2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech 2020)

End-to-end Conversion Speed Analysis of an FPT.AI-based Text-to-Speech Application

Tran Duc Chung[1], Micheal Drieberg[2], Mohd Fadzil Bin Hassan[3], Alexandra Khalyasmaa[4]
[1] *Computing Fundamental Department, FPT Technology Research Institute*
[2] *Department of Electrical and Electronics Engineering*
[3] *Centre for Research and Data Science (CeRDaS), Computer and Information Science Department*
[4] *Automated Electrical Systems Department*
[1] FPT University, Hoa Lac Hi-Tech Park, Hanoi, Vietnam, 155300
[2,3] Universiti Teknologi PETRONAS, Seri Iskandar 32610, Malaysia
[4] Ural Federal University, Yekaterinburg, Russia, 620075
chungtd6@fe.edu.vn, {mdrieberg, mfadzil_hassan}@utp.edu.my, lkhalyasmaa@mail.ru

2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech) 978-1-7281-7063-3/20/$31.00 ©2020 IEEE 10.1109/LifeTech48969.2020.1570620448

***Abstract*—In this paper, an FPT.AI-based text-to-speech (TTS) application is developed that converts Vietnamese text into spoken words. The application is developed based on Django for Python and in the form of an interactive web page which is connected to an FPT.AI server through its application programming interface (API). The application supports conversion of text to seven different Vietnamese speeches. Four out of seven voices can be used to convert up to 500 characters in a single transaction while the others support that of 400 characters. Based on the results obtained, the first conversion time takes up to 10 s to convert 400-character text into speech while the subsequent times, given same text, it takes under 1.8 s for the conversion. This is applicable to all voices.**

Keywords—FPT.AI, TTS, performance, analysis, Vietnamese, voice

I. INTRODUCTION

In recent years, the development of automated responding systems (i.e., chatbot, voicebot [1]–[4]) has enabled the extensive use of speech-to-text (STT) and text-to-speech (TTS) systems. The latter enables a system to speak out a given input text with a human voice [5], [6]. This provides a close-to-nature response to user who interacts with the system. In addition, user experiences with the system can be significantly improved since the system can provide promptly responses to any requests. However, the quality of the responses depends on multiple factors such as input signal quality, understanding of request and context, and conversion engine [4]–[8]. The most important one would be the engine.

There are several parameters that can be used for indicating the performance of a TTS system. The most common parameter is mean opinion score (MOS) which is broadly used to measure the naturalness of the generated speech [8]–[10]. However, this is not enough to indicate the performance of a system from customer perspective. The reason is that end-users only experience end-to-end TTS conversion, therefore, even if the core engine is very fast, the intermediate communication media may affect how fast the system can perform the conversion. Thus, in this research, end-to-end conversion speed of an FPT.AI-based TTS application is analyzed with the main focus on the relationship between the length of the input text and its end-to-end conversion time.

The main contributions of this research are: (i) an FPT.AI-based TTS application using Django for Python, (ii) performance analysis of the application for seven different supported voices and several lengths of input text. The paper is organized as follows: Section II presents the research methodology; Section III discusses experimental results and analysis; and finally Section IV concludes this research.

II. METHODOLOGY

A. About FPT.AI API

In this work, FPT.AI API [11] is used for interfacing between local host and remote FPT TTS server. This represents an actual working condition when an external user needs to use the API for converting text to speech. The TTS API takes four arguments for forming an HyperText Transfer Protocol (HTTP) request before posting it to the server: (i) *POST Data* which contain text to be converted to speech; (ii-iv) *voice*, *speed* and *prosody* to form HTTP header. The latter three arguments are not required by the TTS server during a request. The reason is that by default, normal speed (*speed* = 0) and *female* voice (see Table I) (Thu Dung - Northern Female) are used. The *prosody* is only available for question working with *male* voice. Hence, in this work, it is not considered for performance analysis.

Table I presents the available voices supported by FPT.AI engine. These voices are provided for trial users to convert up to 10,000 characters to speech monthly [11].

TABLE I. FPT.AI AVAILABLE VOICES

No.	Voice	API Parameter
1	Ban Mai (Northern Female)	banmai
2	Le Minh (Northern Male)	leminh
3	Lan Nhi (Southern Female)	lannhi
4	Thu Dung (Northern Female)	female
5	Cao Chung (Northern Male)	male
6	Ha Tieu Mai (Southern Female)	hatieumai
7	Ngoc Lam (Hue's - Central Female)	ngoclam

For each request, a response will be returned to host application by the server. It has JavaScript Object Notation (JSON) format and contains a static HTTP link to download the converted audio file in *.mp3 format. In addition, the response has an error-or-success indicator plus a message detailing the system's feedback. Since, it may take some time for the server to generate the audio file, this work aims to assess the end-to-end conversion timing of the system.

B. Required Tools

In order to complete this work, the following tools are used:

- Integrated Development Environment (IDE): Visual Studio Code version 1.39.2
- Programming Languages: hypertext markup language (HTML), cascading style sheets (CSS), Python.
- Framework: Bootstrap, Django for Python [12].

- FPT.AI API [11].

Here, the IDE is used for providing development environment. Three programming languages are used in conjunction to form the application. In particular: HTML is used for developing web-based interface; CSS is used for styling the interface; Python is core programming language which performs back-end operations. The Bootstrap framework helps to build the responsive interface while Django supports quick development of the web application. In addition, the FPT.AI engine performs conversion from text to speech and returns a converted audio file in *.mp3 format.

C. Application Structure

In this research, the application structure is presented in Fig. 1. Here, it is seen that there are three main input parameters that user can key into the system: *text* to convert to speech, the desired *speed* of generated speech and *voice* type. These inputs are fed into the web application developed based on Django for Python. This application is used at local host.

There are four main files which are at most important to the application. The *urls.py* file located in *mysite* folder is used to initialize a localhost which address is *http://127.0.0.1:8000/*. When user runs the application, it will either return an address to *admin* folder containing administration information or synchronize with file *urls.py* located in *polls* folder to connect server and client by their addresses.

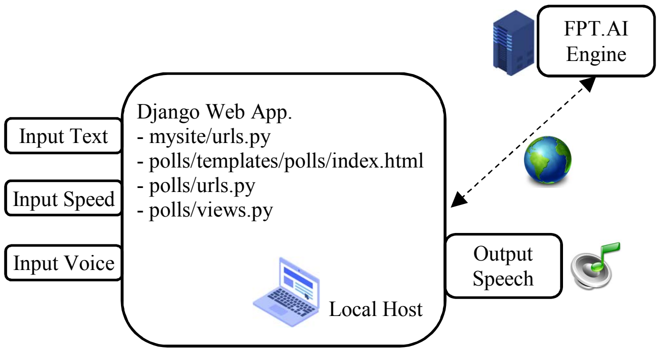

Fig. 1. FPT.AI-based TTS application structure.

The *urls.py* file in the *polls* folder imports path of local server and connects between *views.py* and *index.html* files by paths. Here, the *views.py* file contains the *getVoice* function which performs all back-end tasks to complete the TTS conversion. On the other hand, the *index.html* file layouts the web application and provides an interface between local host and end-user.

The executable content of the *views.py* file is presented in the Fig. 2. In Fig. 2, *views.py* is the core controller. It defines a module containing all necessary functions for completing a TTS task. In this file, the function *index(request)* will perform rendering of the web interface given at *index.html* template to obtain user's information such as desired text, speech speed, and voice for conversion. These data are stored into *input_text*, *speech_voice*, *speech_speed* variables respectively. A link to FPT.AI server is created and stored into *url*. The human-machine interface address that the file will send the request to FPI.AI engine is *https://api.fpt.ai/hmi/tts/v5*. Next, an HTTP header is created based on API key generated for the dedicated user, user's desired speech speed and voice.

```
from django.shortcuts import render
from django.http import HttpResponse
from django.http import HttpRequest
import requests

# Create your views here.
def index(request):
    return render(request, "polls/index.html")

def getVoice(request):
    input_text = request.POST["voice"]
    speech_voice = request.POST["name"]
    speech_speed = request.POST["speeds"]
    url = 'https://api.fpt.ai/hmi/tts/v5'
    header = {
        'api-key':
'BJtsEZz3CCpeRCixZL6EvVpjPmhCewHQ',
        'speed': speech_speed,
        'voice': speech_voice
    }
    response = requests.request('POST', url,
data=input_text.encode('utf-8'), headers=header)
    substr = response.text.split('"')[3]
#success
    context = {"link": substr}
return render(request, "polls/index.html",
context)
```

Fig. 2. views.py content.

Each registered user is allowed to use the TTS service for multiple requests amounting a total of 10,000 characters monthly [11]. This is sufficient for a new user to experience this service from FPT. When a request is triggered, its response is stored in *response* variable. The application then checks if the response returns a *success* state, it will create an event to update the application to playback the returned speech.

D. Measurement Approach

With the design described in the above section, user can experience FPT.AI-based TTS services after registration to the system. However, in order to analyze the end-to-end conversion timing, the *getVoice* function in the *views.py* file is edited as in the following algorithm (see Fig. 3).

In Fig. 3, when the application starts, it captures *startTime* and receives from user text *Data*, desired speech *Speed*, and *Voice*. The application then prepares header information to send TTS conversion request to the server. Before this action is performed, *timeout* variable is set to 0.

When application receives response from FPT.AI server, it checks if character quota has exceeded, in this case, *link context* is set to *null* because no more conversion is permitted. In the other case, if the response is succeeded, audio link string (*urlstr*) is extracted from the response. Next, the application checks if the link is available on the server by sending a get request to it. If the link is not ready yet, it continues to check until 10 s has passed, by then the response is considered as timeout. Here, it should be noted that the 10-s period is typically more than sufficient time for a user to wait for a response from a machine at which it does not cause negative experiences to the user. On the other hand, when the link is available, it is passed to the *link context*. This is used to update the information retrieved from server onto the web application (*index.html*). At this stage, the application obtains *endTime*. It writes log information for further investigation if needed. When timeout occurs, instead of logging processing time (the difference between *endTime* and *startTime*), a timeout text is recorded. Finally, the application triggers a rendering event to update the web interface and playback the retrieved speech.

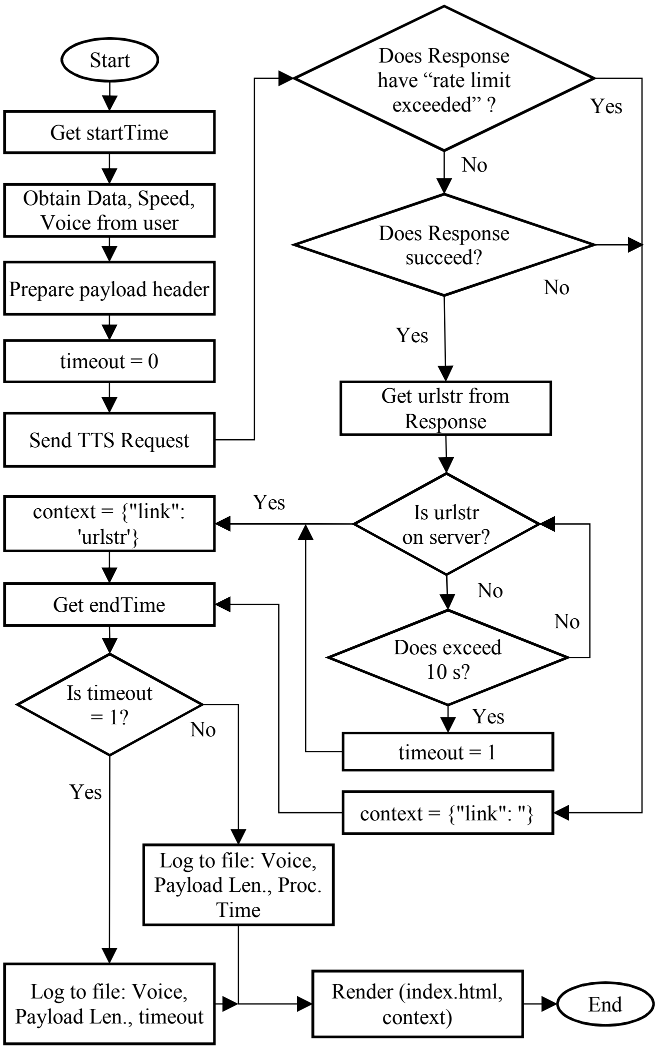

Fig. 3. getVoice's modified algorithm.

In order to assess the end-to-end conversion speed of the TTS engine, the experiment is carried out for all seven supported voices on an Intel Core i5-5250U system with 4 GB DDR3 RAM. The selected lengths of input text are: 10, 100, 200, 300, 400, and 500 characters. The least length typically represents a couple of spoken words while the most length is suitable for a short spoken paragraphs. For representing randomness nature of TTS system, a random text for each length is obtained from a local news web page.

The measurement was performed for 10 times, with the same text for each length. The first conversion time represents randomness nature of TTS application while the second to the tenth times represent the nature of TTS application in which multiple users would like to convert the same text from a news page to speech for listening purpose. Furthermore, it is found that the TTS system will perform the conversion for a given text if it is new to the system and has not been recorded before. This is to save the conversion timing and enhance user experience.

III. RESULTS & ANALYSIS

As a result of the development, the application's web interface is presented in Fig. 4.

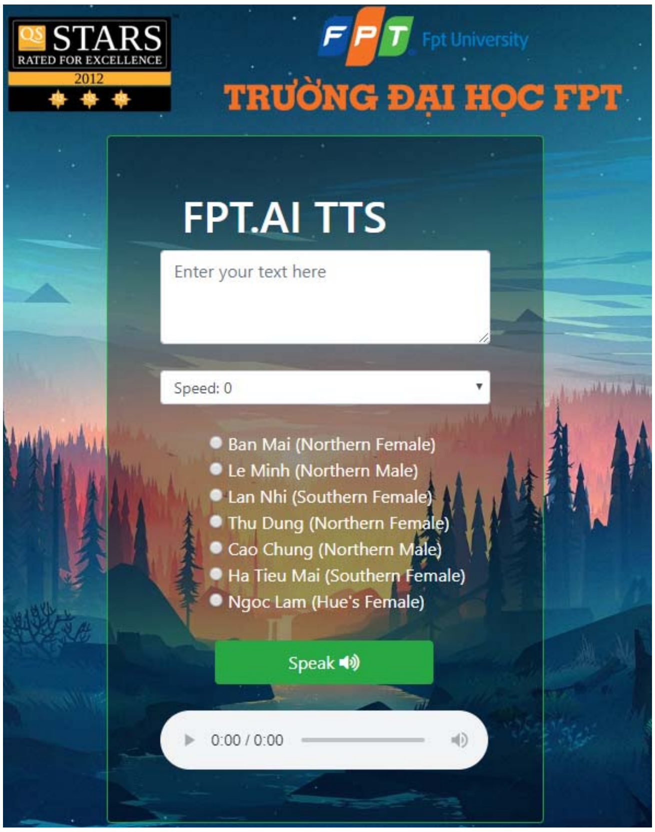

Fig. 4. Application's web interface.

In Fig. 4, it is seen that there are three main regions: (i) text area for writing a random text; (ii) configuration area for selecting speed of speech and a supported voice out of seven available voices; and (iii) user action area for triggering TTS conversion and playing-back the returned speech. The back-end of this application is processed by Python programming language and it is connected to FPT.AI core engine which allows up to 10,000-free-character-conversion for TTS application. There are seven available speeds of output speech ranging from -3 to +3 with step size of 1. When *speed* is 0, the speech is in normal condition, not fast nor slow. When *speed* is -3 it is the slowest, and when *speed* is +3 it is the fastest. *speed* equaled to 0 is also the default setting of the application.

Fig. 5 presents a graph of the end-to-end processing time of the first TTS conversion as a function of input text length. In most cases, the first processing time plays an important role in generating a newly stored speech of a given text. In the subsequent requests of the same input text, the processing time is typically much smaller than that of the first time. This can be observed by comparing figures 5 and 6. This is because that given the same input text, the FPT.AI engine will generate the same output speech and store it into a static HTTP link. The generated file has *.mp3 format.

Therefore, the end-to-end conversion time in the subsequent requests are typically the time to retrieve the generated audio file from the server which does not include file generation time. In Fig. 5, it is seen that as the text length increases, the time taken to perform TTS task increases as well. Out of 7 voices, 3 voices namely Ban Mai, Le Minh, Lan

Nhi could not generate speech from 500-character input text. For the particular case of Lan Nhi, during the first request, at 400-character input, the system could not generate speech. The three voices also take more time to generate speech compared to the others. It takes about 10 s and 9 s correspondingly to generate a speech from 400 characters. For Lan Nhi, at most, it takes about 8.8 s to generate speech from 300 characters. Only Thu Dung voice has conversion speed proportional to the number of input characters.

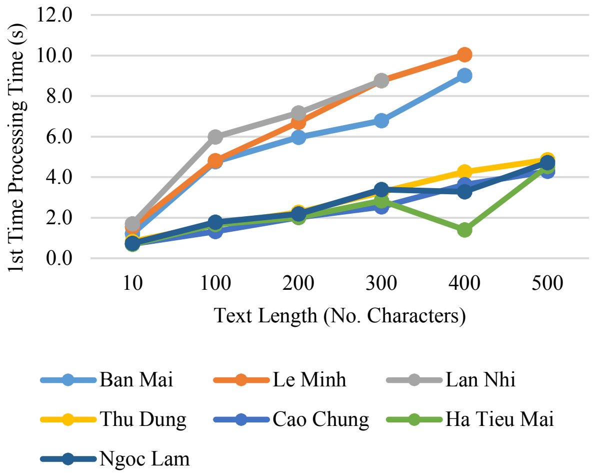

Fig. 5. The first end-to-end conversion time vs text length.

In order to further analyze the conversion speed of the developed engine, the average end-to-end conversion time vs text length is presented in Fig. 6.

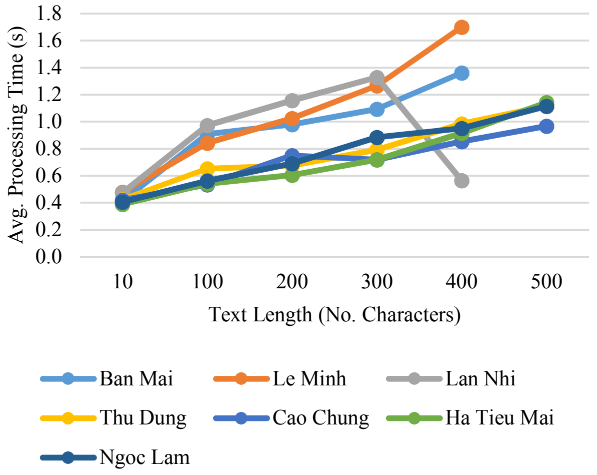

Fig. 6. Average end-to-end conversion time vs text length.

Here the patterns for each voice are similar to those seen in Fig. 5. The difference is that on average, it takes less than 2 s to generate a speech from a given text with length up to 500 characters. For the case of 400-characters, Lan Nhi's average end-to-end conversion time is 0.565 s. This indicates that although the first conversion could not be obtained (see Fig. 5, text length = 400 characters), the subsequent requests are still addressed properly. This information shall be useful for application developer to take into consideration in designing their TTS engine.

IV. CONCLUSIONS

This work has presented a development framework for processing text to generate different male and female voices for Vietnamese. Based on the results obtained, it is seen that as input text length increases, its end-to-end conversion time to obtained the converted speech also increases accordingly. It is important to note that TTS application can help people in various working and entertainment environment, especially with those having difficulties in communication. In future, we will further improve the developed application to provide more attractive graphical user interface, and improve its voice's quality.

ACKNOWLEDGMENT

The authors would like to thank FPT University (FPTU), Universiti Teknologi PETRONAS, and Ural Federal University for supporting this research. Furthermore, the authors would like to express sincere thanks to Nguyen Quang Huy, Nguyen Quoc Hieu, Bui Huu Minh Tri, and Nguyen Ba Tien Anh - the students of class SE1319 at FPTU for their programming supports in this work.

REFERENCES

[1] B. Liu *et al.*, "Content-Oriented User Modeling for Personalized Response Ranking in Chatbots," *IEEE/ACM Trans. Audio Speech Lang. Process.*, 2018, doi: 10.1109/TASLP.2017.2763243.
[2] H. Cuayáhuitl *et al.*, "Ensemble-based deep reinforcement learning for chatbots," *Neurocomputing*, 2019, doi: 10.1016/j.neucom.2019.08.007.
[3] S. Arsovski, H. Osipyan, M. I. Oladele, and A. D. Cheok, "Automatic knowledge extraction of any Chatbot from conversation," *Expert Syst. Appl.*, 2019, doi: 10.1016/j.eswa.2019.07.014.
[4] D. C. Tran, H. S. Ha, and A. Khalyasmaa, "A Question Detection Algorithm for Text Analysis," in *2020 5th International Conference on Intelligent Information Technology (ICIIT)*, 2020, pp. 1–6.
[5] F. Eyben *et al.*, "Unsupervised clustering of emotion and voice styles for expressive TTS," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2012, doi: 10.1109/ICASSP.2012.6288797.
[6] W. Ping *et al.*, "Deep Voice 3: 2000-Speaker Neural Text-to-Speech," in *Proc. ICLR*, 2018.
[7] D. C. Tran and A. K. M. K. A., "Effects of Soft-Masking Function on Spectrogram-based Instrument – Vocal Separation," in *2019 16th International Conference of the Pacific Association for Computational Linguistics (PACLING)*, 2019, pp. 1–5.
[8] J. Shen *et al.*, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018, doi: 10.1109/ICASSP.2018.8461368.
[9] D. S. Bormane, S. D. Shirbahadurkar, and U. D. Shiurka, "Performance of Marathi language TTS synthesis based on perceptual test and spectrogram analysis," in *2010 The 2nd International Conference on Computer and Automation Engineering, ICCAE 2010*, 2010, doi: 10.1109/ICCAE.2010.5451850.
[10] H. Tora and B. Uslu, "Naturalness analysis of the speech synthesized by a TTS card," 2016, doi: 10.1109/siu.2016.7496096.
[11] FPT, "FPT AI," 2019. [Online]. Available: https://fpt.ai.
[12] Django Software Foundation, "Django," 2019. [Online]. Available: https://www.djangoproject.com/.