Vietnamese Speech Synthesis with End-to-End Model and Text Normalization

Do Tri Nhan*‡, Nguyen Minh Tri*‡, Cao Xuan Nam†‡
*Advance Program in Computer Science
†Faculty of Information Technology
University of Science, Ho Chi Minh City, Vietnam
‡Vietnam National University, Ho Chi Minh City, Vietnam

***Abstract*—Speech synthesis systems are now getting smarter and more natural thanks to the power of deep neural networks. However, each language has a different phonological and contextual characteristics, we have conducted experiments, statistics, and applied Vietnamese phonetics to improve speech synthesis systems based on Tacotron2 neural networks. Our methods achieve the accuracy of 97% in text normalization task, and the synthesized speeches with a MOS score of 3.97, asymptotic to 4.43 of the voices that are directly recorded. We also provide a library for standardizing Vietnamese text called Vinorm and a package that converts text into a phonetic format called Viphoneme, which is used as an input for end-to-end neural networks, make the synthesis process faster, more intelligent and natural than using character inputs.**

I. Introduction

Text-to-speech system (TTS) has many applications like generating audio from the text for news reading, producing music, or replacing people's voice in case that person loses their ability to speak. Google's translator has its TTS system that supports many languages and many news websites in Vietnam have supported the automatic TTS system so that readers can hear the news even though they are busy doing other things.

A. Text-to-Speech overview

Many methods have been used to generate natural speeches from texts like the Unit Selection method, Statistic based method, or the most modern technique - Deep Neural Network.

1) Unit Selection: Unit Selection is a concatenate synthesis method that focuses on synthesizing audio based on unit-level like character or phoneme [1]. A lookup table is generated based on a large dataset that has information on each unit like its frequency, duration, position and nearby units. This method can generate voices that are almost the same with human voices, however, more data are needed to generate more natural voices.

2) Statistic based model: Speech synthesis model based on statistics is also one of the most commonly used speech synthesis models. The most famous statistical model is Hidden Markov model [2], which is a statistical time series model that utilizes the speech results. The acoustic parameters created from HMM which are selected according to the language parameters are used to control vocoder. However nowadays, with the development of GPU performance, those statistical models are replaced by Deep Learning model.

3) Deep Neural Network: Deep Learning is a method that especially suitable for unstructured data like image, text and sound. The appearance of the CNN model boosts the performance of those Deep Neural Network based TTS models since we can extract more information and features from audio spectrogram. One of the advantage of this learning method is that it does not need expert knowledge, in contrast with the requirement of having a large amount of data [3].

4) End-to-End models: Due to the development of Deep Neural Network learning method, many TTS systems moved to use end-to-end models and gain significant improving results, such as Tacotron2 [4] and FastSpeech [5]. These systems do not use complex linguistic and acoustic features, they learn to produce audio directly from text, generate human-like speech using neural networks. The system first generates mel-spectrograms from texts and then using vocoder like WaveNet to generate audios. They are able to generate emotional, smooth and clean speech, works well on out-of-domain and complex words, learns pronunciations based on phrase semantics and robust to spelling errors [6].

B. Related Works: WaveGlow and Tacotron2:

A modern speech synthesis system consists of two main parts: mel-spectrogram generator and vocoder. In 2016, Wavenet was introduced, is a combination of wavelet and neural networks, this technique estimates waveform samples from given input feature vectors - mel-spectrogram in speech synthesis [7]. Wavenet is a vocoder, which improves the synthesis process better than previous techniques, but the weakness of wavenet is that sequential generation is too slow for production environments, leading to the introduction of a Flow-based and GAN-TTS approaches. Flow-based approaches can be mentioned as Parallel WaveNet [8], ClariNet [9], FloWaveNet [10], the most typical of which is WaveGlow [11].

With the mel-spectrogram generator, there are methods such as Feed-Forward Transformer [12] or Attention based [13], in which the most typical is Tacotron2 [4] and the latest is FastSpeech2 [14]. Tacotron2 includes a recurrent sequence-to-sequence feature prediction network that maps input text to

†Corresponding author. Email: cxnam@fit.hcmus.edu.vn

978-0-7381-0553-6/20/$31.00 ©2020 IEEE

mel-scale spectrograms, with a highlight that is the attention mechanism.

In order to apply these advanced models to Vietnamese, we need to standardize the data as well as propose using phonetic-based instead of character-based approach as an input of the neural network for taking the advantages of the Vietnamese language.

In this paper, we use Tacotron2 [4] and WaveGlow [11] for end-to-end Vietnamese speech synthesis system. We have rewritten the frontend of the speech synthesis system, we provide two python libraries:

Vinorm[1] to standardize text such as numeric characters, or abbreviations, local slang, and Viphoneme[2] to convert Vietnamese to grapheme format and from grapheme to International Phonetics Alphabet (IPA).

The rest of the paper is organized as follows: Section 2 reveals some of the major proposed methods on text normalization, text phonetization and speech synthesis with Tacotron2 model. Section 3 introduces the authors experiments on training and evaluation of our proposal and statistics about Vietnamese syllables usage in current online newspapers. Section 4 presents the conclusion and discusses some future works.

II. Proposed Methods

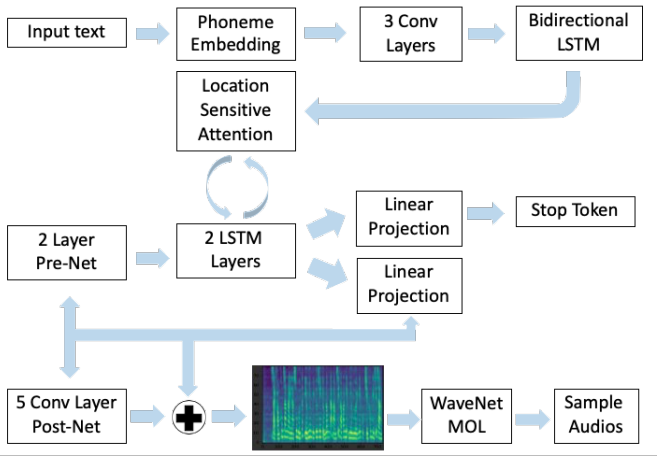

Figure 1: Tacotron2 architecture.

A. Vinorm: Text Normalization

Text Normalization is an important step in Text-to-Speech systems, helping to filter noise and making the input to be consistent with only Vietnamese syllables. The main task of the front-end of the TTS system is to standardize the text for the back-end system, the input is the raw text, we need to decide how to verbalize non-standard words, convert numbers, abbreviations, and words that cannot be pronounced into syllables, including dots, commas [15]. Every language needs different normalization processing methods because this problem is language-dependent [16]. It is impossible to build a complete text normalization because the language is ambiguous and evolves over time [17].

Text Normalization of Vietnamese Speech Synthesis today is still building grammars by hand instead of using automatic inference from large corpora because it has been the lack of annotated data [18]. To standardize text into readable words, the TTS system process through two steps, Rule-based and Dictionary-Checking.

1) Rules with Regular Expression: By using the Regular Expression to catch patterns that need normalization, which appears frequently in the language, containing numbers and characters, then replaced by syllables found in the dictionary. The output of this step is the paragraph without any digital characters.

Compared to the old VOS-frontend system [19], we do not lower the entire input before processing, thus we can handle cases with different pronunciation for uppercase and lowercase of the same words, and create a premise for backend processing when the capitalized words will be emphasized more through speech synthesis step.

We propose a new set of rules that are more systematic and general, scalable for future works. Based on the different contextual characteristics, rules are divided into four main categories to handle cases need standardization, including Special case, Timedate, Address and Mathematical. The patterns in each rule set will be proceeded one by one, browse through the entire text to match text need normalization, then return the corresponding normalized string

Special cases rules capture cases that are out of context, with specific formats, including Phone number, Football, Website, Email, etc.

For phone numbers, the identifying feature is a sequence of numbers starting with 0 or a plus sign, the number of digits from 10 to 14 digits. Phone numbers in each country will have a different way of writing, and in fact, there will be different formats, so the three types of phone number rules are listed, the pattern for capturing hotline numbers is also included. Website patterns have identifiable characteristics with a prefix is http(s), www, ftp or suffix is popular domains. Common email pattern is used for matching then spell each letter and symbol by English letter. Sport patterns include specific formats such as lineups, scores and hyphen-minus and dot is not spelled.

Time-date are rules for capturing phrases that show date and time elements. With the time indicating hours, minutes, and seconds, we identify by signals such as "h, g" or suffixed by AM / PM. These rules need to ensure the validity of time, if time is invalid, the system keeps them for later processing. If "-" followed by captured regex, it is read as "đến". Date patterns have a form such as DD / MM / YYYY or with the prefix “Ngày, sáng, trưa, chiều, tối, đêm, hôm, etc”. In addition, the timedate rules also capture and handle "FROM-TO" pattern cases.

Mathematical patterns capture cases containing normal number, floating Point Number, roman numerals, mathemati-

[1]https://pypi.org/project/vinorm
[2]https://pypi.org/project/viphoneme

cal expression or unit of measurement.

With normal number, we implement a function to convert numbers to letters. However, the normal number not only consists of successive digits, but also has a way of writing that splitting them into groups of 3 numbers, separated by commas, dot or space (e.g 12,000,000). This style of writing leads to confusion with floating point numbers, so to avoid false standardization, these cases will be matched first, then floating number, and finally normal number. For strings longer than 15 characters, each digit is converted individually, if the normalized text of number is too large, the string is be separated by commas to read more fluently. If there is an "- +" in front of the string, it is replaced with "cộng, “trừ”, except in the case where the number stands at the beginning of a sentence, "+ -" is treated as a bullet symbol. We categorized into two types of floating point, dot or comma. Floating point is replaced by "phẩy", the integer part is read as normal number and discard frontier zero, with fractional part, frontier zero is replaced with "không" and the following numbers read as normal number.

Unit of measurement are terms that refer to quantities, percentage or currency, followed by numbers, which can be an alphabetic or symbols. Because the system does not lower the input text, it can correctly standardize for upper and lowercase units (e.g Mbps and MBps). Units dictionary distinguish the upper and lower case. Besides the usual units, the patterns also need to capture "Unit/Unit" formats, matching strings are checked in the unit dictionary for readable words, if the matching is not in this list, we return the origin matching, keep for later processing. When it is sure that both sides of the slash are units, the "/" is replaced by "trên", this will avoid confusion in the case as "nam/nữ". Unit of measurement is also captured in FROM-TO pattern, there are two common types e.g: "10-20 km/h" and "10 km/h - 20 km/h", our system make sure not to be confused with cases where "-" is read as "minus".

For Roman numerals, to ensure accuracy does not fail in capturing, comparing to the old VOS, we just consider uppercase [X, I, V] is able to be Roman numerals, [L, C, D, M] are also Roman characters, but they rarely appear in the text, sometimes even leading to confusion with acronyms. The Roman numerals also have a FROM-TO structure, such as "XVI-XXI", but it is easier to handle and more consistent than the unit of measurement. The matching is checked for correctness by converting it into a decimal number, then converting the result back into roman form to compare with the original matching, then the matching is converted to normalized text.

Address-Code are rules for capturing phrases about addresses, locations, codes and all phrases containing numbers. Codenumber pattern match all remain cases with numbers, the matching sequence will be trim punctuation at both sides and separated into each alphanumeric substrings (e.g: MH370 is split into MH and 370). For each substring, if it is a fully capitalized letter sequence, it will be transcribed, if it contains lowercase letter, the system will keep that letter clusters and add space separate for each cluster. If the substring is numeric and the length of continuous number string is greater than 4, each number will be transcribed, otherwise, it will read the whole number as a normal number. Cases include special character or symbol will be mapped with corresponding phonetics, but with "-" is not spelled.

2) Dictionary Checking: After running through the rules sets, the string just contains letters, space and special character. This string will be split into many segments which separated by spaces, each token containing no space and no special characters before and after the string. The system runs over each token to validate if it is readable by checking it in Dictionary Vietnamese syllable which includes 7698 syllables. If the token does not exist, we look it up in the mapping dictionary instead, search and replace it with the corresponding word.

Abbreviations mapping dictionary includes 3 different mapping methods. With acronyms such as "NSƯT, GDĐT", we have to normalize to its original form. The second type is initialisms, including words like STEM and UNESCO, so we have to transcribe it. The last type is the acronyms that need to spell each letter such as PNJ, FPT, AFF, we do not handle it in this step and consider it as unknown to limit misunderstandings when not sure.

With dictionary mapping abbreviations, uppercase and lowercase tokens are often referred as two separate objects, with different contexts and probabilities, an acronym can have more than one meaning. Acronyms that appear less frequently are filtered out to limit misunderstandings

Teen Code - Slang - Lingo mapping dictionary is updated by adding more than 300 slangs like "H’Hen Nie, Ea H’leo", added some lingo that not appear in Popular Dictionary, words that backend doesn’t support like “Đắk Lắk, Pleiku”. It also handles inconsistency in writing of the same word, such as "thuỷ - thủy" or "tuỳ - tùy". Loanwords like "oxy, axit" and common misspellings are also updated.

Special Symbols After browsing through the dictionaries, if the token does not belong to any Mapping Dictionaries, we continue to split the token into smaller substring separated by symbols and special characters. The system shows the corresponding reading for each symbol, non-stop punctuation such as brackets, quotes are removed. The system continues checking each substring in the mapping dictionaries again, this process (tokenize strings with space first, if the token is unknown, then tokenize with special character) avoids errors when the string is written adjacent to each other. This way will handle both cases “GD-ĐT” and “ê-kíp”, hyphen in the first case will be omitted to "GDĐT", the second case will be replaced with space to "ê kíp", and many similar cases with symbolic problems are also solved. If the token is still not in any mapping dictionary, we treat it as unknown word, if it is uppercase, spell each character as English letter, if it is lowercase and containing vowels, we will leave the backend to handle by letter to sound, if it does not contain the vowel, spell each character as Vietnamese letter.

Punctuations: The final step is to standardize the output text, including removing duplicate white spaces, handling

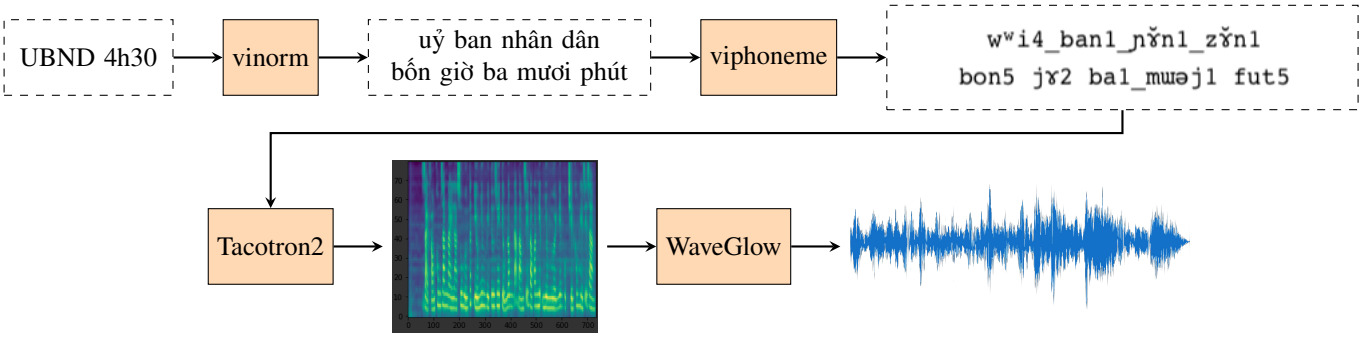

Figure 2: Proposal method pipeline.

punctuations including removing no voice marks like "()[]", and replace all punctuation marks with only comma and dot, which represent as the sound unit.

B. Viphoneme: Text Phonetizer

In order to synthesize words that have never appeared in the train set or out of vocabulary words (OOV), we now use the grapheme instead of the character as the input for the end-to-end model. This makes the model converges faster.

In order to represent mix-codes, foreign languages, languages need to have a uniform form of representation such as IPA or ARPABET. We have replaced the grapheme representation symbols with the IPA representation, which is both more concise than the ARPABET format, which many languages can be converted into.

Because IPA is for describing sound, we not only create general lexicon for Vietnamese but it also depends on the speaker's own region in the train (dialect). The presentation of IPA for Vietnamese has many ways, and is still not unified. We refer to the method of Pham 2006 [20], customize the way some phonemes represent and some labiovelar on-glide. Because IPA does not display tones, we have signed the blanks, grave, acute, hook, tilde, dot accents with the numbers from 1 to 6. The output will be in the following format:

$$(C1)(w)V(G|C2)+T$$

With C1 is initial consonant onset, w is labiovelar on-glide, V is vowel nucleus, G is off-glide coda, C2 is final consonant coda and T is tone. For example, the word "xuống" is parsed into structure as above "x-u-ô-ng-2", then these grapheme are replaced with IPA symbols.

Some special cases when converting from raw text to phoneme format such as the unification of the position of tones in words or the elimination of words, e.g: quyết -> qu-uyê-t-2. We then ran through the list of syllables in Vietnamese to make sure that all the words were unique.

C. Audio Processing and Model Configuration:

To be able to generate speech as closely as possible to human voices and match Wavenet's input, we reduce the sampling rate of each input audios data from 44100 Hz to 22050 Hz by using FFmpeg library to ensure the audio sample rate changes but keeps the speech rate unchanged. We remove silence at the start and the end, then adding one second silence to the end of each audio in order to help the model to recognize the end of the sentence better.

Finally, we use the vinorm and viphoneme package as mentioned above to change our text from normal Vietnamese characters to Vietnamese phonemes in IPA form. There are a total of 144 IPA characters including tones, Vietnamese phoneme, English phoneme, dot, comma, and other special characters, each IPA character will be mapped corresponding to a number. Therefore the input text will be converted to a sequence of numbers and this sequence will be the input for the embedding layer.

Our model almost the same with the vanilla Tacotron2 model, with changing from Character Embedding to Phoneme embedding, based on our suggested text normalization method.

III. Experiment

A. Vietnamese Syllables Statistics

In order to update the Vietnamese syllables used today, we proceeded to build a News Corpus with sources of 9 online newspapers, crawled from 7/2018. The Corpus is divided at the sentence level, consisting of a total of 6,308,173 sentences, the number of unnormalized sentences is 3,740,507 sentences, accounting for more than 59.29 %.

dantri	danviet	nld	thanhnien	tto
653545	386444	103218	634974	177287
tuoitre	vnexpress	vnn	zingnews	
167162	157202	481566	979109	

Table I: Number of sentences from popular Vietnamese newspapers

We continue to word-tokenize every sentence, totaling 10.88 million tokens. The tokens will be classified into groups such as Special Case, Time-date, Math and Number, English, Slang, Teencode, Acronyms, Initialism, Proper Noun, etc.

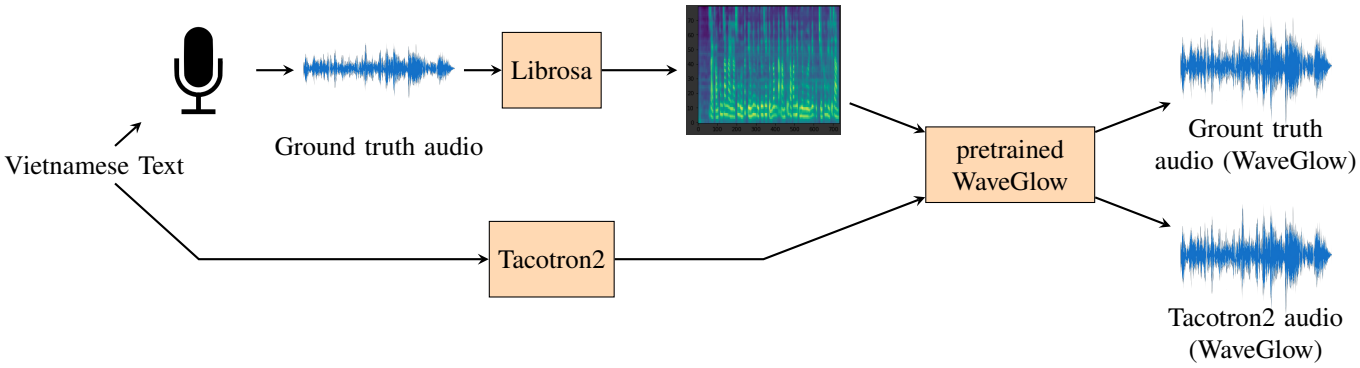

Figure 3: Ground truth (WaveGlow) generating process.

Category	Percent	Number of sentences
Special	0.2	21383
Timedata	1.21	131341
Math	22.67	2467113
English	35.65	3880214
Acronyms Lowercase	0.01	830

Category	Percent	Number of sentences
Acronyms Uppercase	6.31	686907
Teencode	0.01	944
Intialism	9.07	987432
Proper	9.17	998180
Other	15.71	1709682

Table II: Statistics on Tokens need to be standardized in Vietnamese

Some notable points from the statistics on News Corpus are: special cases only account for 0.2%, most of the Timedate cases are the publication date of the news, English accounts for 40 percent, and most of the abbreviations are all in uppercase.

B. Vietnamese Normalization and Phonetization

To select the language for the text normalization problem, we decided to choose C ++ because it is suitable for Cross-Platform Deployment, has a fast running time, uses less memory than Perl and is compatible with the current VOS backend. Some famous frameworks for text to speech systems also use C and C ++ such as Festival Speech Synthesis System of University of Edinburgh [21], Flite of CMU [22], Hts-engine use for Jtalk, Sinsy, and eSpeak is also written in C and C ++ [23].

One of the problems when using C ++ is to handle Vietnamese Unicode, we use ICU4C library version 64.2, an International Components for Unicode. This library is opensource, well-documented, robust and reliable. The ICU provides basic regular expression operators and especially Case Insensitive Matching, which helps the regular expression to capture both uppercase and lowercase letters, preserving the properties of the input text, which will be beneficial for handling in back-end steps, helping voice more natural, change the overall intonation like stress on capitalized words.

We provide a python package on Ubuntu 18.04 that can be installed at the Python Package Index called ViNorm.

From the data collected as mentioned above, we extracted 100 tricky need-normalized cases[3] to use as the baseline for improvement, 500 random cases in practical contexts for testing our proposal. These test cases do not include normal sentences, foreign words and proper nouns. With the 500 test cases, we improved the frontend of VOS from 60% to 97%. Some cases are wrong when mapping acronyms due to its plurality, such as BTC, we can read as "Ban tổ chức", "Bộ tài chính", or it can also be a stock symbol.

Method	Accuracy
Front-end VOS 2.0	60%
Updated Front-end VOS	97%

C. Speech synthesis

1) Dataset: The dataset used in this experiment is provided by InfoRe Jsc, which is also the Big Corpus set in International Workshop on Vietnamese Language and Speech Processing (VLSP) 2019 [24]. The dataset included about 22 hours with 13,462 utterances of north-accent female Vietnamese. Because the data set contains lots of noisy audio, we filtered out and removed more than 2000 samples, many of samples that the reader stopped in the wrong place also affect the training process.

2) Training: We train the model with Nvidia Quadro k6000 and use a batch size of 32, with learning rate is 10^{-5}. We run the model with 200 epochs and it converges after the 134000^{th} iteration. The total amount of training is about 240 hours, approximately 10 days.

Some audio parameters for training models such as filter length are 1024, hop length is 256, window length is 1024, number of mel channels is 80.

3) Results and Evaluation: Since we don't train WaveGlow model, we use the pretrained of WaveGlow provided by NVIDIA, which is able to generate good results on both English and Vietnamese.

[3]https://github.com/NoahDrisort/NICS_Appendix

To prevent the result from being the maximum decoder step, we decrease the gate_threshold parameter to 0.05 so that the result will have appropriate time length. The resulted audios still contain a lot of noise so we reduce them by using the noise reduction API and then, increase the sound level.

To evaluate the Tacotron2 model when applied to Vietnamese, without being dependent on vocoder waveglow, ground truth audios for testing are converted into mel-spectrograms and then converted these mel-spectrograms back to audios by using pre-trained WaveGlow as shown in fig. 3. This processed ground truth is called Groundtruth (Mel + WaveGlow), they will be compared with voices synthesized and standardized by our model.

To evaluate the result, we choose the MOS (mean opinion score) scoring system on the test set to check the quality of audios [25]. Each person who joins the survey listens to 40 audios, which include 20 audios generated from Tacotron2 and 20 corresponding ground truth audio generate from the process. They were asked to grade from 5 to 1, based on how natural and smooth those speeches compare to real human speeches. The final score of each audio type will be equal to the total score divides by the number of survey participants. Below is the MOS result gain from the survey of at least 20 people.

Model	MOS
Tacotron2 (WaveGlow)	3.97
Groundtruth (Mel + WaveGlow)	4.43

IV. Conclusion

Through statistics, we have given an overview of the standardized text of a Vietnamese speech synthesis system, the text normalization proposals for TTS systems were introduced and packaged in Pypi is called Vinorm. In this package, we improve the numeric processing modules, the processing special character module to produce the result to match the context of document. Larger dictionaries are used to cover more words and implementing module to recognize words which have different pronunciation in uppercase and lowercase. Updating for the regular expression step and the expansion of mapping dictionaries have helped VOS solve many special cases, especially for tokens that contain both numbers and letters. The results when compared to the VOS 2.0 text standardizer were superior when handling specific cases. Continuing with this result, the Viphoneme package was introduced as a bridge that converts Vietnamese text input into the input sequence of the Tacotron2, an end-to-end neural network model. The output has natural and fluent voice, quite similar to the real voice used for training with a MOS score of 3.97. This result opens up new directions in our research with a modern end-to-end model in Vietnamese, and it is possible to improve the result with a clean dataset with fewer noise samples.

Acknowledgement

This work is supported by AILAB and research funding from Advanced Program in Computer Science, University of Science, Vietnam National University - Ho Chi Minh City

References

[1] S. Kayte, M. Mundada, and C. Kayte, "A review of unit selection speech synthesis," vol. 5, p. 5, 11 2015.
[2] S. Kayte, M. Mundada, and J. Gujrathi, "Hidden markov model based speech synthesis: A review," *International Journal of Computer Applications*, vol. 130, pp. 975–8887, 12 2015.
[3] Y. Ning, S. He, Z. Wu, C. Xing, and L.-J. Zhang, "A review of deep learning based speech synthesis," *Applied Sciences*, vol. 9, p. 4050, 09 2019.
[4] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," 2017.
[5] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," 2019.
[6] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," 2017.
[7] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," 2016.
[8] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel wavenet: Fast high-fidelity speech synthesis," 2017.
[9] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," 2018.
[10] S. Kim, S. gil Lee, J. Song, J. Kim, and S. Yoon, "Flowavenet : A generative flow for raw audio," 2018.
[11] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," 2018.
[12] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6706–6713, 07 2019.
[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.
[14] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," 2020.
[15] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. (2001) Normalization of non-standard words.
[16] M. Chu, H. Peng, and Y. Zhao. (2009, Feb. 24) Front-end architecture for a multi-lingual text-to-speech system. US Patent 7,496,498.
[17] D. Yarowsky. (1993) Text normalization and ambiguity resolution in speech synthesis.
[18] R. Sproat. (2010) Lightly supervised learning of text normalization: Russian number names.
[19] V. Q. D. Ha, N. M. Tuan, C. X. Nam, P. M. Nhut, and V. H. Quan. (2010) Vos: the corpus-based etnamese text-to-speech system.
[20] A. H. Pham, "Vietnamese rhyme," *Southwest Journal of Linguistics*, vol. 25, pp. 107–142, 01 2006.
[21] A. W. Black, P. Taylor, and R. Caley, "The festival speech synthesis system: system documentation," 1997.
[22] M. H. Lee, "Migrating dari clustergen flite text-to-speech voice from desktop to android," 2014.
[23] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. Black, and K. Tokuda, "The hmm-based speech synthesis system (hts) version 2.0," *Proc. of ISCA SSW6*, 09 0002.
[24] N. T. T. Trang and N. X. Tung, "Text-to-speech shared task in vlsp campaign 2019: Evaluating vietnamese speech synthesis on common datasets."
[25] R. C. Streijl, S. Winkler, and D. S. Hands. (2016) Mean opinion score (mos) revisited: methods and applications, limitations and alternatives.