WORD EMBEDDING FOR RECURRENT NEURAL NETWORK BASED TTS SYNTHESIS

Peilu Wang[1,2*] *Yao Qian*[2] *Frank K. Soong*[2] *Lei He*[2] *Hai Zhao*[1]

[1] Shanghai Jiao Tong University, Shanghai, China
[2]Microsoft Research Asia, Beijing, China
{v-peiwan,yaoqian,frankkps, helei}@microsoft.com, zhaohai@cs.sjtu.edu.cn

ABSTRACT

The current state of the art TTS synthesis can produce synthesized speech with highly decent quality if rich segmental and suprasegmental information are given. However, some suprasegmental features, e.g., Tone and Break (TOBI), are time consuming due to being manually labeled with a high inconsistency among different annotators. In this paper, we investigate the use of word embedding, which represents word with low dimensional continuous-valued vector and being assumed to carry a certain syntactic and semantic information, for bidirectional long short term memory (BLSTM), recurrent neural network (RNN) based TTS synthesis. Experimental results show that word embedding can significantly improve the performance of BLSTM-RNN based TTS synthesis without using features of TOBI and Part of Speech (POS).

Index Terms— Speech synthesis, TTS, BLSTM, RNN, word embedding

1. INTRODUCTION

With decades of development, the state-of-the-art TTS synthesis systems now can achieve good performance, but all of these systems highly rely on rich manually labeled features. Labeling such features is expensive and relying on such features limits the application of TTS synthesis system since in some cases such as on-line voice-to-voice translation, it is impossible to manually label such rich features. Although some features such as POS can be predicted well by using machine learning method, the prediction of some key features such as the tone and break index (TOBI) are still far from accurate since the limited training corpus. Considering only a few of text-to-speech training corpora are labeled with TOBI features, it is worthwhile to develop the TTS synthesis system that does not rely on such features and can be trained directly on text-to-speech corpus.

Recently, neural network (NN) based TTS synthesis systems have shown fairly good performance. [1] proposes a TTS synthesis system using deep neural network (DNN) and shows that this DNN based system outperforms the conventional hidden Markov model (HMM) based system. [2] further explores the usage of DNN in TTS and also proves DNN's superior performance. Bidirectional long short term memory (BLSTM) recurrent neural network (RNN) [3, 4, 5] which can make use of features of a long context is introduced to replace DNN in [6, 7]. The corresponding results show that RNN based TTS performs better than both HMM and DNN. Despite of its success in NN based systems, one specific advantage of neural network is that neural network models trained on different corpora or tasks can share part of their layers. This makes neural network model easy to take advantage of big data by using some parts of layers trained on other tasks which have large training data. With the help of big data, neural network can achieve comparably good performance without using rich manually labeled features. One typical and most commonly used shared part of layers is word embedding.

Word embedding is a low dimension continuous-valued vector that are used to represent word. It becomes popular in recent years with the success of neural network language model. In recent publications about word embedding, [8, 9] achieve word embedding by training a neural network language model . [10, 11, 12, 13, 14] get word embedding by training a neural network model to predict current word given the preceding and succeeding words. One common point of these works is that they all make use of large unlabeled text data. [14] are trained on corpus with 10 billion words and the sizes of the other works' training corpora are all over 10 million words. With the help of word embedding trained on big data, [12] reports their neural network models with only few basic features achieve comparably good performance as state-of-the-art methods using rich manually labeled features in dealing with 4 basic natural language processing tasks. [15] also proves the help of word embedding while their task is predicting the phrase-break position in sentences for TTS synthesis. [16] has tested DNN based TTS system with vector space representation and its result is not very positive. However, it should be noted that vector space representation is a byproduct of singular value decomposition of a matrix of an information retrieval task [17], while word embedding discussed in this paper are all trained using neural

*Work performed as an intern in the Speech Group, Microsoft Research Asia

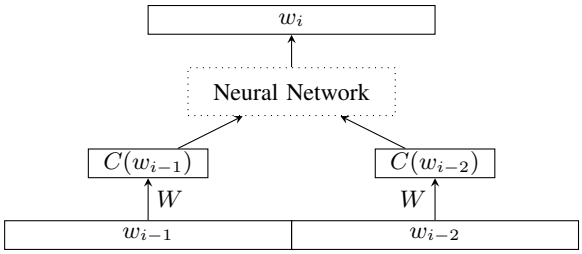

Fig. 1. Structure of neural network language model.

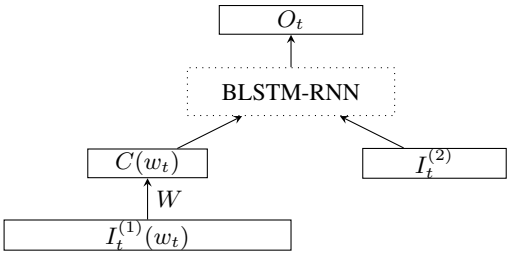

Fig. 3. Structure of BLSTM-RNN with word embedding.

network models on a much larger corpus.

In this paper, we alter the BLSTM-RNN based TTS synthesis system proposed in [7] to make use of word embedding. The alteration for using word embedding is discussed in Section 2. Four types of published word embeddings [9, 10, 12, 14] are tested. Experimental results and analysis are presented in Section 3 followed by conclusions.

2. BLSTM-RNN BASED TTS SYNTHESIS WITH WORD EMBEDDING

2.1. BLSTM-RNN based TTS synthesis system

The schematic diagram of BLSTM-RNN based TTS synthesis system is shown in Fig 2.

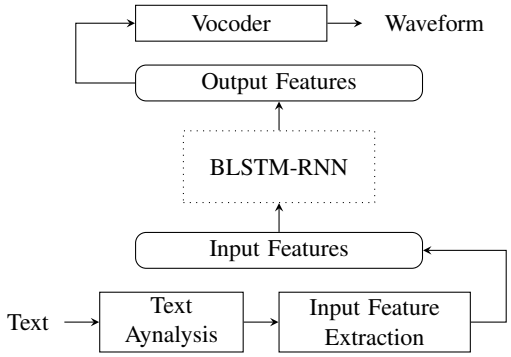

Fig. 2. BLSTM-RNN based TTS synthesis system.

In this system, a text sentence is given as input and the output is the speech of this sentence. The sentence is first converted to a sequence of input features I_1,I_2,...,I_T where I_t denotes the t-th input feature at frame t. Each input feature I_t is composed of the binary features for categorical contexts, e.g. phone labels, POS labels of the word, and TOBI labels, and numerical features for the numerical contexts, e.g., the number of words in the phrase or the position of current frame in the current phone. Then, input features are mapped to output features O_1,O_2,...,O_T by a trained BLSTM-RNN where O_t denotes the t-th output feature at frame t. The output features are composed of fundamental frequency (F0), voiced/unvoiced flag and line spectral pairs (LSP). Finally, a waveform synthesis module is used to transform the output features to speech waveform.

In the training phase, input features and output features of training corpus are time-aligned frame-by-frame by well-trained HMM [18]. BLSTM-RNN is then trained on a sequence of input and output feature vector pairs which can be regarded as a data set of a typical sequence labeling task.

2.2. Achieving word embedding

Word embedding can be achieved by training a neural network language model (NNLM) to predict current word's probability distribution given previous words. Fig 1 illustrates the structure of a typical NNLM [8]. In this sample, 2 previous words are given as the input and the output is current word. All words are represented with one hot representation whose dimension is $|V|$ where V is the vocabulary. Each input word vector w_x is mapped to $C(w_x)$ which has a much lower dimension m by multiplying a weight matrix $W(m \times |V|)$: $C(w_x) = Ww_x$. As the training of other weights in neural network, W is also initialized with random values and updated by back-propagation algorithm. The main difference between training W and training an additional layer is that all input words share one W. When the training of this NNLM is finished, word embedding of w_x is also achieved which is $C(w_x)$. Once word embeddings are trained, they can be easily applied to other task's neural network model by just substituting W with the well trained W.

In our experiments, four types of published well-trained word embeddings are involved. They all achieve word embeddings by training a neural network model on unlabeled text and their usages of word embeddings are the same as that illustrated in Fig 1: mapping input word to a lower dimension word embedding by W and updating W during training the neural network model. More details about these word embeddings and their differences are presented in Section 3.

2.3. Using word embedding

The usage of word embedding in BLSTM-RNN for TTS synthesis is shown in Fig 3. Here $I_t^{(1)}$ is current word's one hot representation in frame t and $I_t^{(2)}$ is the all features except

for TOBI and POS related features. O_t is the output feature vector of frame t. As in NNLM, w_t is mapped to its word embedding $C(w_t)$ by multiplying matrix W, then $C(w_t)$ and $I_t^{(2)}$ are cascaded as the input of BLSTM-RNN. In practice, the role of W is implemented by a dictionary D that records word embedding of each word ($D[w_t] = Ww_t$). The dictionary can be kept fixed or be updated by the back-propagated errors of BLSTM-RNN's input layer during the training process. The update algorithm is shown in Algorithm 1, where

Algorithm 1 update algorithm

1: **for** i in $1...m$ **do**
2: $D[w_t]_i$= $D[w_t]_i$-$learningrate * bkerror_i$
3: **end for**

$D[w_t]$ returns word w_t's word embedding, $D[w_t]_i$ is the i's dimension of this word embedding and $bkerror_i$ is the back-propagated error of i's node in the input layer.

The update of D is actually independent with the update of BLSTM-RNN. It is not required that D is updated in every training epoch, thus various of update strategies can be used such as updating D every n epochs. In our approach, we first fix D and update RNN until no best validation score appears, then fix RNN and update D. These two steps are repeated until no improvement is observed. This strategy lead to a better performance than simply updating D in every epoch.

BLSTM-RNN has been proven to be capable of capturing non-local information in a sequence [3, 4, 5]. Therefore, in our approach, only current word's word embedding is added to the input features and the latent dependencies of these additional features along an utterance are expected to be learned by BLSTM-RNN. The system in our experiments is altered on CURRENT, a machine learning library for RNN which uses GPU to accelerate the computations [19].

3. EXPERIMENTS AND RESULTS

3.1. Experimental setup

A corpus of a female, American English, native speaker is used in all experiments. The corpus consists of 5514 utterances (about 5 hours). 5050 of these utterances are used for training, 264 are used for validation and 200 are used for testing. Speech data are sampled at 16 kHz, windowed by a 25 ms window, and shifted every 5 ms.

A group of systems are designed to test the performance of BLSTM-RNN based TTS synthesis system with different word embeddings. The basic information of involved word embeddings and their sources are listed in Table 1, where the unit of training corpus's size is word and RCV1 represents the Reuters Corpus Volume 1 news set. *WR1* to *WR6* are trained by published works and are available to download from Internet. For each word embedding, the detailed training method can be found in corresponding work listed in Ref column and the URL to download can be found in Src column. Note that *WR2* to *WR4* are trained with same method and corpus but with different dimension. They are used to explore the effect of dimension of word embeddings.

Table 1. Tested word embeddings.

Alias	Dim	Vocab Size	Train Corpus	Ref	Src
WR1	80	82K	Broadcast news (400M)	[9]	[20]
WR2	25	269K	RCV1 (37M)	[11]	[21]
WR3	50	269K	RCV1 (37M)	[11]	[21]
WR4	100	269K	RCV1 (37M)	[11]	[21]
WR5	50	130K	RCV1+Wiki (221M+631M)	[12]	[22]
WR6	300	3M	Google news (10B)	[14]	[23]

All of these systems share the same output features and neural network structure. The output features of all these systems consist of voiced/unvoiced flag and log F0, 40 LSP and gain, totally 1+1+40+1=43 dimensions. The neural network structure of all systems is 4 hidden layers with 512 nodes per layer, where the structure of bottom 2 hidden layers is feed-forward while the structure of top 2 hidden layers is BLSTM. This structure follows the configuration in [7] and it also achieves the best performance compared with other structures, i.e., different layers and nodes per layer combination, according to our experiments.

Before training, log F0 values in unvoiced states are interpolated to keep the continuity of acoustic features within a sentence. Both input and output features are normalized to have zero-mean and unit-variance. The weights of RNN are initialized randomly and are trained using mini-batch back-propagation algorithm with constant learning rate.

3.2. Evaluation results and analysis

Each system's performance is assessed objectively by evaluating the distortion between predicted output features and actual output features in the test set. The objective measures include the root mean squared error (RMSE) of F0, voiced/unvoiced (V/U) error rate and log spectral distance (LSD) of LSP. Subjective measure is also used as an additional evaluation.

There are four types of BLSTM-RNN based TTS systems with different input features. We split all possible input features into three subsets:$I^{(1)}$, $I^{(2)}$ and $I^{(3)}$, where $I^{(1)}$ is word embedding, $I^{(2)}$ contains phoneme identity, the position of phoneme, the position of syllable and word in phrase and sentence, the length of syllable and word and state index, involving all input features of [7] except those manually labeled features, i.e., POS and TOBI, and $I^{(3)}$ is composed of POS and TOBI. These four types of systems are configured as,

1) *rnn*: baseline system, using input features $I^{(2)}$;

2) *rnn_WR#*: using input features $I^{(1)}$ + $I^{(2)}$ (WR# is the alias of involved word embedding listed in Table1);

3) *rnn_upper_bound*: upper bound system, using input features $I^{(2)}$ + $I^{(3)}$, in which $I^{(3)}$ are manually labeled for both training an testing sets;

4) *rnn_ub_WR1*: using input features $I^{(1)}$ +$I^{(2)}$ + $I^{(3)}$.

Table 2 presents objective evaluation results of these systems. All systems making use of word embedding achieve an improvement of the baseline system in F0 RMSE and LSD, half of these systems get improvement in V/U Error. *rnn_WR1* achieves the greatest improvement which is fairly notable although the performance still falls behind *rnn_upper_bound*. It shows that the best performed word embedding can achieve over 50% of the prosody improvement brought over by the manually labeled features, compared with the one trained with baseline features (*rnn*) and with state-of-the-art features (*rnn_upper_bound*). Note that not all word embeddings get a notable improvement and some even get a worse performance in V/U Error. The training corpus, corpus's size and training neural network's structure of word embedding may attribute to this discrepancy. Therefore, exploring the effect of these factors is one of future directions and it is expected that the performance can be further improved with fine tuned word embeddings. Besides, *rnn_WR2* to *rnn_WR4* show that the dimension of word embedding is not a very important factor. *rnn_WR3* and *rnn_WR5* show that, in out task, the training corpus size may not so important as [14] mentioned. While word embedding brings improvement to *rnn*, *rnn_ub_WR1* do not show a further improvement compared with *rnn_upper_bound*. It is conjectured that word embedding can provide part of information involved in manually labeled features, so it fails to provide further help when those features are used.

Table 2. Objective evaluation results of tested systems.

Alias	F0 RMSE (HZ)	V/U Error (rate)	LSD (dB)
rnn	17.53	2.25%	3.69
rnn_WR1	16.04	2.22%	3.60
rnn_WR2	17.15	2.22%	3.66
rnn_WR3	16.79	2.32%	3.63
rnn_WR4	17.24	2.30%	3.63
rnn_WR5	16.75	2.34%	3.63
rnn_WR6	17.49	2.00%	3.66
rnn_upper_bound	14.85	1.81%	3.63
rnn_ub_WR1	14.93	1.92%	3.61

In experiments above, weights of word embedding are fixed during the training process. In fact, weights can be tuned following Algorithm 1. Table 3 shows the result of *rnn_WR1_tuned* which tunes word embedding during training. Tuning word embedding during training leads to a further improvement in all three measures, but not notable which is consistent with [12, 15].

Table 3. Performance of tuning word embedding.

Alias	F0 RMSE (HZ)	V/U Error (rate)	LSD (dB)
rnn_WR1	16.04	2.22%	3.60
rnn_WR1_tuned	16.02	2.18%	3.58

In subjective evaluation experiments, two AB preference tests are conducted through Amazon Mechanical Turk [24]. In each test, 50 randomly selected texts from test set are synthesized by two tested systems to generate a set of utterance pairs. For each pair, 20 native English speakers are asked to choose which one of the synthesized utterance is better or choose no preference. The proportion of these choices are averaged and regarded as the criterion of current AB test. Table 4 shows the results.

Table 4. Results of two AB tests where neutral denotes "no preference" and p is the t-test p-value of each test. Both results are statistically significant.

rnn_WR1_tuned	rnn	neutral	p-value
38%	29%	33%	$< 10^{-3}$
rnn_WR1_tuned	rnn_upper_bound	neutral	p-value
30%	36%	34%	$< 10^{-2}$

In the result of AB test between *rnn_WR1_tuned* and *rnn*, participants on average prefer more utterances (38%) synthesized by *rnn_WR1_tuned* than the baseline system *rnn* (29%). The second test shows that *rnn_WR1_tuned* (30%) falls behind *rnn_upperbound* (36%). (Some samples of converted utterances are given on: `http://research.microsoft.com/en-us/projects/dnntts/default.aspx`) The subjective results show that word embedding can significantly improve the performance of the baseline system, which doesn't use TOBI and POS as input features, although it still has a gap to the upper bound system, which uses manually labeled POS and TOBI as input features for both training and testing sets.

4. CONCLUSIONS

In this paper, we investigate the use of word embedding for BLSTM-RNN based TTS synthesis system. Four different kinds of published word embeddings are tested. Experimental results show that word embedding can notably improve the performance of BLSTM-RNN based TTS synthesis system without using manually labeled features. The improvement varies a lot with different word embeddings trained on different corpora and tasks. It is expected that using a fine tuned word embedding can achieve a further improvement.

5. REFERENCES

[1] Heiga Zen, Andrew Senior, and Mike Schuster, "Statistical parametric speech synthesis using deep neural networks," in *ICASSP*, Vancouver, BC, Canada, May 2013, pp. 7962–7966.

[2] Yao Qian, Yuchen Fan, Wenping Hu, and Frank K. Soong, "On the training aspects of deep neural network (DNN) for parametric TTS synthesis," in *ICASSP*, Florence, Italy, May 2014, pp. 3829–3833.

[3] Alex Graves, *Supervised sequence labelling with recurrent neural networks*, vol. 385, Springer, 2012.

[4] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] Mike Schuster and Kuldip K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[6] Raul Fernandez, Asaf Rendel, Bhuvana Ramabhadran, and Ron Hoory, "Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks," in *INTERSPEECH*, Singapore, September 2014.

[7] Yuchen Fan, Yao Qian, Fenglong Xie, and Frank K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *INTERSPEECH*, Singapore, September 2014.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "Neural probabilistic language models," in *Innovations in Machine Learning*, pp. 1137–1155. Springer, 2003.

[9] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan "Honza" Cernocký, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, Makuhari, Chiba, Japan, September 2010, pp. 1045–1048.

[10] Andriy Mnih and Geoffrey Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th international conference on Machine learning*, Corvallis, Oregon, USA, June 2007, pp. 641–648.

[11] Joseph Turian, Lev Ratinov, and Yoshua Bengio, "Word representations: A simple and general method for semi-supervised learning," in *ACL*, Uppsala, Sweden, July 2010, pp. 384–394.

[12] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[13] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng, "Improving word representations via global context and multiple word prototypes," in *ACL*, Jeju Island, Korea, July 2012, pp. 873–882.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *CoRR*, January 2013.

[15] O. Watts, S. Gangireddy, J. Yamagishi, S. King, S. Renals, A. Stan, and M. Giurgiu, "Neural net word representations for phrase-break prediction without a part of speech tagger," in *ICASSP*, Florence, Italy, May 2014, pp. 2599–2603.

[16] Heng Lu, Simon King, and Oliver Watts, "Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis," in *8th ISCA Speech Synthesis Workshop*, Barcelona Spain, August 2013, pp. 281–285.

[17] Oliver Watts, "Unsupervised learning for text-to-speech synthesis," *Ph.d. dissertation*, 2013.

[18] Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *ICASSP*, Istanbul, Turkey, June 2000, vol. 3, pp. 1315–1318.

[19] Weninger Felix and Geiger Jrgen, "Current," `http://sourceforge.net/projects/currennt/`.

[20] Tomas Mikolov, "Rnnlm," `http://rnnlm.org/`.

[21] Joseph Turian, Lev Ratinov, and Yoshua Bengio, "Word representations for NLP," `http://metaoptimize.com/projects/wordreprs/`.

[22] Ronan Colloberte, "Senna," `http://ml.nec-labs.com/senna/`.

[23] Tomas Mikolov and etc, "word2vec," `https://code.google.com/p/word2vec/`.

[24] "Turk," `https://www.mturk.com/`.