README

Janay Monen & Aaricia Herygers

11/4/2021

Real-Voice-Cloning Task

The original repository on which we have based our voice-cloning task can be found here: https://github.com/CorentinJ/Real-Time-Voice-Cloning. In short, the repository implements the phenomenon of '*Transfer Learning from Speaker Verification to Multispeaker Text-to-Speech Synthesis (SV2TTS)*' and contains a synthesizer and vocoder that can produce speech based on only five seconds of speech input.

With our task, we wanted to clone this speech synthesizer and use a couple of our own speech samples to test the outcome in terms of voice and audio quality or in other words, whether the model is able to accurately clone our voice from little speech input.

Data

Models: Based on the original repository, we used the three pretrained models. These are the 'encoder', 'synthesizer' and 'vocoder'. Other than that, we used the relevant sample files, demo-toolkits and requirement.txt documents to set up the task.

Samples: Please note that the original repository uses `.mp3` files as samples for testing the synthesizer, but as we ran into troubles when giving the model such input formats, we resorted to `.wav` files instead.

Expectations

Samples: Since the original repository confirmed that the model is successful and is able to clone a voice after only five seconds of speech input, we expected the model to perform relatively well with the test samples provided in the original data sample set. However, since the models were trained on audio book data, we did not expect the tone to approximate the tone of conversational speech.

Our own data: The original creator states that in case any other dataset than the proposed datasets are used, we might end up being disappointed with the result. Therefore, we expected our own dataset to be of lesser quality as well as less accurate cloning performances. Lastly, we expected speech input with longer durations to result in better cloning results, i.e., the output will sound more like the original.

Results

Based on our predictions of the samples, we can confirm that the models were able to clone the voices up to a certain extent. The overall voice quality was good, but the speech was not very human-like and accents from the original recordings were not cloned (e.g., Scottish accent). Synthesized speech was also returned in a `.wav` format and placed in the original file folder.

Our own data files resulted in speech that was low in quality, fragmented and not very close to the original, confirming our expectations. However, it was able to generate a synthesized output with only a small amount of speech input. Additionally, long speech files (e.g., >1h) did not work with the model, although we could not detect the reason behind it. Moreover, despite the model being compatible with the `.mp3` format, we did not manage to get it to work, so we had to resort to other audio formats.

Note: We attempted to synthesize a Dutch sentence and ended up with an output that produced the Dutch sentence, but with an English-sounding pronunciation.

Setting up the cloning task

Notes: Important to note is the fact that this task cannot be run in any other environments than a local environment, i.e., terminal. This is because we deal with Qt platform plugin "xcb" to develop the Graphical User Interface (GUI), which is not supported in remote environments such as Google Colab. **Therefore, use the terminal to execute the steps below!**

Step 1: Installing/importing relevant packages and downloading the models

We recommend you to downlaod the whole repository, which contains the three pretrained models: encoder, synthesizer and vocoder as well as our added samples and outputs under `Assignment7`. Next, please make sure that you are working in the correct 'working directory' by checking '`pwd`'. If this is not the case, use '`cd`' by giving the correct name of the directory that contains all the files.

Next, before you can set up the task, it is important to download the packages provided below. In case `pip install` does not work, you might need to use `pip3 install`.

- *Python:* Works with Python3.8+, so it is important to have it installed and activated on your computer before continuing (command line: `python3`).
- *Installing PyTorch:* (command line: `pip install torch torchvision torchaudio`)
- *Installing ffmpeg:* (command line: `pip install ffmpeg`)
- *Requirements.txt file*: This file contains the relevant packages that are necessary to run the models, with relevant version numbers if applicable (command line: `pip install -r requirements.txt`). Important to note is that not all packages load correctly, so to ensure that the task is sucessful, install the below-mentioned packages separately in the terminal. You can also refer to the `Cloning_Janay_Aaricia.py` file that has been added for this task.
  - *Librosa:* (command line: `pip install librosa`)
  - *Sounddevice:* (command line: `pip install sounddevice`)
  - *UMAP:* (command line: `pip install umap`)
  - *Unidecode:* (command line: `pip install unidecode`)
  - *Inflect:* (command line: `pip install inflect`)
  - *Matplotlib:* (command line: `pip install matplotlib`)
  - *PyQt5:* (command line: `pip install PyQt5`)

Step 2: Testing your configuration

To test whether your directory with all the models and data has been configured properly, you have to ensure that all files are extracted to the locations within your local copy of the repository such as below, i.e., make sure this structure is followed.

```
encoder\saved_models\pretrained.pt
synthesizer\saved_models\pretrained\pretrained.pt
vocoder\saved_models\pretrained\pretrained.pt
```

You can test this with the '`demo_cli.py`' file that is in the repository/proposed folder.

```
python demo_cli.py --no_mp3_support
#OR
python3 demo_cli.py --no_mp3_support
```

We added the "`--no_mp3_support`" because it resulted in an error without it, something we were not able to fix. If everything has been done accordingly, the last line should be the following: "All test passed! You can now synthesize speech."

Step 3: Testing the voice-cloning tool

Terminal: After the configuration test, you will get an option to select a speech sample from the dataset (your own data or the sample data). For a terminal user interface, you can call on a speech sample from the directory , i.e., `/path/sample_name` and give a +-20-word input (English or you can try another language for fun) to the synthesizer. Once having pressed enter, the models will process the data and provide you with a cloned voice output. To exit this mode, press CTRL+C (on MacOS).

Toolbox: Another option is to use the 'demo_toolbox.py' file, which opens an extra window that allows you to record, upload and play audio files, after which you can synthesize and vocode the given speech data. Additionally, mel spectograms are presented to give you insight into the frequencies of the given sound files (top: original speech, bottom: synthesized speech).

```
python demo_toolbox.py --no_mp3_support
#OR
python3 demo_toolbox.py --no_mp3_support
```

Step 4: Testing out your own dataset(s) - optional

If you wish to test your own datasets, make sure to add them in `.wav` format (or formats other than `.mp3`) to the speech file folder and load them into the models in a similar manner as explained in step 3. Beware that large files may result in python 'killing' the process, which is what occurred when we attempted to load in a file with a >1h duration.