INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Polyphone Disambiguation for Mandarin Chinese Using Conditional Neural Network with Multi-level Embedding Features

Zexin Cai[1], *Yaogen Yang*[1,2], *Chuxiong Zhang*[1], *Xiaoyi Qin*[1,3], *Ming Li*[1]

[1]Data Science Research Center, Duke Kunshan University, Kunshan, China
[2]College of Information and Computer, Taiyuan University of Technology, Taiyuan, China
[3]School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
ming.li369@dukekunshan.edu.cn

Abstract

This paper describes a conditional neural network architecture for Mandarin Chinese polyphone disambiguation. The system is composed of a bidirectional recurrent neural network component acting as a sentence encoder to accumulate the context correlations, followed by a prediction network that maps the polyphonic character embeddings along with the conditions to corresponding pronunciations. We obtain the word-level condition from a pre-trained word-to-vector lookup table. One goal of polyphone disambiguation is to address the homograph problem existing in the front-end processing of Mandarin Chinese text-to-speech system. Our system achieves an accuracy of 94.69% on a publicly available polyphonic character dataset. To further validate our choices on the conditional feature, we investigate polyphone disambiguation systems with multi-level conditions respectively. The experimental results show that both the sentence-level and the word-level conditional embedding features are able to attain good performance for Mandarin Chinese polyphone disambiguation.

Index Terms: Grapheme-to-phoneme conversion, polyphone disambiguation, text-to-speech, sentence encoding

1. Introduction

The grapheme-to-phoneme (G2P) conversion is a fundamental front-end procedure in the Chinese Text-to-Speech (TTS) synthesis system, either the traditional HMM-based speech synthesis system [1, 2] or the End-to-End speech synthesis system [3, 4, 5, 6]. G2P typically generates a sequence of phones from a sequence of characters or graphemes [7]. According to the characteristics of Mandarin Chinese, there are at least 13000 commonly used Chinese characters. However, the number considerably declines to 1300 when converting the characters into phonologically allowed syllables, and even less when using Latin alphabet representation. It appears to be a suitable choice of using phonemes or syllables as units for a TTS synthesis system in a way for effective and better performance [8, 9]. While the G2P system in English TTS synthesis system aims to produce the phoneme sequences for the out-of-lexicon words, The target of a G2P system in Chinese TTS synthesis system is to convert Chinese characters to pinyins (phoneme representations with Latin alphabet in Mandarin Chinese) [10]. Yet one single Chinese character could have several different pronunciations in terms of different usages in a sentence. This kind of characters is called polyphonic characters. Therefore, other than the G2P system, the polyphone disambiguation system is developed to choose the correct pronunciation of a polyphonic character from several candidates based on the context. This issue is also considered to be a homograph problem, which has important applications in speech synthesis and is still not solved today [11].

Rule-based algorithms [12, 13, 14] and data-driven methods [15, 16, 17] are two frequently used approaches for polyphone disambiguation. The rule-based system normally chooses the pronunciation of a polyphonic character depending on the segmental text and a well-designed dictionary. However, this method requires language expertise to produce an elaborate text-analysis system for sentence segmentation as well as a robust dictionary. The dictionary today still cannot cover all the polyphonic cases. As for data-driven approaches, mostly the polyphone disambiguation is considered as a classification task. Inspired by the approaches for G2P in English[7, 18], in recent years more research works on polyphone disambiguation using statistical machine learning techniques, like Decision Tree [16] and Maximum Entropy Model [16, 19].

In this paper, we introduce a data-driven approach using the conditional neural network architecture [20] for polyphone disambiguation. Besides using the polyphonic character embedding feature as the network input, we obtain auxiliary features from the corresponding sentence as a condition for predicting the correct pronunciation. Previous research works in polyphonic character show that: 1) The utilization of context is an effective way to solve the pronunciation disambiguation of Chinese polyphonic characters; 2) Most polyphonic word, which comprises by polyphonic character, could be used to determine the pronunciation of the polyphonic character [12]. In the light of these two characteristics, we first design an encoder module using a recurrent neural network (RNN) structure to extract the sentence-level encoding feature as the context condition. Basically, we embed each character in the sentence and adopt the bi-directional long short-term memory (BLSTM) structure to accumulate the forward context information and backward context information as the conditional feature in the sentence-level. Besides, we use a publicly released and pre-trained word-to-vector dictionary for word-level conditional vector lookup. The prediction network maps the polyphonic character embedding features and the auxiliary features to their unique pronunciation. We investigate three systems under different combinations of conditional features on a publicly available dataset. Results show that either the word-level conditional feature or the sentence-level conditional feature yields significant improvement on polyphone disambiguation.

A similar approach presented in [15] treats polyphone disambiguation as a sequence tagging task and uses BLSTM as the sequence-to-sequence generation model. However, the BLSTM structure in our proposed architecture serves as a sentence-level conditional feature extractor. Our system performs better as the accuracy reaches 94.69% on a same evaluation set.

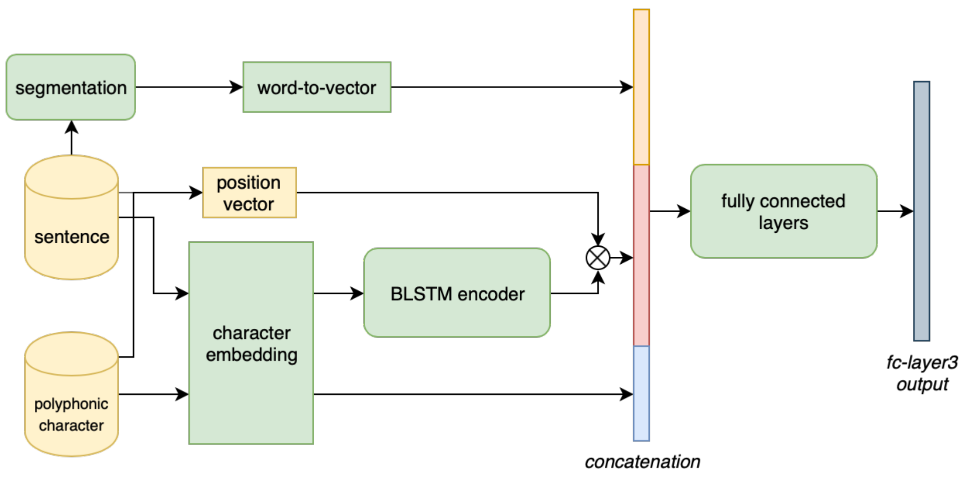

Figure 1: *The network architecture of our proposed system*

2. Chinese Polyphonic Characters

Except for the monophonic characters in Mandarin Chinese, there are polyphonic characters that refer to those with more than one pronunciations. Specifically, we use a mapping function to formulate the conversion from a character to its corresponding pronunciations. Function f is defined as follows:

$$f : C \rightarrow P \tag{1}$$

where C denotes the set of all Chinese characters and P denotes the set of all possible pinyins (the official romanization system for Standard Mandarin Chinese with tone information). The number of all Chinese characters is more than 80000 while the number of all pinyins is about 300. Given a character c, the output p_c would be:

$$p_c = \{p_1, p_2, ..., p_n\} \tag{2}$$

The character c belongs to a polyphonic character when $|p_c| > 1$. For example, the Chinese character “背” can be pronounced as either “bei1” or “bei4”. The number 1 to 5 denotes the tone marks since Mandarin Chinese is a tonal language. Therefore, “bei1” and “bei4” are two different pronunciations.

The pronunciation of a Chinese polyphonic character cannot be determined unless providing its context. In most cases, the pronunciation of a polyphonic character corresponds to the word that comprises the polyphonic character. The characteristics or properties of a certain word relates to the pronunciation of the polyphonic character. For example, the character “将” is pronounced “jiang1” in a word where it is used as a verb or adverb, while it is pronounced “jiang4” when the word is a noun. Specifically, “将” is pronounced “jiang1” in the word “将要”, which is an adverb; “将” is pronounced “jiang4” in the word “大将”, which is a noun. Sometimes the meaning of a word also determines the pronunciation of a polyphonic character. So we can rewrite formula 1 and 2 with:

$$f : C, W_C \rightarrow P \tag{3}$$

$$f(c|w_c) = \{p_1, p_2, ..., p_n\} \tag{4}$$

where W_C is the Chinese word set containing polyphonic characters. Normally, the pronunciation of a character c given the word w_c is unique. These words are called monophonic character words.

With a well-done segmentation, we could find a unique pronunciation of the polyphonic character from word pieces except 53 special words that satisfy $|f(c|w_c)| > 1$. The pronunciation of these polyphonic character words, for example, “朝阳” (zhao1yang2 or chao2yang2), could only be determined in a given sentence with word-level context. Moreover, when a polyphonic character stays a single character as segmental piece after a well-done segmentation, we could only utilize the context information for polyphone disambiguation. For example, in sentence “我不注重得与失” (Gains and losses mean nothing to me) and “我得关注相关动态” (I have to pay attention to the relevant news), where the segmentations are “我\不注重\得\与\失” and “我\得\关注\相关\动态”, the polyphonic word “得” is pronounced “de2” and “dei3” respectively only depending on its context.

Therefore, the pinyin of a polyphonic character is unique given its corresponding word and sentence. Which can be defined as:

$$f : C, W_C, T_C \rightarrow P \tag{5}$$

where T_C denotes the context and W_C could be $\varnothing$ if the polyphonic character word only contains a single character.

3. Method

Different from the traditional grapheme-to-phoneme (G2P) conversion, the polyphone disambiguation is considered as a classification problem. Specifically, the polyphone disambiguation system converts a polyphonic character to its corresponding pinyin. Our proposed system is shown in Figure 1. In terms of the characteristics and properties of the polyphonic character outlined in the previous section, we explore the word-level conditional feature and sentence-level conditional feature from the input sentence to help predict the pinyin.

3.1. Embedding

It is a typical case to convert the characters and sentence into vectors when applying the neural network approaches. First, we initialize an embedding table with size $N_c \times D_c$ for character-to-vector lookup, where N_c is the number of all characters and D_c denotes the character embedding size. Before sentence em-

Table 1: *Detailed network structure and configurations of our proposed three systems*

Input	**Size**			System CW	System CC	System CWC
character	$B \times 1$			√	√	√
sentence	$B \times L$			-	√	√
segmented word	$B \times 1$			√	-	√
Layer	**Input size**	**Output size**	**Configurations**			
char-embedding	$B \times 1$	$B \times 100$	-	√	√	√
text-embedding	$B \times L$	$B \times L \times 100$	-	-	√	√
word2vec	$B \times 1$	$B \times 200$	-	√	-	√
BLSTM encoder	$B \times L \times 100$	$B \times L \times 512$	dropout rate: 0.1 fw-lstm size: 256 bw-lstm size: 256	-	√	√
concatenation	-	$B \times C$	-	$C = 300$	$C = 612$	$C = 812$
fc-layer1	$B \times C$	$B \times 512$	activation: RELU dropout rate: 0.1	$C = 300$	$C = 612$	$C = 812$
fc-layer2	$B \times 512$	$B \times 1024$	activation: RELU dropout rate: 0.1	√	√	√
fc-layer3	$B \times 1024$	$B \times 285$	activation: None dropout rate: 0	√	√	√

bedding, we pad the character sequence to the max length of the sentences in each batch with the symbol "|" , which does not originally appear in the Chinese text to avoid any possible conflicts, without affecting the performance of the encoder module. Hence, the sentence embedding size would be $B \times L_{max} \times D_c$. B refers to the batch size in the network training phase and evaluation phase.

As for word-level conditional feature, we segment the input sentence into word pieces and obtain a word sequence from a pre-trained word-to-vector lookup table. We only choose the word that comprises the polyphonic character from the segmental sentence for table lookup.

3.2. Bidirectional LSTM Encoder

The Recurrent Neural Network (RNN) architecture has an elegant way of dealing with sequential problems since it is able to embody correlations between samples in the sequence[21, 22, 23] . In order to address the exploding and vanishing gradient problems in RNN, the long short-term memory (LSTM) structure was proposed and successfully kept track of arbitrary long-term dependencies between the elements in the input sequences [24]. The LSTM structure is widely used in addressing sequence-to-sequence problems since the fundamental module of Encoder-Decoder architecture has been proven to be very effective. Encoder and decoder are the two main modules in this kind of architecture. The encoder aims to encode the input sequence to a fixed-length vector for the decoder to map the vector back into an output sequence. The Encoder-Decoder architecture demonstrated state-of-the-art performance in researches on the sequential problem including text translation, speech recognition, speech synthesis [6, 25, 26].

In our case, we adopt the encoder component to extract the sentence-level conditional feature from the character sequence. One problem of the LSTM is that it is unidirectional. The LSTM can only accumulate the information of the sequence in one direction. This paper uses the bi-directional LSTM [27] to accumulate the character correlations in both directions. Different from [15], which uses BLSTM as a sequence-to-sequence generator to obtain the pinyin of a polyphonic character, we use BLSTM as an encoder to extract the sentence-level conditional feature. Suppose the character embedding sequence is $T_c = [c_1^\top, c_2^\top, c_3^\top, \cdots c_t^\top, \cdots, c_L^\top]$, we obtain the sentence-level conditional feature $C_{sentence}^c$ as follows:

$$\begin{aligned} C_{sentence}^c &= z_c^t \cdot BLSTM(T_c) \qquad (6) \\ &= z_c^t \cdot concat(fw, bw) \qquad (7) \end{aligned}$$

where z_c^t is the one-hot vector that denotes the polyphonic character position in the character sequence, fw is the output sequence of the forward LSTM given input T_c and bw is the output sequence of backward LSTM. Operation $concat()$ means concatenation. The size of z_c^t is $1 \times L$ and the size of $BLSTM(T_c)$ is $L \times (D_{fw} + D_{bw})$, where D_{fw} and D_{bw} are the output size of forward LSTM and backward LSTM.

3.3. Prediction Network

After concatenating the word-level conditional feature, sentence-level conditional feature and the polyphonic character embedding vector, we use several fully-connected layers following a $softmax$ layer for classification to predict the corresponding pinyin. The length of output vector equals the number of all possible pinyins in our polyphonic character database.

4. Experimental Results

4.1. Polyphonic Character Database

For training and evaluating our proposed polyphone disambiguation systems, we use a publicly available dataset from Beijing Data-Baker Science and Technology Ltd which contains 150 frequently used polyphonic characters and their 151585 corresponding sentences. We divide the corpus into a training set with 140794 sentences and an evaluation set with 10791 sentences [28]. The evaluation set is 7% of most character-pinyin pairs and well split in with all samples concerning every different character-pinyin pair. However, some character-pinyin pairs are less than 15 samples and 20% of those pairs are split for evaluation set.

Table 2: *Experimental performance of six individual systems*

polyphonic character	high-fre pinyin	low-fre pinyin	high-fre pinyin rate	[15] 0 word accuracy	[15] 1 word accuracy	[15] 2 words accuracy	Our System CW accuracy	Our System CC accuracy	Our System CWC accuracy
传	chuan2	zhuan4	86.11%	88.89%	88.89%	88.89%	**94.44%**	91.67%	**94.44%**
只	zhi3	zhi1	70.59%	93.38%	93.38%	**94.12%**	86.03%	**94.12%**	93.38%
处	chu4	chu3	81.58%	91.67%	86.11%	91.67%	**94.44%**	88.89%	**94.44%**
少	shao3	shao4	93.98%	96.24%	95.49%	96.24%	96.24%	93.23%	**96.99%**
为	wei2	wei4	59.28%	59.58%	82.38%	82.9%	62.69%	81.35%	**86.53%**
藏	zang4	cang2	54.55%	80%	79.09%	**85.45%**	80%	76.36%	81.82%
overall	-	-	86.77%	89.13%	91.96%	91.6%	92.44%	92.44%	**94.69%**

4.2. Word-level Embedding

The text segmentation tool we used for obtaining the word and phrase pieces is Jieba[1] Python package. We use the released Tencent AI Lab Embedding Corpus for Chinese Words and Phrases as the pre-trained word vector lookup table. This corpus provides 200-dimensional vector representations for over 8 million Chinese words and phrases embedding. This pre-trained model was trained with large-scale text collected from news, webpages, and novels using directional skip-gram [29, 30].

4.3. System Setup

Our experimental setup is shown in Table 1. In this paper, we propose three different systems according to different combinations of sentence-level conditional feature and word-level conditional feature:

- System WC: concatenate the word-level conditional feature and polyphonic character embedding vector for predicting the corresponding pinyin.
- System CC: concatenate the sentence-level conditional feature and polyphonic character embedding vector for predicting the corresponding pinyin.
- System CWC: adopt both the word-level and sentence-level feature as the condition for polyphonic character embedding vector to predict pinyin.

As shown in Table 1, we adopt a single BLSTM encoder with input size $B \times L \times 100$, where B denotes the batch size and L denotes the length of the padding character sequences in a batch. Both the forward LSTM size and backward LSTM size is 256. The dropout rate of LSTM is set to 0.1 to avoid overfitting [31]. For the prediction module, we adopt three fully connected layers with size 512, 1024 and 285 respectively. The output size 285 is equal to the number of all possible pinyins in this polyphonic character database. The activation function of the first two fully connected layers is RELU. We use dropout layers with 0.1 dropout rate after the first two fully connected layers in the training phase. Depending on different input conditional features, the input size C of prediction network can be 300, 612 and 812, respectively. The concatenated feature comprises the 100-dimensional polyphonic character embedding feature and the conditional feature (200-dimensional word-level embedding vector, 512-dimensional sentence-level encoding feature). We adopt the stochastic gradient descent (SGD) algorithm with an initial learning rate 0.1 for the training phase. The learning rate decays every 600 epochs exponentially to 10^{-4}.

We also implement three baseline systems following [15] for comparison. We strictly follow the approach describes in [15] which adopts two LSTM layers with size 512 and the NLPIR toolkit [32] for POS tagging on the text. We use the polyphonic character database described in section 4.1 for training and evaluating since we do not have the personal labelled data used in [15]. The approach presented in [15] had compared with other polyphone disambiguation approaches and shown that it reaches a better performance. Three systems regarding the different length of context inputs are implemented for comparison in our experiment:

- 0 word, not using the context information
- 1 word, using the past and future context information in 1 word, 3 words in total
- 2 words, using the past and future context information in 2 words, 5 words in total

4.4. System evaluation results

Table 2 gives the performance regarding six systems illustrated in Section 4.3. We list several polyphonic characters for comparison. Both the word-level conditional feature and sentence-level conditional feature help improve the disambiguation system as the performance of System CW and System CC reach 92.44%. By concatenating the word embedding conditional feature and sentence-level conditional feature as auxiliary condition, our approach achieves the best performance with 94.69% accuracy, which is 2.73% higher than the best system in [15].

5. Conclusions

In this paper, we propose a data-driven approach using conditional neural network architecture for Mandarin Chinese polyphone disambiguation. We explore sentence-level encoding vector as a condition as well as the word-level vector obtained from a pre-trained word-to-vector lookup table. Results show that the sentence-level conditional feature obtained from a single bi-directional long short-term memory structure is very useful for polyphone disambiguation. Both the word-level conditional feature and sentence-level conditional feature help improve the disambiguation system as the accuracy reaches 92.44%. Finally, The final system achieves significant improvements with 94.69% accuracy on the evaluation set.

6. Acknowledgments

This research was funded in part by the National Natural Science Foundation of China (61773413), Natural Science Foundation of Guangzhou City (201707010363), Six Talent Peaks project in Jiangsu Province (JY-074), Science and Technology Program of Guangzhou City (201903010040).

[1]https://github.com/fxsjy/jieba

7. References

[1] Y. Qian, F. Soong, Y. Chen, and M. Chu, "An HMM-Based Mandarin Chinese Text-To-Speech System." in *2006 International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2006, pp. 223–232.

[2] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The HMM-Based Speech Synthesis System (HTS) Version 2.0." in *6th ISCA Workshop on Speech Synthesis (SSW-6)*, 2007, pp. 294–299.

[3] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: 2000-Speaker Neural Text-to-Speech," *CoRR*, vol. abs/1710.07654, 2017.

[4] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, and J. Raiman, "Deep Voice: Real-time Neural Text-to-Speech," *CoRR*, vol. abs/1702.07825, 2017.

[5] S. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep Voice 2: Multi-Speaker Neural Text-to-Speech," *CoRR*, vol. abs/1705.08947, 2017.

[6] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[7] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-Phoneme Conversion Using Long Short-Term Memory Recurrent Neural Networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4225–4229.

[8] L.-S. Lee, C.-Y. Tseng, and M. Ouh-Young, "The Synthesis Rules in A Chinese Text-to-Speech System," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 9, pp. 1309–1320, 1989.

[9] H. Dong, J. Tao, and B. Xu, "Grapheme-to-Phoneme Conversion in Chinese TTS System," in *2004 International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2004, pp. 165–168.

[10] L. Yi, L. Jian, H. Jie, and Z. Xiong, "Improved Grapheme-to-Phoneme Conversion for Mandarin TTS," *Tsinghua Science & Technology*, vol. 14, no. 5, pp. 606–611, 2009.

[11] M. Rashad, H. M. El-Bakry, I. R. Isma'il, and N. Mastorakis, "An Overview of Text-to-Speech Synthesis Techniques," *Latest trends on communications and information technology*, pp. 84–89, 2010.

[12] H. Zhang, J. Yu, W. Zhan, and S. Yu, "Disambiguation of Chinese Polyphonic Characters," in *The First International Workshop on MultiMedia Annotation (MMA2001)*, vol. 1, 2001, pp. 30–1.

[13] Z. Zirong, C. Min, and C. Eric, "An Efficient Way to Learn Rules for Grapheme-to-Phoneme Conversion in Chinese," in *2002 International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2002, pp. 59–62.

[14] F.-L. Huang, "Disambiguating Effectively Chinese Polyphonic Ambiguity Based on Unify Approach," in *2008 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 6, 2008, pp. 3242–3246.

[15] C. Shan, X. Lei, and K. Yao, "A Bi-directional LSTM Approach for Polyphone Disambiguation in Mandarin Chinese," in *2017 International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2017.

[16] F. Z. Liu and Y. Zhou, "Polyphone Disambiguation Based on Maximum Entropy Model in Mandarin Grapheme-to-Phoneme Conversion," *Key Engineering Materials*, vol. 480-481, pp. 1043–1048, 2011.

[17] J. Liu, W. Qu, X. Tang, Y. Zhang, and Y. Sun, "Polyphonic Word Disambiguation with Machine Learning Approaches," in *2010 Fourth International Conference on Genetic and Evolutionary Computing (ICGEC)*, 2010, pp. 244–247.

[18] M. Bisani and H. Ney, "Joint-Sequence Models for Grapheme-to-Phoneme Conversion," *Speech communication*, vol. 50, no. 5, pp. 434–451, 2008.

[19] X. Mao, D. Yuan, J. Han, D. Huang, and H. Wang, "Inequality Maximum Entropy Classifier with Character Features for Polyphone Disambiguation in Mandarin TTS Systems," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 1125–1134.

[21] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent Neural Network Based Language Model," in *Eleventh annual conference of the international speech communication association (ISCA)*, 2010.

[22] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of Recurrent Neural Network Language Model," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5528–5531.

[23] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in *2013 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2013, pp. 6645–6649.

[24] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[26] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[27] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[28] https://www.data-baker.com/bz_dyz.html.

[29] Y. Song, S. Shi, J. Li, and H. Zhang, "Directional Skip-Gram: Explicitly Distinguishing Left and Right Context for Word Embeddings," in *the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, vol. 2, 2018, pp. 175–180.

[30] https://ai.tencent.com/ailab/nlp/embedding.html.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] L. Zhou and D. Zhang, "NLPIR: A Theoretical Framework for Applying Natural Language Processing to Information Retrieval," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 2, pp. 115–123, 2003.