Vol.2016-SLP-110 No.7
2016/2/6

Using Continuous Representation of Various Linguistic Units for Recurrent Neural Network based TTS Synthesis

Xin Wang[1,a)] Shinji Takaki[1,b)] Junichi Yamagishi[1,c)]

Abstract: Building high-quality text-to-speech (TTS) systems without expert knowledge of the target language and/or manual time-consuming annotation of speech and text data is an important and challenging research topic in speech synthesis. Recently, the distributed representation of raw word inputs, called "word embedding", have been used in various natural language processing tasks with success. Moreover, the word-embedding vectors have recently been used as the additional or alternative linguistic input features to a neural-network-based acoustic model for TTS systems. Since word-embedding approaches may provide means for obtaining effective linguistic representations from texts without requiring specialized knowledge of the language and/or manual time-consuming annotation, we further investigated the use of the word-embedding for neural-network-based TTS systems in two new directions. First, in addition to the standard word embedding vectors, we attempted to use phoneme, syllable, and phrase embedding vectors to verify whether continuous representations of these linguistic units may improve the segmental and suprasegmental quality of synthetic speech. Second, we examined the impact of normalization methods on the obtained embedded vectors before they were feed into the neural-network-based acoustic model.

Keywords: Text-to-speech, Speech synthesis, Recurrent neural network, Contexts, Word embedding

1. Introduction

Text-to-speech (TTS) synthesis converts text strings into speech waveforms. Due to the non-linear relationships between text and speech, TTS is normally decomposed into front-end and back-end. The front-end performs linguistic analysis and symbolic prosody prediction in order to obtain intermediate linguistic representations between speech and text. Based on the intermediate representations, the back-end performs acoustic feature predictions and synthesize the speech waveform.

The front-end can be further divided into smaller yet specific sub-modules. For English TTS, they include a) grapheme-to-phoneme conversion (G2P), b) syllabification, c) part-of-speech (POS) tagging, d) syntactic parsing, e) symbolic prosody prediction, and so on [1]. These sub-modules are usually statistic models trained using databases in which correct labels are carefully and manually annotated. This results in very accurate and high-quality synthetic speech. However, it is laborious to collect such the databases. This becomes the major barrier, especially when we need to build a TTS system in a new language in which speech and linguistic resources are lacking. Even for the major languages such as English, we encounter a similar problem when we scale up the size of the database or change to a new domain.

Recently, the distributed representation of raw word inputs, called "word embedding", have been used in various natural language processing (NLP) tasks with success [2]. Typically, this low-dimension continuous vector representation of words can be learned from raw word inputs. It has been reported that a well trained embedding vector is able to encode syntactic and semantic information [3]. Therefore, it is interesting to investigate whether such the word embedding vectors can be used as the additional or alternative linguistic representations in front-end modules of TTS systems. In [4], various types of the word embedding vectors, such as those learned by the recurrent neural network (RNN)-based language model (LM) [3] and log-linear model with negative sampling [5], have been utilized for an RNN-based TTS systems. The objective and subjective evaluation showed that the RNN-based TTS system with the word embedding vectors performs marginally worse than that with correct POS and prosodic tags but clearly performs better than a system with neither the POS nor prosodic tags. In [6], the word embedding vectors and triphone embedding vectors were used for text-to-articulatory prediction. Their experiments have also confirmed the same trend as the above TTS experiments.

We further investigated the use of the distributed representation of input text for the neural-network-based TTS systems in two new directions. First, because the word embedding approaches may provide effective representations not only for word units but also at other linguistic levels, we attempted to use phoneme, syllable, and phrase embedding vectors in addition to the standard word embedding vectors. Second, we examined the impacts of several different normalization approaches on the word embedded vectors before they are fed into neural-network-based acoustic models. We showed that the normalization of the obtained word vectors may alter the distance between units in the original embedded space, thus, degrading the systems' performance.

[1] National Institute of Informatics, 2-1-2, Hitotsubashi-cho, Chiyoda-ku, Tokyo, 101-8430, Japan
[a)] wangxin@nii.ac.jp
[b)] takaki@nii.ac.jp
[c)] jyamagis@nii.ac.jp

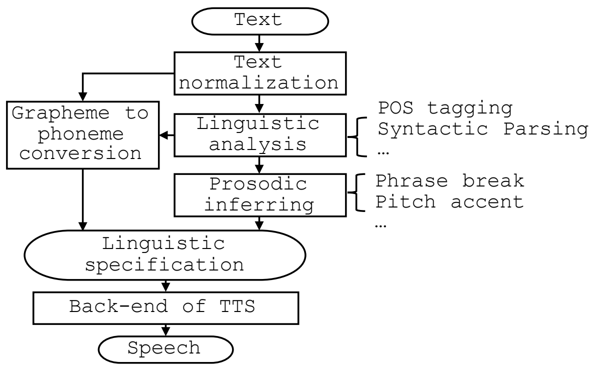

Fig. 1: Simplified diagram of conventional front-end processing flow used for typical TTS systems

The rest of this paper is structured as follows: Section 2 briefly introduces the conventional front-end modules in English TTS systems. Section 3 then introduces the neural-network-based acoustic model, wherein the embedded vectors replaces the conventional prosodic context. Section 4 describes the proposed embedded vectors for syllable, phoneme and phrases. The analysis of the effects of normalization of the embedded vectors is also presented. Section 5 explains the experiments and results, and Section 6 summarizes this work and describes the future work.

2. Conventional front-end processing flow for typical English TTS systems

To convert a text into natural speech, a TTS system must retrieve the pronunciation of words in the text and infer the prosody for the text. However, the association between pronunciation symbols and the word tokens is ambiguous in English. Also, even if the pronunciation of words is known, the prosodic information can not be easily acquired because it is not encoded explicitly in the normal text string [7].

Considering the above challenge, typical English TTS systems deploy the front-end and back-end architecture shown in Fig.1. The front-end of a TTS system infers the symbolic representation of both segmental and prosodic properties of speech. Then, the back-end acoustic model converts the symbolic intermediate representation into a speech waveform, typically using the unit-selection method [8] or the statistical parametric method [9].

Between the front- and back-ends, the intermediate representation encodes both segmental and suprasegmental aspects of speech. The segmental part mainly includes the phoneme sequence of every word in the input text. With a carefully produced pronunciation lexicon and well-tuned G2P algorithms, this segmental information can be inferred with high accuracy [10][11].

However, the prosody, e.g., the intonation, timing and stress of the utterance, is more difficult to predict. First, no consensus has been reached on designing the set of prosodic symbols for English. A widely adopted set is the Tone and Break Indices (ToBI) [12] wherein pitch accent and break index represent the local pitch excursion and association between adjacent words, respectively. With the target prosodic symbols defined, the next problem is to predict these targets for the input text. Typically, the front-end of TTS first usually infers linguistic features of the input text [13]. For example, as shown in Fig.1, a Part-of-speech (POS) tagger and a syntactic parser may be used at this stage in order to derive the sequence of POS tags and the syntactic tree of the input text. Given the inferred POS and syntactic structure, prosodic targets can be predicted [13].

For the task of linguistic analysis and prosodic modelling in the front-end, researchers have proposed several effective methods. For example, Taylor used the hidden Markov model (HMM) to infer the phoneme sequence for each word in the input text [10]; Kupiec used HMM for POS tagging [14] ; Various syntactic parsers also utilized the statistical approach [15]. On the prosodic modelling part, Hirchberg utilized the decision tree to predict the pitch accent based on syntactic features of the text [13]. To construct these statistical modules, expert knowledge on specific topics such as syntax is required to design task-related input and output features. Also, a specific data corpus must be prepared to train each modules. For example, prosodic models are usually trained on the Boston University News Radio Corpus [16] and syntactic parsers are usually trained using the Penn Treebank corpus [17]. Based on expert knowledge in feature designing and data annotation, the front-end of a TTS system can exhibit good performance.

3. Using word embedding for neural-network-based acoustic models

3.1 Shortcomings of conventional front-end framework

The conventional front-end framework is not ideal for TTS, especially on prosody modelling. First, it assumes that discrete prosodic symbols must be defined. However, researchers have not yet reached a consensus on the best definition of discrete prosodic form [1][18]. Even if a consensus is reached, another dispute is whether symbolic prosody is necessary for a speech task. After all, the acoustic feature space is continuous. This inconsistency may result in quantization noise during prosody annotation and acoustic realization given the prosodic symbols [19][20].

Additionally, as mentioned above, prosodic models based on a supervised machine learning method require sufficient training data with consistent annotation. However, consistency of prosodic annotation across annotators may not be as high as expected [21]. Inconsistency in the training data may affect the performance of the prosodic models. The errors predicted by these models may be propagated to the following acoustic model and eventually degrade the quality of the synthetic voice.

3.2 Word embedding

Although the notion of the prosodic form is beneficial, the prosodic form can be defined and modelled implicitly [19]. For example, we may directly feed the one-hot vectors of words to the acoustic model and then rely on the model to infer prosodic information during the training process. In this way, the cost of defining prosodic symbols and preparing various prosodic modules can be decreased. However, this approach is impractical because the one-hot vector has huge dimension (e.g. 50k dimensions for Penn TreeBank corpus). More significantly, these one-hot vectors can not encode syntactic and semantic information that should be useful for inferring prosody.

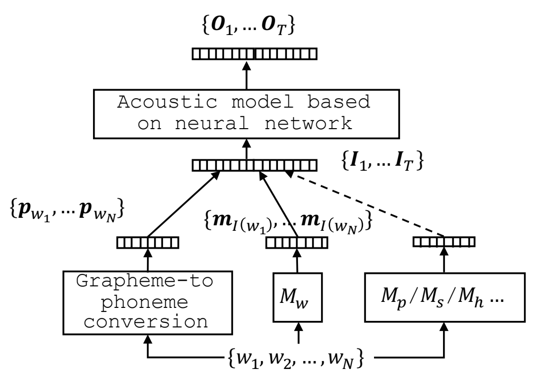
$\{O_1, \dots O_T\}$
Acoustic model based on neural network
$\{I_1, \dots I_T\}$
$\{p_{w_1}, \cdots p_{w_N}\}$
$\{m_{I(w_1)}, \cdots m_{I(w_N)}\}$
Grapheme-to phoneme conversion
M_w
$M_p/M_s/M_h \dots$
$\{w_1, w_2, \dots, w_N\}$

Fig. 2: Neural-network-based TTS with word embedding. The sequence of phonemic context $\boldsymbol{p}$ and embedded vectors $\boldsymbol{m}$ are converted into acoustic feature vectors $\boldsymbol{O}$. $\boldsymbol{M}_*$ denotes the embedded vectors for words or other linguistic units of the input text $\{w_1, \cdots, w_N\}$. Duration prediction to convert $\boldsymbol{p}$ and $\boldsymbol{m}$ sequences to $\boldsymbol{I}$ sequence is not shown here.

Recently, distributed representation of words based on word embedding technique became popular. This method derives a continuous low-dimensional vector representation for words. It was found that, with a specific training scheme, the learned embedded vectors can encode the syntactic and semantic relationship between words. For example, Mikolov utilized an RNN based language model to derive the embedded vectors [3]. The results showed that a syntactic analogy, such as "year to years is as law to laws", can be derived through calculating the cosine distance between word vectors. Besides RNN-based LM, simple log linear models, such as continuous bags of words (CBOW) and skip-gram, have been used to derive the word embedded vectors.

Since such word embedded vectors have much less dimensions than the size of the vocabulary, they can be used as the input to the acoustic model. If embedded vectors indeed encode the syntactic information of words, the acoustic model may infer the prosodic regularity hopefully from the embedded vectors without additional a linguistic analyzer and prosodic model.

3.3 Acoustic modelling based on recurrent neural network

Inferring prosodic parameters implicitly requires a powerful acoustic model. For this study, we used the deep bidirectional RNN-based on long short term memory (LSTM) units [22]. The basic RNN is based on the normal feed-forward neural network plus the hidden state of the previous time step as the input to the current node of the current time step, expecting that dependency of the training data over the time span can be modelled. However, due to the gradient vanishing problem, a vanilla RNN may not be able to capture the dependency over a long time span. As a solution to this problem, LSTM has been proposed to replace the simple non-linear activation function in the hidden node of a vanilla RNN. Particularly, an LSTM unit uses three gates to control the input, output information flow and the state of the memory cell. This LSTM cell can also be incorporated in a bi-directional RNN, which results in bi-directional LSTM (DBLSTM) RNN [22].

For acoustic modelling, the text of an utterance is converted into a sequence of frames $\{\boldsymbol{I}_1, \boldsymbol{I}_2, \cdots, \boldsymbol{I}_t, \cdots, \boldsymbol{I}_T\}$, wherein T is the total number of frames of the utterance to be synthesized. The linguistic vector $\boldsymbol{I}_t$ at time t consists of the embedded vector of a word at time t and the phonemic context of that word. Together with a sequence of acoustic feature vectors $\{\boldsymbol{O}_1, \cdots, \boldsymbol{O}_T\}$, the acoustic model can be trained. Note that, in this paper, the phonemic identity is kept. Only prosodic contexts are replaced by embedding vectors, which are described in the next section.

4. Proposed methods

4.1 Motivation for using embedded vector of various linguistic units

In [4], embedded vectors of words were utilised as the input to the acoustic model instead of the ToBI symbols and POS tags. However, the word is not the only linguistic unit that can be represented in the embedded space. An utterance can be hierarchically decomposed as phrase, word, syllable or phoneme sequences. As Bian et al. argues, the base of embedded vectors can be a sub-word unit such as a prefix, suffix, or syllable [23]. It can also be the whole sentence or document (known as doc2vec) [24].

For speech application, both sub-word and supra-word vectors may be useful. The sub-word vectors may be expected to encode the segmental information of speech. For example, it has been shown that vowel and consonants can be differentiated in a two-dimensional space derived by latent semantic indexing (LSI) [25]. Even though the phonemes can be efficiently represented using one-hot vectors, it would be interesting to explore whether continuous representation can be beneficial.

Another sub-word linguistic unit is syllable. Bian et al. has used vectors of syllable as the input feature for an NLP task [23]. Although the performance is not improved for the NLP task, it may be useful for speech-related tasks. Thus, it would be interesting to explore whether embedded representation of those linguistic units can improve the performance of the acoustic model.

As mentioned above, prosody of speech is mainly suprasegmental. A single word can be realized with different prosody in different contexts, which indicates that the word or sub-word level vectors may be insufficient to encode all the prosodic information. Therefore embedding, vectors of phrases or sentences may be used. There has been a similar attempt for sentimental analysis and [24] has utilised sentence-level vectors to predict the sentiment of a sentence. Hence it is also interesting for us to verify whether sentence-level vectors could benefit the acoustic model in prosodic realization.

4.2 Learning the embedded vectors for various linguistic units

To derive the embedded vectors for syllable and phoneme, we used the Continuous-Bag-of-Word (CBOW) model [5]. For CBOW, the input is a set of one-hot vectors corresponding to each word in the context $\boldsymbol{c} = \{w_{i-n}, \cdots, w_{i+n}\}$, as shown on the left side of Fig.3. The input projection layer maps the one-hot vector of context word w into $\boldsymbol{m}_{I(w)}$. Because the input vector is one-hot, the projected $\boldsymbol{m}_{I(w)}$ actually corresponds to the $I(w)$-th row of the projection matrix $\boldsymbol{M}$, where $I(w)$ is the index of w. Then, the hidden representation $\boldsymbol{h}$ is calculated as the average of $\boldsymbol{v}$:

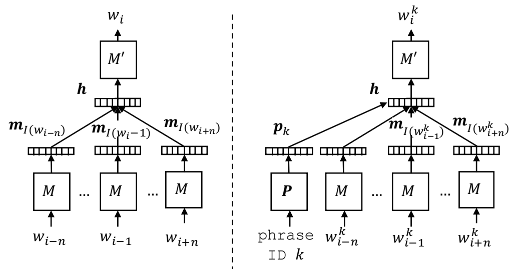
w_i
M'
$\boldsymbol{h}$
$\boldsymbol{m}_{I(w_{i-n})}$
$\boldsymbol{m}_{I(w_{i}-1)}$
$\boldsymbol{m}_{I(w_{i+n})}$
M
w_{i-n}
w_{i-1}
w_{i+n}
w_i^k
$\boldsymbol{p}_k$
$\boldsymbol{m}_{I(w_{i-1}^k)}$
$\boldsymbol{m}_{I(w_{i+n}^k)}$
$\boldsymbol{P}$
phrase ID k
w_{i-n}^k
w_{i-1}^k
w_{i+n}^k

Fig. 3: CBOW (left) and Doc2Vec model (right) models

$$\boldsymbol{h} = \frac{1}{|\boldsymbol{c}|}\sum_{i=1}^{|\boldsymbol{c}|} \boldsymbol{m}_{I(w_i)} \tag{1}$$

where $|\boldsymbol{c}|$ is the size of the context, in this case $|\boldsymbol{c}| = 2n + 1$.

This $\boldsymbol{h}$ will be further transformed by another projection matrix $\boldsymbol{M'}$ into $\boldsymbol{u} = \boldsymbol{M'h}$. The dimension of $\boldsymbol{u}$ is the same as that of the input one-hot vector. Then, based on the softmax function, the 'probability' to generate the word w_i can be written as

$$p(w_i|\boldsymbol{c}; \boldsymbol{M}, \boldsymbol{M'}) = \frac{exp(\boldsymbol{m}'_{I(w_i)}\boldsymbol{h})}{\sum_j exp(\boldsymbol{m}'_j\boldsymbol{h})} \tag{2}$$

where $\boldsymbol{m}'_j$ is the j-th row of $\boldsymbol{M'}$ and $I(w_i)$ is the index of w_i. The projection matrix $\boldsymbol{\theta} = \{\boldsymbol{M}, \boldsymbol{M'}\}$ can be learned through the maximum likelihood (ML) criterion. Each row of the learned $\boldsymbol{M}$ corresponds to the embedded vector of one word. If we replace the w as the token of syllable or phoneme, the same CBOW model can be used to derive the embedded vectors for syllable or phoneme.

For the embedded vector of phrase, we used the doc2vec model. This structure is similar to the CBOW model except for the paragraph matrix $\boldsymbol{P}$. This structure is shown on the right side of Fig.3. At the training stage, the input to the model includes a vector $\boldsymbol{p}_k$ for the current phrase where k is the ID of this phrase. Following the same training procedure as CBOW, $\boldsymbol{P}$ can be updated. Note that updating each row of $\boldsymbol{p}_k$ depends on the errors propagated backwards for all the words w_1^k, w_N^k in the k-th phrase. At the test stage, because a test phrase is most of time different from the training phrases, the test phrase is unseen and can not be retrieved directly from $\boldsymbol{P}$. Instead, the paragraph embedding $\boldsymbol{p}_{k'}$ for unseen phrase k' must be inferred given the words in k'. This can be achieved using the back-propagation algorithm with all the other parameters of the Doc2Vec model fixed.

4.3 Normalization method of the embedded vectors

For training the neural network model, normalization of the input and output features is necessary. However, care should be take when the embedded vectors are normalized. As Mikolov showed, the distance between embedded vectors of words w_1 and w_2 is [3]

$$cos(\boldsymbol{m}_{I(w_1)}, \boldsymbol{m}_{I(w_2)}) = \frac{\boldsymbol{m}_{I(w_1)}^{\top}\boldsymbol{m}_{I(w_2)}}{||\boldsymbol{m}_{I(w_1)}|| \cdot ||\boldsymbol{m}_{I(w_2)}||} \tag{3}$$

If we normalize the vector as $\hat{m}_k = \frac{m_k - \mu_k}{\sigma_k}$, wherein μ_k and σ_k are the $k-$th dimension of the mean and variance vector, respectively, the distance $cos(\hat{\boldsymbol{m}}_{I(w_1)}, \hat{\boldsymbol{m}}_{I(w_2)})$ between the normalised vectors will be unequal to the original distance.

To verify the above thought, we conducted a syntactic test [3] using the word embedding derived using an RNN language model. The original embedded vectors exhibited an accuracy of 16.2%, which is identical to that reported. However, after we normalized the vectors using the global mean and variance vector, the accuracy dropped from 16.2% to 10.68%.

Directly using the raw embedded vectors without normalization is possible. However, the vectors $\boldsymbol{m}$ have unit a length, and the value of each dimension may be too small. Thus, another strategy is to scale the dimension of every embedded vector as $m_k = \frac{m_k}{\sigma_k}$. With this scale, the accuracy on the syntactic test increases from 10.68% to 12.75%. Part of the information encoded in the scaled vectors may be lost. However, it is interesting to know whether the simply scaled vectors can be more useful.

5. Experiments and Results

5.1 Preparing the embedded vectors of linguistic units

Experiments were conducted for the English TTS task. Embedded vectors of word, phoneme, syllable and phrase were involved in the experiments. For the vectors of words, we directly used the vectors trained based on the RNN language model [*1]. The same set of word vectors was also used in [4]. This vector data set covers most of the words in the speech corpus used for the following experiments; only 358 out of 340,000 word tokens were not found. The vectors of these words were simply set as the global mean of all the word vectors.

The embedded vectors at the phoneme and syllable level were derived using the Word2Vec tool [5]. The training data were the English text in the news domain [*2]. At first, text was normalised [*3]. Then, Flite [26] was used to convert the text into the sequences of syllables and phonemes. After that, CBOW models for syllables and phonemes were trained separately using the Word2Vec tool. The training process was iterated for 15 times with negative sampling [5] and finally yielded 200 dimensional embedded vectors of syllable and phoneme.

The phrase level vectors were learned using the distributed memory model of paragraph vectors (PV-DM) [24], which is shown on the right of Fig.3. The same news data corpus for syllable and phoneme vectors were used for model training. Phrases were first extracted from the corpus according to the punctuation in the utterance. Then, the PV-DM model was trained with negative sampling for 15 iterations. Given the PV-DM model, phrase vectors for all the grammatical phrases in the speech data corpus were inferred based on the back-propagation algorithm. The dimensions of the word and phrase vectors were 100 and 100, respectively. For each word in the speech data corpus, the phrase and word vectors learned using the PV-DM model were concatenated into a vector of 200 dimensions.

5.2 Experimental setup for the TTS task

The database for acoustic model training contains 12072 English utterances from a female speaker. 500 utterances were ran-

[*1] http://rnnlm.org
[*2] http://www.statmt.org/wmt14/training-monolingual-news-crawl/news.2013.en.shuffled.gz
[*3] http://word2vec.googlecode.com/svn/trunk/demo-train-big-model-v1.sh

Table 1: Additional features besides quinphones as input to the acoustic model. Feature ID was used as the subscript to identify experimental systems. For example, R_e denotes the RNN-based system using phoneme vectors and quinphones as input.

ID	description	dimension
N	none	-
p	traditional prosodic context	90
e	embedded vectors of phoneme	200
s	embedded vectors of syllable	200
w	embedded vectors of word	80
h	embedded vectors of phrase	200

domly chosen as the test set. Given the transcription of each utterance, its phoneme sequence was acquired using Flite. Meanwhile, Mel-generalized cepstral coefficients (MGC) of order 60, a 1 dimensional continuous F0 trajectory, the voiced/unvoiced condition, and band aperiodicity of order 25 were extracted for each speech frame. Although an RNN-based acoustic model was assumed to be able to model the inter-frame dependency of consecutive frames, we still used the delta and delta-delta components of the acoustic features except the voiced/unvoiced condition. Thus, the number of dimension of the acoustic feature per frame was $(60 + 25 + 1) \times 3 + 1 = 259$.

In this rest of the paper, experiments are introduced in a chronological order. First, we compared the performance of the DBLSTM-RNN-based acoustic model with different embedded vectors as input features. We use R to denote these RNN-based systems, and use subscripts shown in Tab.1 to identify those with specific input features. Note that, all experimental systems uses the quinphone as input features. The prosodic context in Tab.1 includes binary (e.g. whether the current syllable is pitch accented) and positional information (e.g. distance to the next pitch accent) derived using Flite.

The model structure for all RNN systems contains two normal feed-forward layers with a sigmoid activation function and two bi-directional LSTM layers. Except the first feed-forward layer, the number of hidden nodes of the following layers was fixed at 512, 256, 256, respectively. For systems with a the combination of embedded vectors as input (e.g. R_{es}), the size of the first hidden layer was 1024. Otherwise, it was 512.

5.3 Performance of DBLSTM-RNN based systems with different input features

For this initial work, we just adopted three types of objective measures to show the performance of each system: RMSE of the predicted MGC coefficients and RMSE and correlation coefficients of F0 trajectory. To yield meaningful results, we took the average of the objective measures over the last five training epochs of each system. The voiced/unvoiced error rate is not shown because the difference across systems was trivial.

As Fig.4 and Fig.5 show, R_p and R_{pw}, as the systems with the conventional prosodic context, performed the best. Compared with these two systems, the inferior performance of R_N was expected. However, when the word vectors were used as input features, R_w was only slightly better than R_N on F0 modelling. This result was unexpected since it was reported that the word vectors could improve a system without prosodic context [4]. Additional

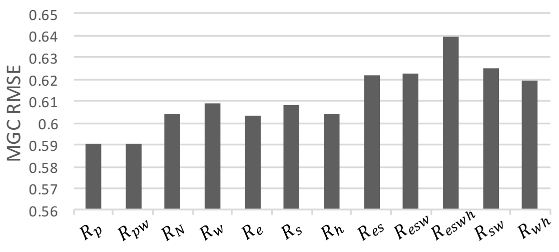

Fig. 4: RMSE of MGC when DBLSTM-RNN was trained using different input features

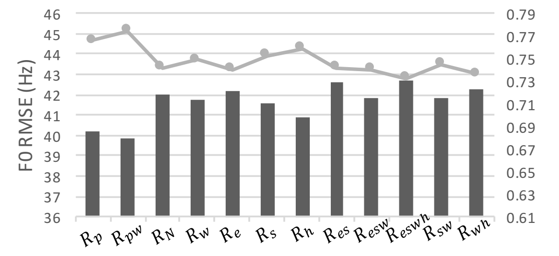

Fig. 5: RMSE (Hz) and correlation coefficients of F0 when DBLSTM-RNN was trained using different input features

tuning of the hyper-parameters may be required to take advantage of the word vectors.

Embedded vectors of other linguistic units showed different effects on the results. Particularly, R_h performed the best on F0 modeling among the systems without prosodic context. This result is interesting because it indicates that a phrase vector may encodes information related to the suprasegmental property of speech. Unfortunately, when combining vectors of different levels, the results only degraded.

The overall performance of the systems with embedded vectors was not positive. At first, we wondered whether the power of the DBLSTM-RNN reduced the effect of the input features so that input features were not important (e.g. even the difference between R_p and R_N was less than 2Hz in F0 modelling). Thus, we prepared another five systems using feed-forward neural network (D) to replace DBLSTM-RNN. The network structure was similar, except the LSTM layers were replaced with feed-forward layers with 512 nodes. The results are listed in Tab.2. The first observation is that the RNN-based system without any prosodic context (R_N) performs better than the DNN-based system with prosodic context (D_p). Another observation is that the difference between DNN systems using or not using prosodic context was much larger than that between RNN systems. By comparing D_N, D_w, and D_h, we observed that the differences between D_N and the other two systems were larger than the DBLSTM-RNN case. This result suggests that that the word and phrase vectors encode useful information. However, their effectiveness may be insignificant when a powerful model is used.

5.4 Effect of normalization on embedded vectors

Another perspective to interpret the unsatisfactory results of the previous section is to re-examine the input features. According to Sec.4.3, additional systems were trained based on embedded vector of words using different pre-processing methods. The

Table 2: Performance of acoustic modeling using DNN (D) and DBLSTM-RNN (R) based on different input features listed in Tab.1. Superscript ^{s} means the feature vector was scaled by the variance vector. Superscript ^{r} means the raw feature vector was used without pre-processing.

Input	MGC RMSE		F0 RMSE(Hz)		F0 Corr.	
feature	D	R	D	R	D	R
p	0.634	0.590	45.71	40.17	0.692	0.766
$p+w$	0.638	0.590	45.30	39.85	0.694	0.774
N	1.009	0.604	53.82	41.98	0.518	0.742
w	1.011	0.609	53.31	41.71	0.533	0.749
w^s	1.004	0.598	53.17	41.26	0.535	0.751
w^r	1.004	0.606	53.59	41.70	0.523	0.750
h	1.006	0.604	53.24	41.00	0.536	0.759
h^s	1.008	0.622	53.36	41.82	0.532	0.744

results are listed in Tab.2. For D_w^s and R_w^s, the embedded vectors were scaled without subtracting the mean vector; For D_w^r and R_w^r, the embedded vectors were directly feed into the model. As shown in Tab.2, the comparison among R_w, R_w^s, R_w^r suggests that the scaling strategy results in the better performance. This was also observed among DNN-based systems.

Additional experiments were conducted for systems with phrase vectors h as input. However, as the comparison among R_h and R_h^s shows, the scaling strategy can not surpass the normal normalization method. However, further investigation is required before claiming that the simple scaling strategy is not useful. First, we must answer whether similarity between phrases can be measured in the same way as word vectors? If the answer is no, normalizing the phrase vectors may not distort the similarity in the phrase vector space. As far as we know, this is still an open question.

6. Conclusion

We investigated the embedded vectors of various linguistic units to replace the conventional linguistic context as input features to an acoustic model. The results suggested that the phrase vectors can be beneficial in acoustic modelling. However, the overall improvement is relatively small. For the word vectors, if the simple scaling strategy is used instead of the normal normalization method, their benefits could be better used by the neural network based acoustic model. Subjective evaluation on typical systems such as R_h will be conducted in the future.

Overall, further investigation is required on embedded vectors for TTS. Typically, all embedded vectors are learned based on the "meaning by collocation" assumption. It is doubtful whether this assumption is valid for speech-related tasks. In fact, many researchers recently argued that "meaning by collocation" is not ideal. Better embedded vectors may still require domain-specific knowledge [27][28]. In the future, embedded vectors may be tuned with speech-related tasks so that sufficient acoustic information can be encoded in the embedded space. Towards the goal of end-to-end TTS framework, an interesting idea is to adopt the Connectionist Temporal Classification (CTC) method [29] to directly map the text sequence into speech waveform.

References

[1] Taylor, P.: *Text-to-Speech Synthesis*, Cambridge University Press (2009).

[2] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P.: Natural language processing (almost) from scratch, *The Journal of Machine Learning Research*, Vol. 12, pp. 2493–2537 (2011).

[3] Mikolov, T., Yih, W.-t. and Zweig, G.: Linguistic regularities in continuous space word representations., *HLT-NAACL*, pp. 746–751 (2013).

[4] Wang, P., Qian, Y., Soong, F. K., He, L. and Zhao, H.: Word embedding for recurrent neural network based tts synthesis, *ICASSP-2015*, IEEE, pp. 4879–4883 (2015).

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J.: Distributed representations of words and phrases and their compositionality, *NIPS-2013*, pp. 3111–3119 (2013).

[6] Zhu, P., Xie, L. and Chen, Y.: Articulatory movement prediction using deep bidirectional long short-term memory based recurrent neural networks and word/phone embeddings, *INTERSPEECH-2015* (2015).

[7] Halliday, M. A. K.: *An Introduction to Functional Grammar*, London, UK: Edward Arnold, 2nd ed edition (1994).

[8] Hunt, A. J. and Black, A. W.: Unit selection in a concatenative speech synthesis system using a large speech database, *ICASSP-1996*, IEEE, pp. 373–376 (1996).

[9] Tokuda, K., Nankaku, Y., Toda, T., Zen, H., Yamagishi, J. and Oura, K.: Speech synthesis based on hidden Markov models, *Proceedings of the IEEE*, Vol. 101, No. 5, pp. 1234–1252 (2013).

[10] Taylor, P.: Hidden Markov models for grapheme to phoneme conversion, *INTERSPEECH-2005*, pp. 1973–1976 (2005).

[11] Black, A. W., Lenzo, K. and Pagel, V.: Issues in building general letter to sound rules, *SSW3-1998* (1998).

[12] Silverman, K. E. A., Beckman, M. E., Pitrelli, J. F., Ostendorf, M., Wightman, C. W., Price, P., Pierrehumbert, J. B. and Hirschberg, J.: TOBI: a standard for labeling English prosody, *ICSLP-1992*, pp. 867–870 (1992).

[13] Hirschberg, J.: Pitch accent in context predicting intonational prominence from text, *Artificial Intelligence*, Vol. 63, No. 1, pp. 305–340 (1993).

[14] Kupiec, J.: Robust part-of-speech tagging using a hidden Markov model, *Computer Speech & Language*, Vol. 6, No. 3, pp. 225–242 (1992).

[15] Collins, M.: Head-driven statistical models for natural language parsing, *Computational linguistics*, Vol. 29, No. 4, pp. 589–637 (2003).

[16] Ostendorf, M., Price, P. J. and Shattuck-Hufnagel, S.: The Boston University radio news corpus, *Linguistic Data Consortium* (1995).

[17] Marcus, M. P., Marcinkiewicz, M. A. and Santorini, B.: Building a large annotated corpus of English: The Penn Treebank, *Computational linguistics*, Vol. 19, No. 2, pp. 313–330 (1993).

[18] Hirst, D.: The phonology and phonetics of speech prosody: between acoustics and interpretation, *Speech Prosody* (2004).

[19] Shriberg, E. and Stolcke, A.: Prosody modeling for automatic speech recognition and understanding, *Mathematical Foundations of Speech and Language Processing*, Springer, pp. 105–114 (2004).

[20] Batliner, A. and Möbius, B.: Prosodic models, automatic speech understanding, and speech synthesis: Towards the common ground?, *The integration of phonetic knowledge in speech technology*, Springer, pp. 21–44 (2005).

[21] Wightman, C. W.: ToBI or not ToBI?, *Speech Prosody* (2002).

[22] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. and Schmidhuber, J.: LSTM: A search space odyssey, (online), available from ⟨http://arxiv.org/abs/1503.04069⟩ (2015).

[23] Bian, J., Gao, B. and Liu, T.-Y.: Knowledge-powered deep learning for word embedding, *Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 132–148 (2014).

[24] Le, Q. and Mikolov, T.: Distributed representations of sentences and documents, *ICML-14*, pp. 1188–1196 (2014).

[25] Watts, O. S.: Unsupervised learning for text-to-speech synthesis, PhD Thesis, University of Edinburgh (2013).

[26] HTS Working Group: The English TTS System "Flite+hts_engine" (2014).

[27] Rubinstein, D., Levi, E., Schwartz, R. and Rappoport, A.: How well do distributional models capture different types of semantic knowledge?, *Proceedings of ACL*, Vol. 2, pp. 726–730 (2015).

[28] Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X. and Liu, T.-Y.: RC-NET: A general framework for incorporating knowledge into word Representations, *CIKM-14*, pp. 1219–1228 (2014).

[29] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks, *ICML-2006*, pp. 369–376 (2006).