applied sciences

MDPI

Article

Acoustic Word Embeddings for End-to-End Speech Synthesis

Feiyu Shen , Chenpeng Du and Kai Yu *

MoE Key Lab of Artificial Intelligence, X-LANCE Lab, Department of Computer Science and Engineering AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China; francis_sfy@sjtu.edu.cn (F.S.); duchenpeng@sjtu.edu.cn (C.D.)
* Correspondence: kai.yu@sjtu.edu.cn

Abstract: The most recent end-to-end speech synthesis systems use phonemes as acoustic input tokens and ignore the information about which word the phonemes come from. However, many words have their specific prosody type, which may significantly affect the naturalness. Prior works have employed pre-trained linguistic word embeddings as TTS system input. However, since linguistic information is not directly relevant to how words are pronounced, TTS quality improvement of these systems is mild. In this paper, we propose a novel and effective way of jointly training acoustic phone and word embeddings for end-to-end TTS systems. Experiments on the LJSpeech dataset show that the acoustic word embeddings dramatically decrease both the training and validation loss in phone-level prosody prediction. Subjective evaluations on naturalness demonstrate that the incorporation of acoustic word embeddings can significantly outperform both pure phone-based system and the TTS system with pre-trained linguistic word embedding.

Keywords: speech synthesis; acoustic input tokens; naturalness; word embedding

Citation: Shen, F.; Du, C.; Yu, K. Acoustic Word Embeddings for End-to-End Speech Synthesis. *Appl. Sci.* **2021**, *11*, 9010. https://doi.org/10.3390/app11199010

Academic Editor: José A. González-López

Received: 15 August 2021
Accepted: 22 September 2021
Published: 27 September 2021

1. Introduction

Recently, end-to-end text-to-speech (TTS) synthesis models with sequence-to-sequence architectures [1–3] have achieved great success in generating naturally sounding speech. To avoid regressive frame-by-frame generation, non-autoregressive TTS models, such as FastSpeech [4] and FastSpeech2 [5], are proposed for fast generation speed. Most of the above end-to-end TTS systems use only phonemes as input tokens and ignore the information about which word the phonemes come from.

However, word identities are important for the TTS system to generate highly natural speech. In human communication, we know the words that we are speaking, which is crucial to pronounce the sentences properly. Our motivation is based on the fact that many words have their specific prosody type, which significantly affects the naturalness. For example, when we read "It is so big", we probably emphasize the word "so", but not the word "is". Similarly, "so" is often the emphasized word in many cases. We want the model to remember this kind of information for each word. We use the term "prosody" here to refer to the extra information used in synthesis in addition to "phoneme", "duration", "pitch", etc.

In this paper, we propose a novel and effective approach that directly trains the acoustic word embeddings in the TTS system. Similar to the phoneme embeddings that contain the information about how the phonemes are pronounced, our acoustic word embeddings directly indicate how the words are pronounced. Both the phoneme and word sequence are utilized as input to the TTS system and passed through two encoders separately. The two output hidden states are then concatenated for prosody prediction and speech synthesis.

Several key factors are investigated in this work. First, we compare three different model architectures for the word encoder and find that using a convolution layer followed by Transformer layers achieves the best performance. Second, we determine the influence

of the word frequency threshold. Finally, we carry out subjective evaluations in terms of naturalness, which demonstrate that our proposed system is not only better than the system that uses only phoneme sequence as input, but also better than the prior works that simply add linguistic word embeddings from pretrained GloVe or BERT.

2. Related Work

In traditional HMM/DNN-based statistical parametric speech synthesis, the model takes full-context features as input [6]. It is obtained from text analysis and contains various linguistic contextual information in addition to phoneme identities. Ref. [7] trains a word language model to obtain linguistic word embeddings and then uses them as part of the input features. In the end-to-end architectures, there are also several prior works trying to solve the problem by adding pretrained linguistic word embeddings in TTS. Ref. [8] obtains word embeddings from a pretrained Chinese to English word translation model and then takes both the phoneme sequence and word sequence into consideration in TTS.

Recently, the large pretrained language model BERT [9] exhibits an impressive performance on many natural language processing (NLP) tasks, so it is also introduced to TTS [10–12]. Refs. [10,11] extract hidden states of BERT and pass them to the TTS model. Ref. [12] tries to fine-tune the BERT parameters with a prosody prediction task but still freezes the word piece embeddings. All these works report that they have achieved some gains in naturalness.

However, the word embeddings in all the prior works were obtained from pretrained NLP models and thus contain only linguistic information, which is not directly relevant to how the words are pronounced. Therefore, improvement in TTS is often very limited. In this paper, we train acoustic word embeddings directly with the TTS model instead of using linguistic word embeddings.

3. End-to-End Speech Synthesis

3.1. FastSpeech2

Recently, non-autoregressive TTS models such as FastSpeech2 [5] have become popular due to their fast inference speed and stability. Compared with the original FastSpeech [4], FastSpeech2 is optimized to minimize the mean square error (MSE) $\mathcal{L}_{\texttt{MEL}}$ between the predicted and the ground-truth mel-spectrograms, instead of applying a teacher–student training. Moreover, the duration target is not extracted from the attention map of an autoregressive teacher model but the forced alignment of speech and text. Additionally, ref. [5] condition the prediction of mel-spectrogram on the variance information such as pitch and energy with a variance adaptor. The adaptor is trained to predict the variance information with an MSE loss $\mathcal{L}_{\texttt{VAR}}$.

In this work, we use FastSpeech2 as our acoustic model. It contains a phoneme encoder that transforms the input phoneme sequence $\mathbf{p}$ to a hidden state sequence $\mathbf{h}$, that is

$$\mathbf{h} = Encode(\mathbf{p}) \tag{1}$$

The hidden state sequence $\mathbf{h}$ is then passed to a variance adaptor and a decoder which outputs the mel-spectrogram, that is

$$\mathbf{y} = Decode(Adapt(\mathbf{h})) \tag{2}$$

3.2. Objective Evaluation of Phone-Level Prosody Prediction

In the standard FastSpeech2 system, prosody modeling is not explicitly considered, which makes it hard to objectively evaluate the prosody prediction performance of the TTS systems without a subjective listening test. Accordingly, in this work, we introduce a phone-level prosody prediction module [13] to our models, which autoregressively predicts the distribution of prosody embeddings for each phoneme. It can not only lead to improved naturalness compared to the standard FastSpeech2 system, but also allow the use

of prosody embeddings log-likelihood to easily evaluate phone-level prosody prediction performance in an objective way. The structure is illustrated in Figure 1.

Specifically, in the training stage, the prosody embeddings

$$\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K] \tag{3}$$

are extracted for all the K phonemes by a prosody extractor from the corresponding mel-spectrogram segment. It is then projected and added to the corresponding hidden state sequence $\mathbf{h}$ in order to better reconstruct the mel-spectrogram. We use $\mathbf{e}_k$ to represent the prosody embeddings for the k-th phoneme. In this work, the distribution of $\mathbf{e}_k$ is assumed to be a GMM whose parameters are predicted by the prosody predictor. In the inference stage, we sample the $\hat{\mathbf{e}}_k$ from the predicted prosody distribution for each phoneme.

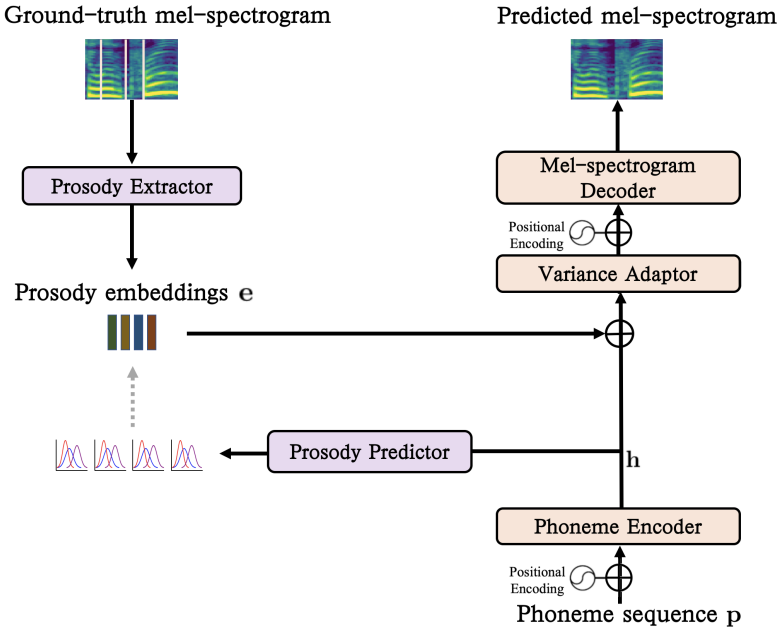

Figure 1. End-to-end speech synthesis with GMM-based prosody modeling.

The training criterion for the prosody prediction is the negative log-likelihood of the prosody embeddings $\mathbf{e}$, so we obtain the loss function for training the prosody predictor

$$\mathcal{L}_{\mathtt{PP}} = \sum_{k=1}^{K} -\log p(\mathbf{e}_k; \mathbf{e}_{<k}, \mathbf{h}) \tag{4}$$

Consequently, the overall architecture is optimized with the loss function

$$\mathcal{L} = \beta \cdot \mathcal{L}_{\mathtt{PP}} + \mathcal{L}_{\mathtt{FastSpeech2}} \tag{5}$$

where $\mathcal{L}_{\mathtt{FastSpeech2}}$ is the loss function of FastSpeech2, which is the sum of variance prediction loss and mel-spectrogram reconstruction loss as described in [5], and β is the relative weight between the two terms.

4. Acoustic Word Embeddings

Most of the recent popular TTS systems use phonemes as acoustic input tokens and ignore the information about which word the phonemes come from. However, word identities are important for the TTS system to generate highly natural speech. In human communication, we know the words we speak, which is crucial to pronouncing the sentences properly. Several prior works simply use linguistic word embeddings from NLP tasks, but the improvement is often limited. In this paper, we propose using acoustic word

embeddings for natural speech synthesis. Specifically, the word sequence is provided to the TTS system as well as the phoneme sequence after text normalization. We elaborate on the details in the following subsections.

4.1. Text Normalization

The raw text contains complicated situations such as variations in word forms, rare words, and rare punctuation marks. Therefore, it is necessary to carry out text normalization before utilizing the words in TTS.

First, we used an NLP tool, Stanza [14], to convert each word to its prototype. Accordingly, the generated words contain no plural form, the third person singular, past tense, etc. This is a crucial step to dramatically reduce the vocabulary size and alleviate the sparsity in acoustic word embeddings training. Then, we removed rare punctuation marks, preserving only commas, periods, colons, semicolons, question marks, and exclamatory marks.

In order to make sure that each acoustic word embedding is well trained, we only considered high-frequency words in the training set. In this work, we set a word frequency threshold. Only the words with frequencies higher than the threshold were included in the vocabulary, while the other words were treated as out-of-vocabulary (OOV) words.

We also computed the alignment between word sequence and phoneme sequence. At the position where a silence appears in the phoneme sequence, if there is a corresponding punctuation mark, we aligned the punctuation mark with the silence. Otherwise, we added a blank token in the word sequence for proper alignment.

Figure 2 illustrates an example of text normalization. Here, we transformed the "did" and "asked" to their prototype "do" and "ask". Quotation marks were removed, and the "farmer" was treated as an OOV. We added a blank token to the word sequence to align it with the silence at the beginning of the phoneme sequence.

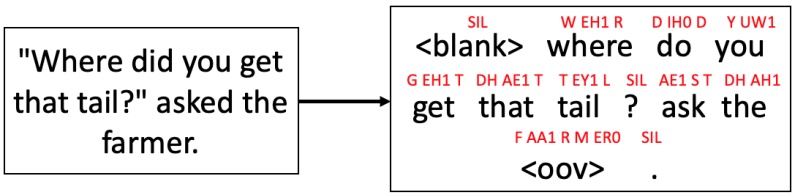

Figure 2. An example of text normalization.

4.2. Model Architecture

In this work, we utilized both words and phonemes as the inputs. Thus, we introduced a word encoder and a word-phoneme aligner in our TTS system, whose architecture is shown in Figure 3. The word encoder takes the normalized word sequence $\mathbf{w}$ as input and generates a hidden state sequence $\mathbf{h}_w$. Then, each hidden state in $\mathbf{h}_w$ is duplicated according to the number of phonemes aligned with the corresponding word. Accordingly, the output $\mathbf{h}_w^{dup}$ has the same sequence length as the phoneme encoder output $\mathbf{h}_p$. Then, we concatenated $\mathbf{h}_w^{dup}$ and $\mathbf{h}_p$ together to obtain $\mathbf{h}$, which was then used for phone-level prosody prediction. Generally, the final hidden states $\mathbf{h}$ were obtained by encoding the word and phoneme sequence, that is

$$\mathbf{h} = Encode(\mathbf{p}, \mathbf{w}) \tag{6}$$

The word encoder contains a 1D convolutional layer with a kernel size of 3 followed by six layers of Transformer blocks [15]. The convolutional layer is designed to directly consider the adjacent word contexts, and the Transformer layers are used for sequence modeling. It should be noted that all the word embeddings and the word encoder are jointly trained with the entire TTS system. The training criteria are the same as the basic TTS model, which is defined in Equation (5).

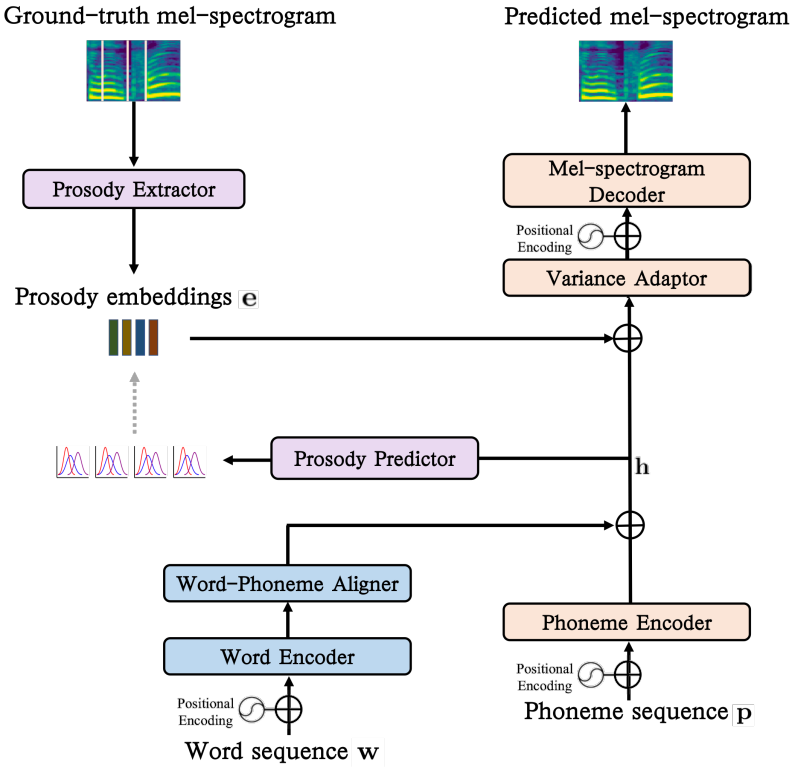

Figure 3. Model architecture with acoustic word embeddings.

5. Experiment and Result

5.1. Experimental Setup

LJSpeech [16] is a single speaker English dataset containing about 24 h worth of speech and 13,100 utterances. We randomly left out 50 utterances for validation and testing. The speech was resampled to 16KHz for simplicity. Before training TTS, we computed the phoneme alignment of the training data with an HMM-GMM ASR model trained on Librispeech [17] and then extracted the duration of each phoneme from the alignment for training.

The TTS models in this work are based on FastSpeech2 [5] with GMM-based prosody modeling [13]. The number of the Gaussian components in the GMMs was set to 20, and the β in Equation (5) was set to 0.02. An Adam optimizer [18] was used for TTS training in conjunction with a Noam learning rate scheduler [15]. We used a 320-dimensional mel-spectrogram as the acoustic feature with a 12.5 ms frame shift and 50ms frame length. MelGAN [19] was used as the vocoder for waveform reconstruction.

5.2. Word Encoder Architectures

In this section, we compare the performance of three common architectures for the word encoder. (1) None: the baseline that not uses the word encoder, as is described in Section 3. (2) BLSTM: a layer of 512 dimensional bi-directional LSTM. (3) Transformer: six layers of 512 dimensional Transformer blocks. (4) Conv+Transformer: a layer of 1D CNN with kernel size 3 followed by six layers of 512 dimensional Transformer blocks.

We first investigated the model size and the inference speed of the TTS systems. We synthesized the test set on an Intel Xeon E5-2680 CPU. As is shown in Table 1, when the model size grows, the inference speed becomes slow. The largest model with Conv+Transformer has an inference speed 39% slower than the baseline that not uses word encoder. If we use BLSTM as the word encoder, both the model size and the inference speed are very close to the baseline.

Table 1. Number of parameters and inference speed (second/frame) with various encoder architectures.

Architecture	Num Params	Inference Speed
None	96.075 M	1.7138×10^{-3}
BLSTM	98.341 M	1.9806×10^{-3}
Transformer	132.872 M	2.2783×10^{-3}
Conv+Transformer	133.003 M	2.3787×10^{-3}

Figure 4 illustrates the log-likelihood curves of phone-level prosodies with various word encoder architectures. In both the training and validation sets, we can observe that all the systems with acoustic word encoders outperform the baseline system that does not use word information. Moreover, the Conv+Transformer achieves the best performance out of the three common architectures. This is reasonable because it is already known that the Transformer has better capabilities than a simple LSTM [15] in sequence modeling, and the convolutional layer directly considers the adjacent word contexts. Therefore, we used Conv+Transformer in all the following experiments for the best performance.

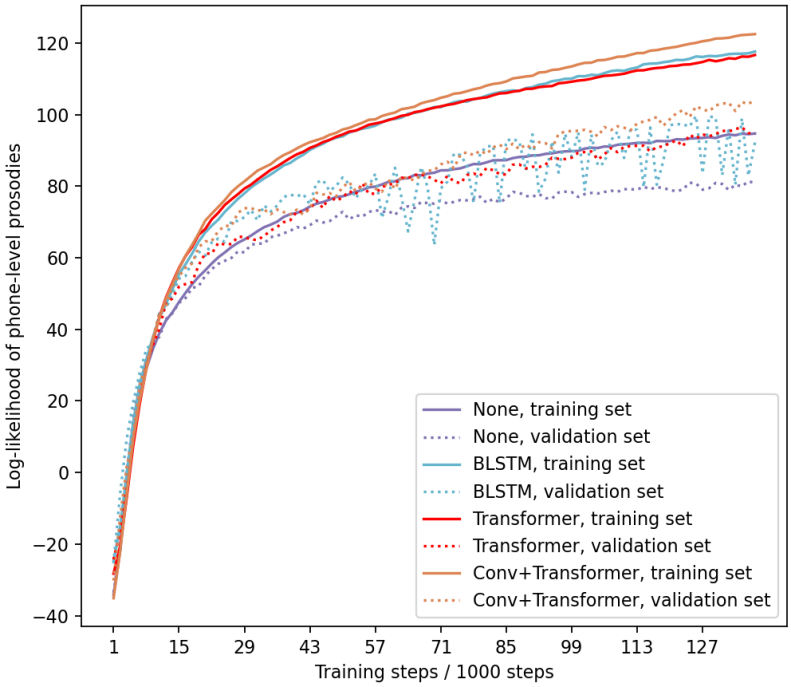

Figure 4. The log-likelihood curves of phone-level prosodies with various word encoder architectures.

5.3. Word Frequency Threshold

In this work, we only considered high-frequency words in the training set and treated the other words as OOV. The word frequency threshold determines the vocabulary size, which also affects the system's performance. Here, we apply three different thresholds, i.e., 10, 50, and 100, and then count the vocabulary size and the ratio of OOV words in the training set. The results are shown in Table 2. For example, when the word frequency threshold is set to 50, the vocabulary contains 529 words and covers 77.3% of words in the training set. Generally, when the word frequency threshold grows, the vocabulary size decreases, and the OOV ratio increases.

We also plotted the log-likelihood curves of phone-level prosodies with various word frequency thresholds. As is depicted in Figure 5, we saw the best performance at the threshold of 50. When the threshold is too large, fewer words are considered, and the system performance is harmed. However, when the threshold is too small, many low-frequency words are included in the vocabulary. These acoustic word embeddings are only trained with very limited data, which also leads to a decline in system performance.

Table 2. The vocabulary sizes and OOV ratios with various word frequency threshold.

Word Freq Thres	Vocabulary Size	OOV Ratio
10	2118	0.087
50	529	0.227
100	266	0.300

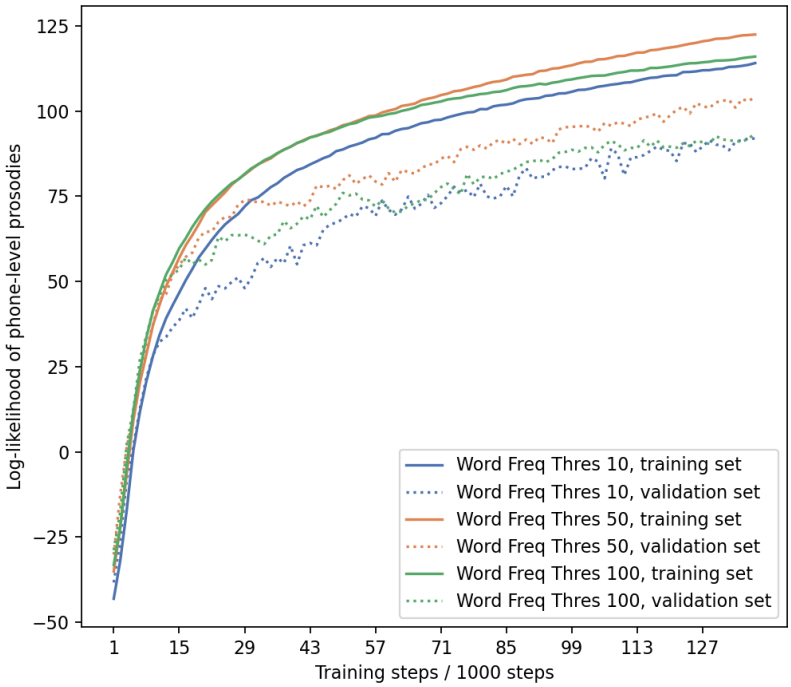

Figure 5. The log-likelihood curves of phone-level prosodies with various word frequency thresholds.

5.4. Naturalness

For simplicity, we abbreviate the system with acoustic word embeddings to AWE. With the analysis above, we use Conv+Tranformer architecture as the word encoder and set the word frequency threshold to 50 in AWE. In this section, we compare AWE with three other systems in terms of naturalness. (1) None: As is described in Section 3, we first built a basic FastSpeech2-based TTS system without using word information. (2) GloVe: We followed the prior work [8] that extracted linguistic word embeddings from pretrained NLP tasks. Similarly, the word embeddings were encoded and then added to the phoneme encoder output. In our experiments, we obtained the linguistic word embeddings from GloVe [20]. (3) BERT: We followed prior work [11] that extracted word representations from pretrained BERT and then applied the word representations to the TTS system. It should be noted that BERT takes subword units as input, and each word often corresponds to multiple BERT hidden states. Thus, we applied an average pooling to the multiple states in order to obtain the word representations [21].

An AB preference subjective listening test was carried out in terms of naturalness (The audio examples are available at https://cpdu.github.io/acoustic_word_embedding (accessed on 11 August 2021)). Ten listeners with no prior experience in TTS participated in this test. Ten pairs of synthetic speeches were provided to the participant in one trial to select the better one of each pair. Each pair consists of one speech randomly selected from 50 AWE synthetics and the other randomly chosen from 50 BERT/GloVe/None synthetics. Each participant went through three trials (BERT, GloVe, and None) and made 30 judgments in total. Figure 6 demonstrates the subjective results. As expected, the proposed system with acoustic word embeddings not only outperformed the baseline that does not use any word information directly, but also outperforms the systems that use linguistic word embeddings.

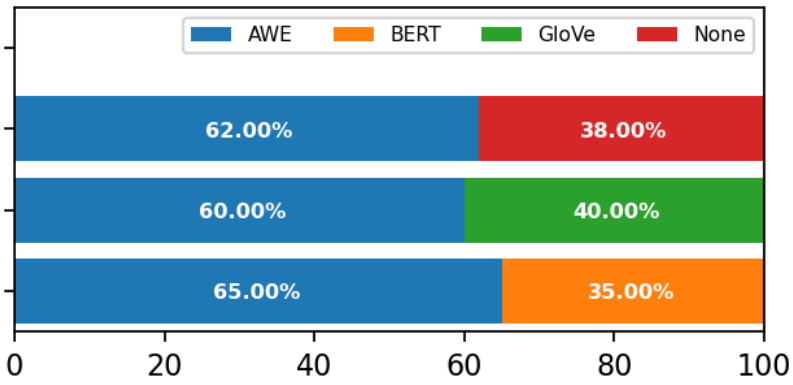

Figure 6. AB preference test in terms of naturalness. The *p*-values are 0.0105, 0.0284 and 0.0018, respectively.

6. Conclusions

In this paper, we propose a novel and effective approach that directly trains acoustic word embeddings in the TTS system. Both the phoneme and word sequences were utilized as input to the TTS system and passed through two encoders separately. The two output hidden states were then concatenated for phone-level prosody prediction. Our experiments on the LJSpeech dataset showed that using convolution followed by Transformer layers as the word encoder achieves the best performance. We also find that the word frequency threshold should be carefully selected. A too large or too small threshold can lead to a decline in performance. Finally, we compared the proposed system with the baseline that does not directly use word information and several prior works that use pretrained linguistic word representations. The subjective listening test showed that our system outperforms all the other systems in terms of naturalness.

Author Contributions: Conceptualization, C.D.; data curation, F.S.; formal analysis, F.S. and C.D.; supervision, K.Y.; writing—original draft, F.S. and C.D.; writing—review and editing, C.D. and K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The LJ Speech Dataset [16]. Available online: https://keithito.com/LJ-Speech-Dataset/ (accessed on 16 August 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, Y.; Skerry-Ryan, R.J.; Stanton, D.; Wu, Y.; Weiss, R.J.; Jaitly, N.; Yang, Z.; Xiao, Y.; Chen, Z.; Bengio, S.; et al. Tacotron: Towards End-to-End Speech Synthesis. In Proceedings of the ISCA Interspeech, Stockholm, Sweden, 20–24 August 2017; pp. 4006–4010.
2. Shen, J.; Pang, R.; Weiss, R.J.; Schuster, M.; Jaitly, N.; Yang, Z.; Chen, Z.; Zhang, Y.; Wang, Y.; Ryan, R.; et al. Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions. In Proceedings of the IEEE ICASSP, Calgary, AB, Canada, 15–20 April 2018; pp. 4779–4783.
3. Li, N.; Liu, S.; Liu, Y.; Zhao, S.; Liu, M.; Zhou, M. Close to human quality TTS with transformer. *arXiv* **2018**, arXiv:1809.08895.
4. Ren, Y.; Ruan, Y.; Tan, X.; Qin, T.; Zhao, S.; Zhao, Z.; Liu, T. FastSpeech: Fast, Robust and Controllable Text to Speech. In Proceedings of the NIPS, Vancouver, BC, Canada, 8–14 December 2019; pp. 3165–3174.
5. Ren, Y.; Hu, C.; Qin, T.; Zhao, S.; Zhao, Z.; Liu, T. FastSpeech 2: Fast and High-Quality End-to-End Text-to-Speech. *arXiv* **2020**, arXiv:2006.04558.
6. Zen, H.; Senior, A.W.; Schuster, M. Statistical parametric speech synthesis using deep neural networks. In Proceedings of the IEEE ICASSP, Vancouver, BC, Canada, 26–31 May 2013; pp. 7962–7966.
7. Wang, P.; Qian, Y.; Soong, F.K.; He, L.; Zhao, H. Word embedding for recurrent neural network based TTS synthesis. In Proceedings of the IEEE ICASSP, Brisbane, Australia, 19–24 April 2015; pp. 4879–4883.
8. Ming, H.; He, L.; Guo, H.; Soong, F.K. Feature reinforcement with word embedding and parsing information in neural TTS. *arXiv* **2019**, arXiv:1901.00707.

9. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the NAACL, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
10. Hayashi, T.; Watanabe, S.; Toda, T.; Takeda, K.; Toshniwal, S.; Livescu, K. Pre-Trained Text Embeddings for Enhanced Text-to-Speech Synthesis. In Proceedings of the ISCA Interspeech, Graz, Austria, 15–19 September 2019; pp. 4430–4434.
11. Xiao, Y.; He, L.; Ming, H.; Soong, F.K. Improving Prosody with Linguistic and Bert Derived Features in Multi-Speaker Based Mandarin Chinese Neural TTS. In Proceedings of the IEEE ICASSP, Barcelona, Spain, 4–8 May 2020; pp. 6704–6708.
12. Kenter, T.; Sharma, M.; Clark, R. Improving the Prosody of RNN-Based English Text-To-Speech Synthesis by Incorporating a BERT Model. In Proceedings of the ISCA Interspeech, Shanghai, China, 25–29 October 2020; pp. 4412–4416.
13. Du, C.; Yu, K. Mixture Density Network for Phone-Level Prosody Modelling in Speech Synthesis. *arXiv* **2021**, arXiv:2102.00851.
14. Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; Manning, C.D. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Online, 5–10 July 2020.
15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the NIPS, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
16. Ito, K.; Johnson, L. The LJ Speech Dataset. 2017. Available online: https://keithito.com/LJ-Speech-Dataset/ (accessed on 11 August 2021).
17. Zen, H.; Dang, V.; Clark, R.; Zhang, Y.; Weiss, R.J.; Jia, Y.; Chen, Z.; Wu, Y. LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech. In Proceedings of the Interspeech, Graz, Austria, 15–19 September 2019. [CrossRef]
18. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
19. Kumar, K.; Kumar, R.; de Boissiere, T.; Gestin, L.; Teoh, W.Z.; Sotelo, J.; de Brébisson, A.; Bengio, Y.; Courville, A.C. MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis. In Proceedings of the NIPS, Vancouver, BC, Canada, 8–14 December 2019; pp. 14881–14892.
20. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
21. Zhang, S.; Ma, X.; Duh, K.; Durme, B.V. AMR Parsing as Sequence-to-Graph Transduction. In Proceedings of the ACL, Florence, Italy, 28 July–2 August 2019; pp. 80–94.