHTONGS = {'2:6', '6', '96', 'E6', 'E:6', 'I6', 'O6', 'U6', 'Y6', 'a6', 'a:
    CHECKED = {'9', 'E', 'I', 'O', 'U', 'Y', 'a'}
    CONSONANTS = {'C', 'N', 'R', 'S', 'Z', 'b', 'd', 'dZ', 'f', 'g', 'h', 'j', 'k', 'l', '
    DIPHTONGS = {'OY', 'aI', 'aU'}
    FRICATIVES = {'C', 'S', 'Z', 'f', 'h', 'j', 's', 'v', 'x', 'z'}
    GLOTTAL_STOP = '?'
    PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    PURE = {'2:',  'E:', 'a:',  'e:',  'i:',  'o:',  'u:',  'y:'}
    SCHWA = {'@'}
    SONORANTS = {'N', 'R', 'l', 'm', 'n'}
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {'2:',  '9', 'E', 'E:', 'I', 'O', 'OY', 'U', 'Y', 'a', 'a:',  'aI', 'aU', 'e:
    all = {'"', '%', "'", '2:',  '2:6', '6', '9', '96', ':',  '@', 'C', 'E', 'E6', 'E:', '
class voxpopuli.ItalianPhonemes

    ACCENTS = {''}
    AFFRICATES = {'dZ', 'ddZ', 'ddz', 'dz', 'tS', 'ts', 'ttS', 'tts'}
    CONSONANTS = {'J', 'JJ', 'L', 'LL', 'S', 'SS', 'b', 'bb', 'd', 'dZ', 'dd', 'ddZ', 'ddz
    FRICATIVES = {'S', 'SS', 'f', 'ff', 's', 'ss', 'v', 'vv', 'z'}
    GEMINATE_AFFRICATES = {'dZ', 'ddZ', 'ddz', 'dz', 'tS', 'ts', 'ttS', 'tts'}
    GEMINATE_FRICATIVES = {'SS', 'ff', 'ss', 'vv'}
    GEMINATE_LIQUIDS = {'LL', 'll', 'rr'}
    GEMINATE_NASAL = {'JJ', 'mm', 'nn'}
    GEMINATE_PLOSIVES = {'bb', 'dd', 'gg', 'kk', 'pp', 'tt'}
    LIQUIDS = {'L', 'LL', 'l', 'll', 'r', 'rr'}
```

```
    NASAL = {'J', 'JJ', 'm', 'mm', 'n', 'nn'}
    PLOSIVES = {'b', 'bb', 'd', 'dd', 'g', 'gg', 'k', 'kk', 'p', 'pp', 't', 'tt'}
    SEMIVOWELS = {'j', 'w'}
    SINGLE_AFFRICATES = {'dZ', 'ddZ', 'ddz', 'dz', 'tS', 'ts', 'ttS', 'tts'}
    SINGLE_FRICATIVES = {'S', 'f', 's', 'v', 'z'}
    SINGLE_LIQUIDS = {'L', 'l', 'r'}
    SINGLE_NASAL = {'J', 'm', 'n'}
    SINGLE_PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    VOWELS = {'E', 'O', 'a', 'e', 'i', 'o', 'u'}
class voxpopuli.PortuguesePhonemes

    CONSONANTS = {'J', 'L', 'R', 'S', 'Z', 'b', 'd', 'f', 'g', 'k', 'l', 'm', 'n', 'p', 'r
    FRICATIVES = {'S', 'Z', 'f', 's', 'v', 'z'}
    LIQUIDS = {'L', 'R', 'l', 'r'}
    NASALS = {'J', 'm', 'n'}
    PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {'6', '6~', '6~j~', '@', 'E', 'O', 'a', 'aj', 'aw', 'e', 'e~', 'i', 'i~', 'o'
    all = {'"', '%', "'", '6', '6~', '6~j~', ':',  '@', 'E', 'J', 'L', 'O', 'R', 'S', 'Z',
class voxpopuli.GreekPhonemes

    AFFRICATES = {'dz', 'ts'}
    CONSONANTS = {'C', 'D', 'G', 'T', 'b', 'c', 'd', 'dz', 'f', 'g', 'gj', 'j', 'jj', 'k',
    FRICATIVES = {'D', 'G', 'T', 'f', 's', 'v', 'x', 'z'}
    LIQUIDS = {'l', 'r'}
    NASALS = {'m', 'n'}
    PALATALS = {'C', 'c', 'gj', 'jj'}
    PLOSIVES = {'b', 'd', 'g', 'k', 'p', 't'}
    SEMIVOWEL = {'j'}
    STRESSES = {'"', '%', "'", ':',  '`'}
    VOWELS = {' a', ' e', ' o', ' u', 'i'}
    all = {' a', ' e', ' o', ' u', '"', '%', "'", ':',  'C', 'D', 'G', 'T', '`', 'b', 'c',
class voxpopuli.ArabicPhonemes

    CONSONANTS = {'?', '?`', 'D', 'D`', 'G', 'S', 'T', 'X\\', 'Z', 'b', 'd', 'd`', 'f', 'g
    FRICATIVES = {'?`', 'D', 'D`', 'G', 'S', 'T', 'X\\', 'Z', 'f', 'h', 's', 's`', 'v', 'x
    LATERAL = {'l', 'l`'}
```

```
NASALS = {'m', 'n'}
PLOSIVES = {'?', 'b', 'd', 'd`', 'g', 'k', 'p', 'q', 't', 't`'}
SEMIVOWELS = {'j', 'w'}
STRESSES = {'"', '%', "'", ':',  '`'}
TRILL = {'r'}
VOWELS = {'a', 'a:',  'i', 'i:',  'u', 'u:'}
all = {'"', '%', "'", ':',  '?',  '?`', 'D', 'D`', 'G', 'S', 'T', 'X\\', 'Z', '`', 'a'
```

Index

Symbols

A

B

C

D

F

G

I

L

N

O

P

S

T

V