D

Appendix D: Description of Software

This appendix chapter provides a listing of the key elements of software as well instructions for accessing the code and reproducing the experiments. The software is divided into three different repositories: `bachbot` includes tools to preprocess musical data and train/sample/harmonize/score LSTM sequence models, and the other two repositories are the front-end javascript client and back-end `node.js`/Azure server.

D.1 `bachbot`

The `bachbot` repository contains code related to:

1. Preprocessing and sequential encoding for polyphonic music (`bachbot/scripts/datasets.py`)
2. A modified version of `torch-rnn` supporting harmonization and UTF8 token sequences (`bachbot/scripts/harm_model`)
3. Training `torch-rnn` the `bachbot` sequence model (`bachbot/scripts/train.py`)
4. Automatic composition (*i.e.* sampling) with a trained model (`bachbot/scripts/sample.py`)

5. Harmonization with a trained model (`bachbot/scripts/harm_model/harmonize.lua`)
6. Benchmarking against N-gram language models `bachbot/scripts/benchmarks/`
7. Benchmarking against other memory-cell implementations `bachbot/scripts/theanet/`

We provide the following files to demonstrate some use cases:

1. `bachbot/scripts/0-prepare_all.zsh` – prepares preprocessed and encoded corpuses for model training, automatic composition, and harmonization
2. `bachbot/scripts/1-train.zsh` – trains the LSTM sequence model with parameters used for our experiments
3. `bachbot/scripts/2-sample-decode.zsh` – automatic composition; samples a token sequence from the LSTM and decodes into `musicxml`
4. `bachbot/scripts/3-harmonize.zsh` – harmonization; performs all harmonization tasks, decodes and scores the results

D.2 Subjective evaluation

We provide our infrastructure for conducting large-scale web-based human evaluation, which can be easily adapted to other applications. Our code is split into two parts: front-end and back-end.

D.2.1 `subjective-evaluation-client`

The client is written in Javascript (ECMAScript 2016) and requires compilation (`npm run build`) before it can be deployed. Some important files/folders:

1. `src/components` – contains the React code for front-end components (*e.g.* audio playback, quiz question-response form, user-info form, landing page)
2. `src/redux` – contains the Redux code for application state management and data collection
3. `test` – contains tests documenting and enforcing correct application behavior

D.2.2 `subjective-evaluation-server`

The server is written using `node.js` and requires an Azure connection string to be available in the shell environment. The repository is organized as follows:

1. `src/app.js` – static serving of `experiments.json` and handling of POST `/submitResponse` which persists responses to Azure BlobStorage
2. `src/public` – directory for static assets
3. `scripts/` – utilities for interacting with Azure (*e.g.* setting CORS headers, uploading experiments, downloading responses)

D.3 Instructions for access and reproducting experiments

All software has been made open-source and is available on GitHub:

- https://github.com/feynmanliang/bachbot
- https://github.com/feynmanliang/subjective-evaluation-client
- https://github.com/feynmanliang/subjective-evaluation-server

To reproduce experimental results, we recommend cloning the `bachbot` repository and exploring `README.md` and the source code in `scripts/bachbot.py`.

Although we use Azure for the back-end, `subjective-evaluation-client` was written to be vendor-agnostic and can be delivered by any static content web-server or content delivery network. Its dependencies are:

- A URL providing `experiments.json`
- A REST end-point handling POSTs to `/submitResponse` which persists JSON blobs containing user responses

After cloning the repository, run `npm install && npm start`.

A Microsoft Azure account is required for deploying `subjective-evaluation-server`. Its dependencies are:

- Azure App Service – for running the `node.js` web server
- Azure BlobStorage – for persisting JSON blobs of user responses

After cloning the repository, run `npm install && npm start` to start the server locally. Look at `npm run deploy` for more details on deploying to Azure.

Although not required, we encourage the use of a CDN to serve `mp3` files, parameters for the current quiz questions (`experiments.json`), and even the compiled front-end javascript (`bundle.js`). We use Azure CDN for this purpose.

IPython notebooks reproducing the data analyses, figures, and tables shown in this dissertation are available at https://github.com/feynmanliang/bachbot-thesis.