Pattern Recognition Systems Project Report

Intelligent Speech Generation System

Teyvator

Team Members:

Ma Xin	**A0261775W**
Zhang Zhimu	**A0261649W**
Sun Wenyuan	**A0261977M**
Wang Zhiyuan	**A0261827Y**

5-Nov-2022

Content

1. Abstract

The development of the game and anime industry has also stimulated the development of the voiceover industry, a good voice actor's interpretation can make the public love the character more. So the demand for voice dubbing has emerged, which is mainly for secondary creation of videos or voice assistant and map navigation. This is where the AI should be involved.

In the project, we are dedicated to training a text-to-speech (TTS) model which can generate text from speech end to end with multi-speaker mode. The model we mainly use is called VITS[1], proposed in 2021. It is designed to train on English text and speech audio but we managed to design several APIs that make it be trained on a custom Chinese audio text set.

We have also designed a frontend with a backend for the user to operate the text to speech process with GUI, where users can input the custom text and choose the voice from 10 different characters.

Beside the engineering part, we are also doing research and carrying out experiments to modify the VITS model and extend its function, like adding a voice conversion function which takes an audio file as input, and it can generate a speech with the same tone but a converted voice.

2.Project Introduction

2.1 Background Research

With the continuous development of new media technology, the creation of videos is no longer the preserve of professional media people, and more and more non-professionals are involved in video creation. In order to better refine the expression of their videos, creators want to create appropriate audio to assist in the expression of the video content. However, due to the creator's own limitations, the dubbing approach does not fully meet the creator's needs for the sound of the video. In this case, a speech generation system capable of generating audio appropriate to the text message and other needs of the author is the key tool to solve this problem. For example, a speech generation system can capture the voice patterns of a specific celebrity and generate a specific voice based on the text information entered by the creator. An efficient speech generation system

can greatly enhance the creator's design efficiency, and the resulting voices can enhance the appeal of the video.

In addition, speech generation systems are also widely used in the field of live webcasting. For example, a streamer sometimes interacts with audiences via a live bullet screen. However, communication can be hampered if the streamer does not take note of what the audience expresses on the bullet screen. Therefore, a speech generation system that can convert text messages on the bullet screen into speech in real time can facilitate communication between streamers and audiences and increase audience adhesion. As a result, speech generation systems have a high commercial value.

Based on the above background research, our group attempted to build a speech generation system, which is called Teyvator, based on the voices of game characters in the popular game Genshin Impact. This project also fulfilled the requirements of the PRS module.

2.2 Project Aim

Design and generate an intelligent speech generation system, transferring various text datasets to specific speeches and applying the voice patterns of game characters in Genshin Impact to satisfy a series of usage requirements.

2.3 Project Objective

Determine the practice module requirements.

Carry out a literature review.

Capture ideal text datasets from online text libraries from different fields.

Design the front end to interact with users.

Construct a speech generation system algorithm to recognize different texts.

Design the back end to achieve data interaction.

Integrate the front end, system algorithm, and back end.

Test and improve the system.

3. Knowledge Modeling

Knowledge modeling includes the following aspects:

1. Business Value
2. Design
3. Implementation
4. Chosen Tools, Techniques or Framework

The first part is the description of business value. The Design involves Data Acquisition, Data Processing and Knowledge Representation. The implementation indicates the workflow chart. The fourth section describes the various tools and techniques we used to build the system.

3.1 Business Value

The speech generation system has become one of the most important functions in video creation and live webcasting fields. There are also some present applications of speech generation systems like Amazon's Polly [2], Google WaveNet voices [2]. The specific business values of intelligent speech generation system are the following aspects:

1. It is able to develop productivity.
2. It can enhance the viewer/audience adhesion, increasing potential benefits.
3. It allows text-to-speech translation in real-time.
4. It is able to assist video creators to solve problems with speech.

3.2 Design

3.2.1 Data Acquisition

The training of the project's model requires feeding a large number of data, and the acquisition process of data set is as follows:

1. Using an unpacking tool called Extractor, unpack the .pck file in the game directory to get the character audio files of all the characters in the game.
2. Selected the *audio* files (.wav file) of 10 wanted characters. Each *audio* file contains a sentence spoken by a character, for a total of 10 characters,

with about 250 *audio* files per character. In the end, we collected 2,601 pieces of data.

3. Process the selected audio files into a *text* data set.
4. Concatenate the audio file paths, *speaker id* (encode from 0 – 9) and the text to form the file list.

The figure below shows the example of the data set:

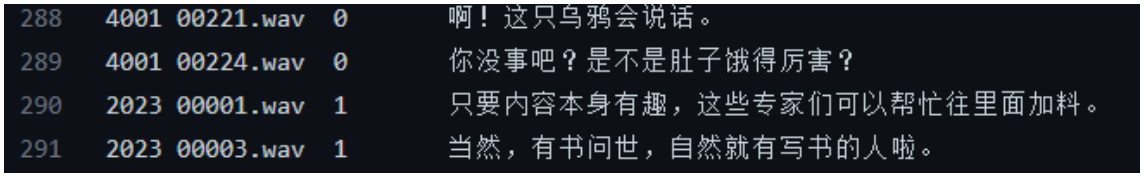


Figure 3.1 Example of data set

3.2.2 Data Processing

In order to better train the model, we need to preprocess the data set. The process is as follows:

1. Divide the file list into a training set and a validation set with a ratio of 1:9.
2. Convert the Chinese characters of text into phonemes.
3. Resample the sample rate of audio files from 48kHz to 22.05kHz (which is believed to have better efficiency).

The Figure below shows how to convert the Chinese characters of text into phonemes:

Figure 3.2 Convert the Chinese characters to phonemes

3.2.3 Knowledge Representation – System Rules

The below table shows formal logic rules of the system:

Table 3.1 System logical rules

LHS	RHS
If do not input text in textarea	Then show the message: Please input some text!
If do not select a speaker in select box	Then show the message: Please select a speaker!
If input some Chinese characters in the textarea and select a speaker in the select box.	Then the system will generate a speech and play it.

3.3 Implementation (Knowledge Model - Process flow and description)

The following flowchart defines the specific processes of using the Intelligent Speech Generation System:

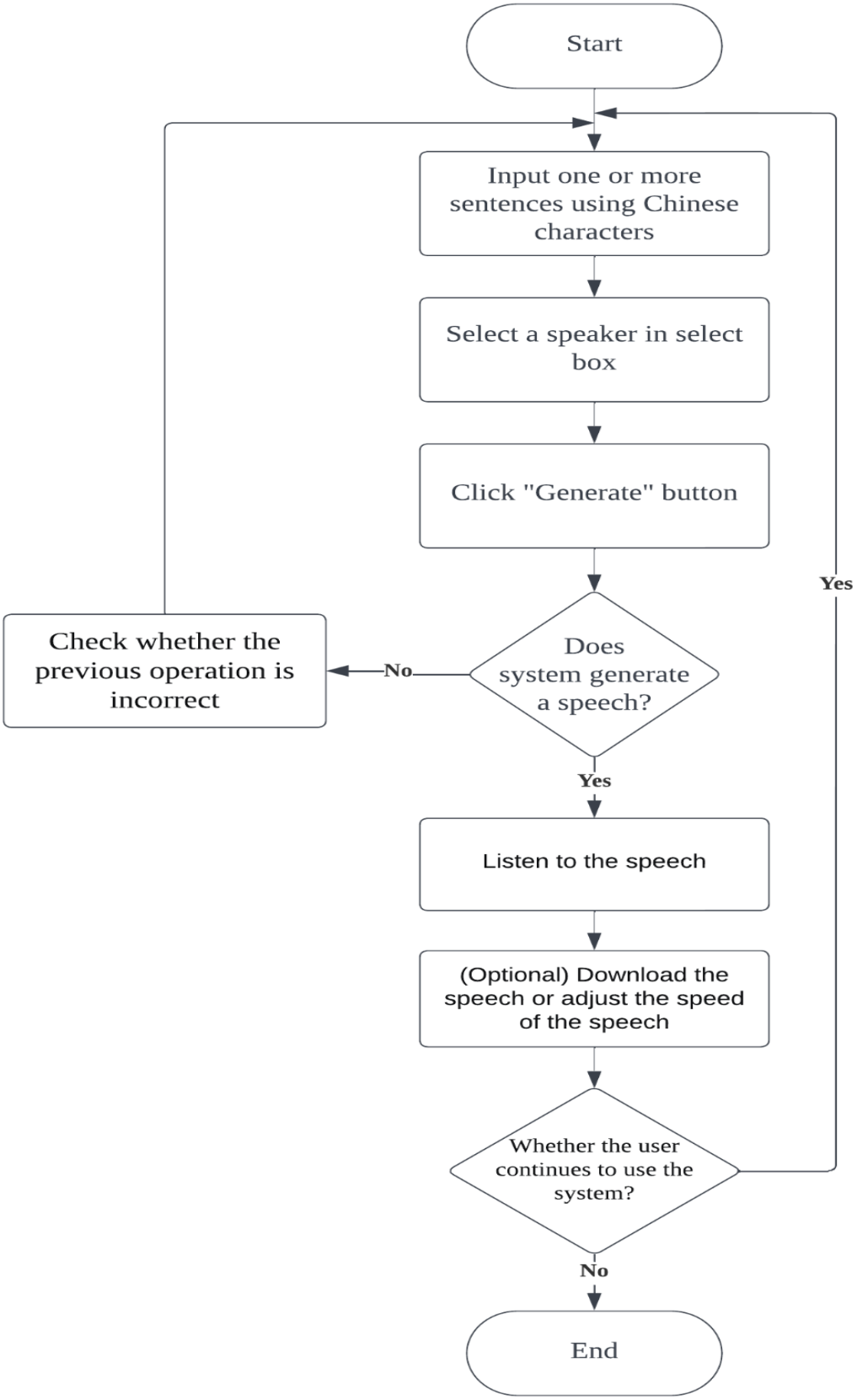


Figure 3.3 Implementation flow chart

3.4 Chosen Tools, Techniques or Framework

Application: Extractor (use to unpack the .pck file), VScode (use to compile the code)

Frontend: Node.js, React.js

Algorithm: PyTorch

Backend: Django

4. Solution Outline

4.1 Algorithm Design

Our TTS model is based on cVAE (Conditional Variational Auto Encoder), flow and GAN (generative adversarial net), which is a hybrid model called VITS proposed in an article on ICML 2021 and it conceptually looks like this.

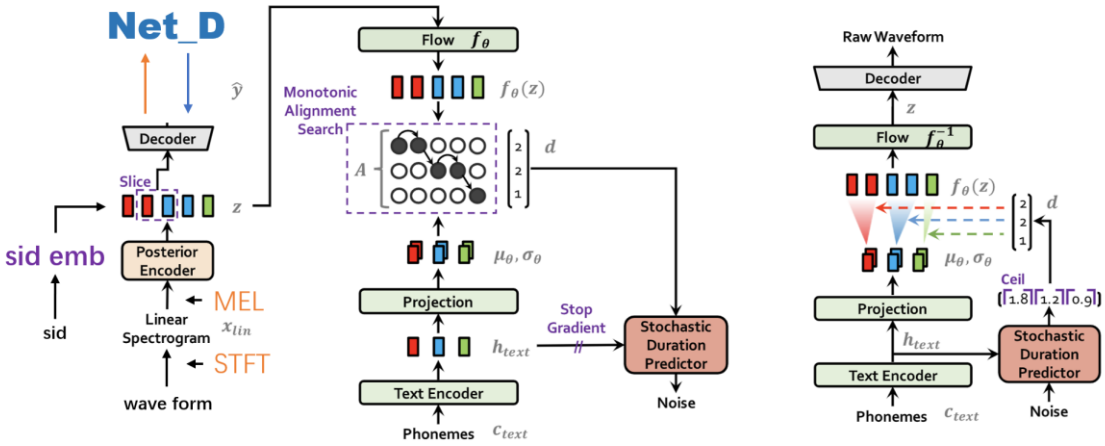


Figure 4.1 conceptual structure of VITS [1]

In the training procedure, we feed in audio file, speaker id and phonemes to the model as input. The waveform audio file after loading will be converted to MEL spectrum by using a short time fourier transform and then goes to the audio encoder, which is a posterior encoder.

The posterior encoder and the decoder will be trained through the input mel spectrum and generate latent variables z, then all of the generated z goes to the

flow, and certain slices of z which depend on the length of the spectrum goes to the decoder.

The adversarial learning happens between the decoder and the discriminator. The discriminator is mainly added to make the generated voice more natural and less robotic.

The speaker id (sid) goes together with the MEL spectrum, to index the slices of a multiple speaker embedding, whose size is the number of speakers by 256. The indexed slice of that embedding will go together with the spectrum through the audio encoder, decoder and flow.

The text encoder is a prior encoder, we feed in the phonemes and get the feature and its distribution. The feature goes to the stochastic duration predictor (SDP) and the distribution will be projected and do the MAS with the flowed latent variable z.

In the monotonic alignment search we mean to let the model learn how each phoneme is pronounced. So here we use dynamic planning to solve out an alignment matrix to maximize the likelihood of each phonemes and spectrum.

Then the matrix goes to the SDP to train it in order to generate a different duration of each phoneme.

As for the inference procedure, we remove the audio encoder and the discriminator, and the flow is reversed. We input the text and it will be processed to be phonemes and go all the way down till we get an output of waveform speech.

4.1.1 Algorithm Details

4.1.1.1 Conditional Variable Auto Encoder (cVAE)

The cVAE is the backbone algorithm of VITS with an objective of maximizing the variational lower bound, also called the evidence lower bound (ELBO), which can be expressed as the following equation.

$$log\ p_\theta\ (x|c)\ \geq\ E_{q_\phi(z|x)}[log\ p_\theta\ (x|z)\ -\ log\frac{q_\phi(z|x)}{p_\theta\ (z|c)}]$$

where $p_\theta\ (z|c)$ denotes a prior distribution of the latent variables z given condition c, $p_\theta\ (z|c)$ is the likelihood function of a data point x, and $q_\phi(z|x)$ is an approximate posterior distribution.

The optimization object is the negative ELBO, which can be viewed as the sum of reconstruction loss $-\ log\ p_\theta\ (x|z)$ and KL divergence $log\ q_\theta\ (z|x) - log\ p_\theta\ (z|c)$, where $z \sim q_\phi(z|x)$.

4.1.1.2 Reconstruction Loss

The reconstruction loss is based on the mel-spectrogram instead of the raw waveform since mel-spectrogram is more consistent with human hearing. The decoder up-samples the latent variables z to the waveform domain $\hat{y}$ through a decoder and transform $\hat{y}$ to the mel-spectrogram domain $\hat{x}_{mel}$. Then the reconstruction loss is formed as follows:

$$L_{recon} = ||x_{mel} - \hat{x}_{mel}||_1$$

4.1.1.3 Monotonic Alignment Search

A monotonic alignment search (MAS) strategy is applied to estimate an alignment A between input text and target speech by searching an alignment that maximizes the likelihood of data parameterized by a normalizing flow f:

$$A = argmax\ log\ p(x|c_{text}, \hat{A})$$

But in our case, as the object is the ELBO rather than the exact log-likelihood, the MAS is therefore refined to find an alignment that maximizes the ELBO, which reduces to finding an alignment that maximizes the log-likelihood of the latent variables z:

$$A = argmax\ log\ p_\theta(x_{mel}|z) - log\ \frac{q_\theta(z|x_{lin})}{p_\theta(z|c_{text}, \hat{A})}$$

4.1.1.4 KL-Divergence

The input condition of the prior encoder c is composed of phonemes c_{text} extracted from text and an alignment A between phonemes and latent variables as illustrated in section 4.1.2.3. In order to provide more high-resolution information for the posterior encoder, the target speech x_{lin} is in the form of a

linear-scale spectrogram rather than the mel-spectrogram. The KL divergence is then:

$$L_{kl} = log\ q_\phi(z|x_{lin}) - log\ p_\phi(z|c_{text}, A)$$

4.1.1.5 Adversarial Training

The model use a discriminator D that distinguishes between the output generated by the decoder G and the ground truth waveform y. Two types of losses are applied in speech synthesis: the least-squares loss function for adversarial learning, and the additional feature-matching loss for generator training:

$$L_{adv}(D) = E_{(y,z)}[(D(y) - 1)^2 + (D(G(z)))^2]$$

$$L_{adv}(G) = E_z[(D(G(z)) - 1)^2]$$

$$L_{fm}(G) = E_{(y,z)}[\sum_{l=1}^{T} \frac{1}{N_l} ||D^l(y) - D^l(y)(G(z))||_1]$$

where T denotes the total number of layers in the discriminator and D^l outputs the feature map of the l-th layer of the discriminator with N_l number of features.

4.1.1.6 Stochastic Duration Prediction (SDP)

To generate a human-like rhythm of speech, a stochastic duration predictor is used. It is a flow-based generative model that is trained via maximum likelihood estimation.

The duration of each input token d_i is calculated by summing all the columns in each row of the estimated alignment $\Sigma_j\ A_{i,j}$.

Since each input phoneme duration is a discrete integer, which needs to be dequantized so as to use the continuous normalizing flows and scalar, a variational dequantization and a variational data augmentation is applied. To be specific, two random variables u and v are introduced, which have the same time resolution and dimension as that of the durations as that of the duration sequence d, for variational dequantization and variational data augmentation respectively. The support of u is restricted to be [0, 1) so that the difference $d - u$ becomes a sequence of positive real numbers, and by concatenating the v and d channel-wise a higher dimensional latent representation is obtained. Then the two

variables are sampled through an approximate posterior distribution $q_\phi(u, v|d, c_{text})$. Finally, we can result in an objective which is a variational lower bound of the log-likelihood of the phoneme duration:

$$log\ p_\theta(d|c_{text}) \geq E_{q_\phi(u,v|d,c_{text})}[log \frac{p_\theta(d-u, v|c_{text})}{q_\phi(u, v|d, c_{text})}]$$

The training loss L_{dur} is then the negative variational lower bound

4.1.1.7 Final Loss

With the combination of VAE and GAN training, the total loss can be expressed as follows:

$$L_{vae} = L_{recon} + L_{kl} + L_{dur} + L_{adv}(G) + L_{fm}(G)$$

4.1.2 Experiment Result

The VITS model is trained on colab for about 2000 epochs after 4 days of training, and the mel-loss jitters between 16 and 20, which can be regarded as converging.

The model performs equally well on most of the training set and validation set, and can generate fluent speech with multiple voices. Different characters can speak the same sentence differently in their own speaking style and tone, they could even speak the same modal word in their own habits.

Besides the pronunciation of the words, it also learns from the punctuation. For example, a period will result in longer pause and firmer ending that a comma, and a question mark will lead to a slightly upward tone.

But when it comes to some customized sentences with words that never appear in the training set, the generated speech would become unnatural, especially on the pronunciation of those unfamiliar words.

The model also learns the rhythm of different characters, which means for the same sentence, different characters would say it in their own speaking style. This is both good and bad. For example, if one character only has speeches in a cheerful tone in its training set, it may not read a sad sentence well.

Anyway, the performance of the trained model has met the expectations, but with more data sets and time and better computational power, it can still be improved.

4.1.3 Algorithm Refinement

Since the model doesn't have the ability to understand the emotion of the words spoken, the best way to optimize the intonation generation is to input an audio form of speech as reference, and we managed to make a demo of such approach by reusing the trained posterior encoder, which is the audio encoder during the training procedure. Currently it can only convert a trained voice to another trained voice while the tone stays the same. In order to generate custom speech with custom tone from that, we can either let the model learn from the voice of the user or just choose a trained voice that appears most similar to the user's voice, by this it may be able to generate high quality speech with a converted voice and the same tone and proper emotion.

4.2 System Design

The whole project is mainly built by Python, applying React library in frontend, Django in backend, and PyTorch for VITS algorithm. The following figure shows the system architecture.

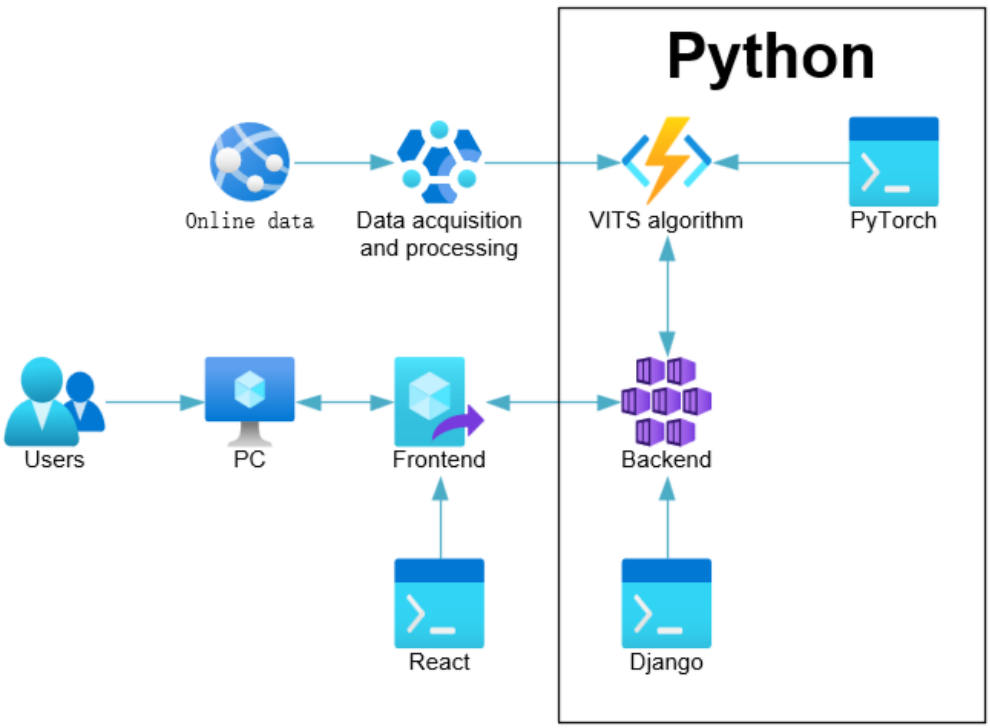


Figure 4.2 System framework

4.2.1 Frontend Construction

We use React to build our front-end interface. React is a JavaScript library for building user interfaces and the reason why we choose React.js is because React has two obvious advantages.

Declarative: React effectively updates and renders the correct components when the data changes. Declarative views make code more predictable and understandable.

Component-Based: Applications are built using components, each of which has its logic and controls. We can reuse these components across different pages, increasing efficiency.

The below figure shows the whole pipeline of one use case.

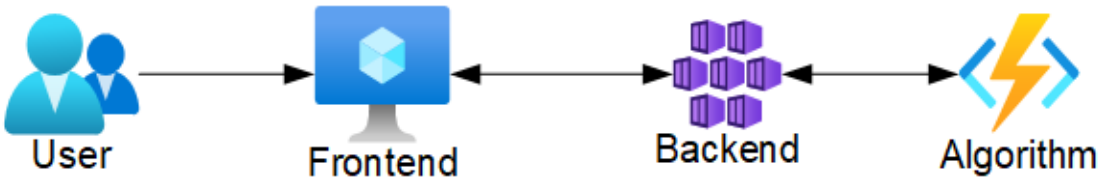


Figure 4.3 Pipeline of use case

In the frontend, users need to input some words or sentences using Chinese characters in text area and then select a speaker in the Select box. After that, the user can click the “generate” button and then listen to the speech. If the user does not input some text or does not select a speaker, the system will show the message to let the user follow the rule.

4.2.2 Backend Construction

The backend service is built with the Django framework. First, create a django project. Second, create an app, which is used to process the user input texts and the text conversion interface. Third, by creating an API, it could get user input text and a speaker name from the frontend, then convert the text to speech by the algorithm module.

Table 4.1 Request API

API URL	DESCRIPTION
/index/	Get the text and speaker name from the frontend and convert the text to speech by the algorithm. The voice of the speech is according to the speaker's name

The backend service has following functions:

Ø Receiving a request and package the data into work component

Ø Integration with speech convert services (Connect to the algorithm model)

Ø Provide API for front-end services

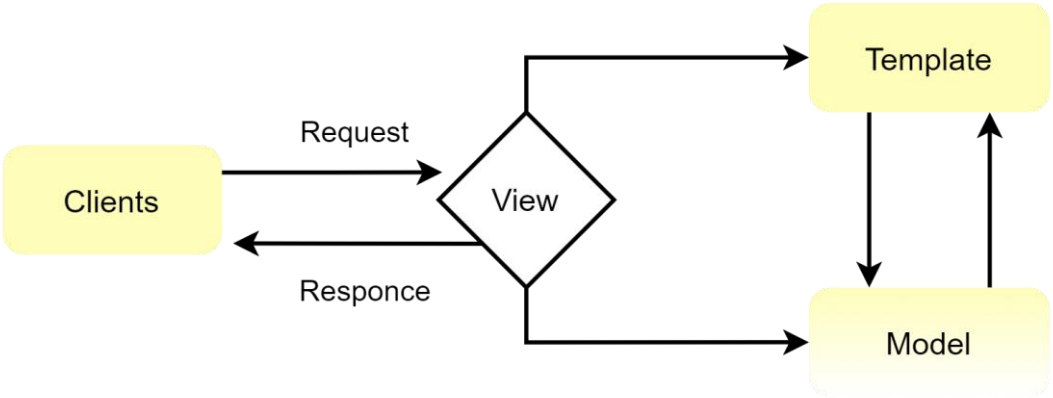


Figure 4.4 MVC model

4.3 System Features

This Intelligent Speech Generation System is web-based and you could simply use it by using any Internet-connected device that enters the URL. Its anchor voice style is richer which contains ten timbres in total. According to which different voices are classified in detail, including male anchor, female anchor, child anchor. You can input the text with any text you want. Then, the system will automatically convert the text which user input to the speech which user chose in Chinese.

4.4 System Performance

After finishing construction of the frontend, backend and algorithm model, we try to define a series of commands to test the capability of our Intelligent Speech Generation System.

Table 4.2 Testing of system

Test commands	Test results
Input text choose speaker name	Output the speaker speech
download conversion speech	The speech will output in wav file
Adjust the speed of speech	The speed can be faster or slower

5. Conclusion and Discussion

5.1 Conclusion

In this project, students put into practice what they have learned in class. In the process of practice, we also met many challenges in this process. However, we ended up using Django and React to build a complete Intelligent Speech Generation System on the front end and back end. The intelligent speech generation system can successfully transfer text to desired speech.

5.2 Discussion- Limitations

In general, the diversity and novelty of conversion are very important to evaluate the effectiveness of emotion expression. However, an increase in input text may result in an increase in output errors. There is a lack of in-depth analysis of the accuracy and diversity of input conversion.

In this system, the variety of converted language can be improved, insert more types of emotion expressions. The conversion of speech can be more accurate using after adding different emotion expressions.

5.3 Discussion- Future Development

1) Algorithm

In order to improve the accuracy rate, a variety of algorithms should be used to compare, and select a more appropriate algorithm. Even more, by combining various algorithms to achieve the best effect.

2) Support more languages

The game Genshin Impact is a Chinese game. Hence, the game has the most Chinese voice material of any character, which is able to supply sufficient datasets to train the algorithm. In this case, our intelligent speech generation system now only supports Chinese. In the future, we will expand the language library of the system to support more languages.

3) Mobile application

Our intelligent speech generation system now only runs on PCs. In the future, we will try to transfer our system to mobile terminals like smart phones, improving ease of use.

6. Reference

[1] Kim J, Kong J, Son J. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech[C]//International Conference on Machine Learning. PMLR, 2021: 5530-5540.

Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech (mlr.press)

[2] Dale, R. 2022. *The Voice Synthesis Business: 2022 Update.* [Online]. [Accessed 02 November 2022]. Available from:

https://becominghuman.ai/the-voice-synthesis-business-2022-update-68401b4b0f57

Appendix A: Mapped System Functionalities against knowledge, techniques and skills of modular courses:

Modular Coursers	System Functionalities / Technique Applied
Problem Solving Using Pattern Recognition	**Deeplearning:** The whole model is based on neural networks **Deeplearning: Create:** We applied generative models and adversarial methods in our project
Pattern Recognition and Machine Learning Systems	**Machine Translation:** The model converts words to phonemes and to embeddings. **Neural Network Models and Designs:** We use neural network models as backbone methods.
Intelligent Sensing and Sense making	**Signal Representation Using Machine Learning:** Our model uses STFT to process the input wav file. **Statistical Signal Analysis:** Our model uses distribution to represent the features of spectrums.

Appendix B: Installation & User Guide

Installation

For the front-end, we need to run the follow command:

Install node.js from the website https://nodejs.org/en/

npm install webpack webpack-cli

npm i -g create-react-app

npm install react-router-dom

npm install axios

npm install --save react-audio-player

npm install antd --save

yarn add node-sass@npm:dart-sass

For the back-end, we need to run the follow command:

pip install django-cors-headers

pip install djangorestframework

pip install Cython

pip install librosa

pip install matplotlib

pip install numpy

pip install phonemizer

pip install scipy

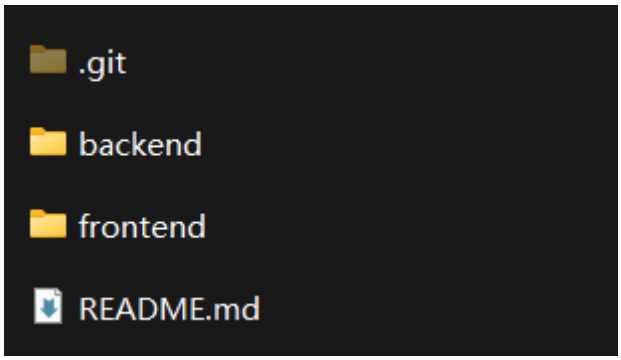
pip install Unidecode

pip install pypinyin

pip install pypinyin_dict

pip install jieba

User guide



Use vscode or other compilers, open the backend folder, then run command: **python manage.py runserver** to run back-end server.

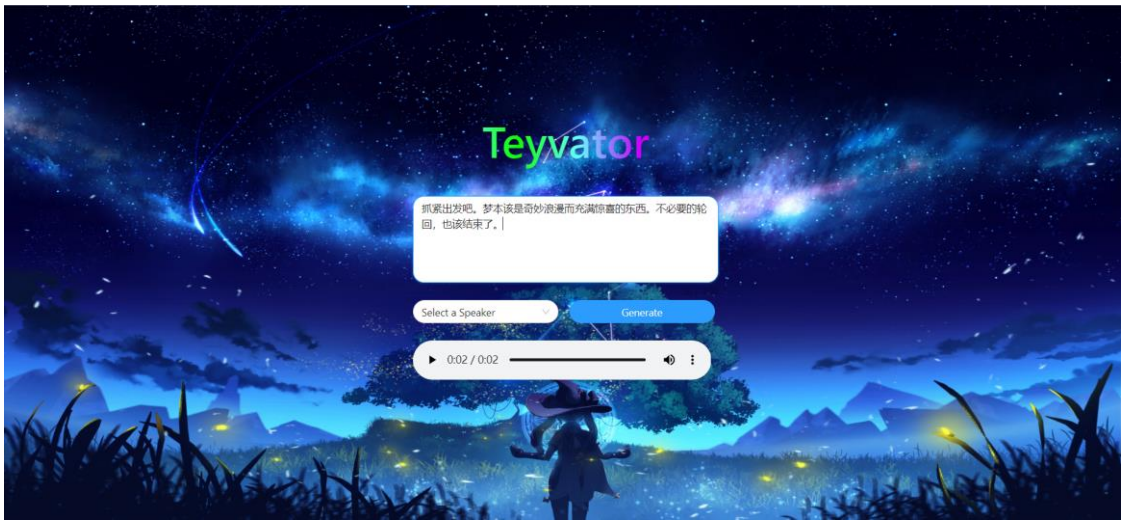
```
PS D:\Github_speed_1_0_8_0 (2)\5002Project\backend> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 05, 2022 - 16:35:06
Django version 4.1.2, using settings 'backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Use vscode or other compilers, open the frontend folder, then run command: **npm start** to run front-end server.

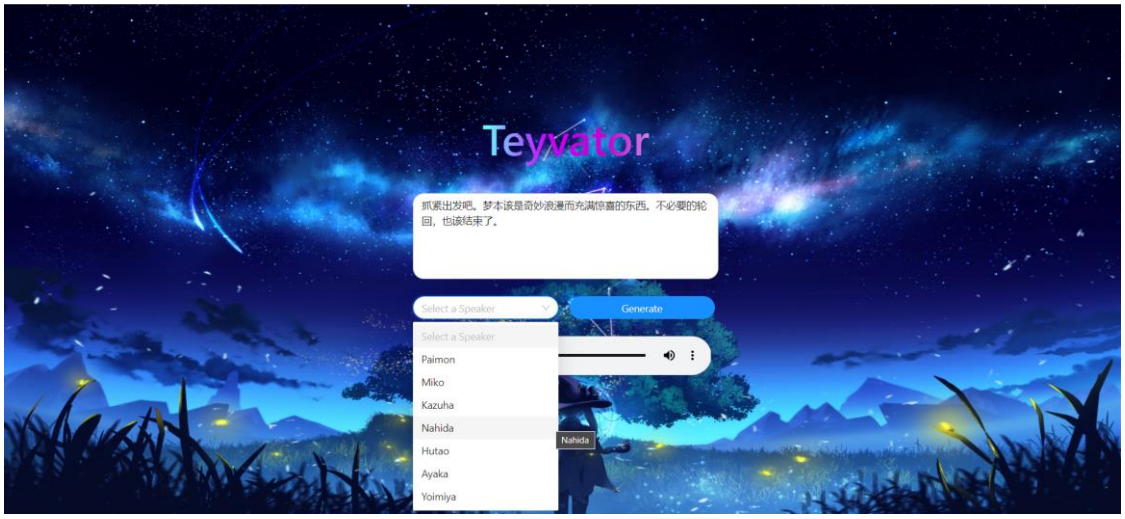
```
You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use yarn build.

webpack compiled successfully
```

Enter the URL: http://localhost:3000

1. Input some text using Chinese characters.



2. Select a speaker in the Select box



3. Click “Generate” button