INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Label Embedding for Chinese Grapheme-to-Phoneme Conversion

Eunbi Choi[1,2*], *Hwa-Yeon Kim*[2], *Jong-Hwan Kim*[2], *and Jae-Min Kim*[2]

[1]KAIST, South Korea
[2]NAVER Corp., South Korea

ebchoi1030@kaist.ac.kr, {hwayeon.kim, jhwan.kim, kjm.kim}@navercorp.com

Abstract

Chinese grapheme-to-phoneme (G2P) conversion plays a significant role in text-to-speech systems by generating pronunciations corresponding to Chinese input characters. The main challenge in Chinese G2P conversion is polyphone disambiguation, which requires selecting the appropriate pronunciation among several candidates. In polyphone disambiguation, calculating probabilities for the entire pronunciations is unnecessary since each Chinese character has only a few (mostly two or three) candidate pronunciations. In this study, we introduce a label embedding approach that matches the character embedding with the closest label embedding among the possible candidates. Specifically, negative sampling and triplet loss were applied to maximize the difference between the correct embedding and the other candidate embeddings. Experimental results show that the label embedding approach improved the polyphone disambiguation accuracy by 4.50% and 1.74% on two datasets compared to the one-hot label classification approach. Moreover, the bidirectional long short-term memory model with the label embedding approach outperformed the previous most advanced model, BERT, demonstrating outstanding performance in polyphone disambiguation. Lastly, we discuss the effect of contextual information in character embeddings on the G2P conversion task.

Index Terms: Grapheme-to-phoneme conversion, polyphone disambiguation, label embedding, negative sampling, text-to-speech

1. Introduction

Grapheme-to-phoneme (G2P) conversion is a task that converts letters into pronunciations, which is the key component in synthesizing high-quality voice in text-to-speech (TTS) systems. G2P in Chinese is particularly important as every Chinese character represents its own meaning. If a Chinese character is mispronounced, its meaning can be easily misinterpreted.

The Chinese G2P system maps the phonemes, known as pinyin, to every character in a given sentence. Pinyin consists of an alphabet part and a numeric part corresponding to the tone (e.g., *guang1* and *xie3*). Chinese characters are either monophones that can only be read with one pronunciation or polyphones that can be read with multiple pronunciations. The majority of Chinese characters are monophones with the same pinyin regardless of which part of the sentence they are located in. Therefore, monophones can be converted into pinyins simply using prior knowledge through direct one-to-one mapping.

However, polyphones are diversely pronounced, depending on the context. Every polyphone has a list of possible pinyins and is uttered as one pinyin among the candidates. In Sentence 1, the character 还 is pronounced as *hai2* in the first appearance and *huan2* in the second appearance. The meaning of this sentence is that "You still have to return him ten more dollars," and each pinyin is derived from the context of "still have to (还要)" and "return to (还给)."

- Sentence 1: 你还要还给他十美元
  Pron.: ni3 **hai2** yao4 **huan2** gei3 ta1 shi2 mei3 yuan2

The challenge in Chinese G2P conversion, therefore, is the disambiguation of polyphonic characters, which requires selecting the correct pinyin among the candidate pinyins of the polyphones depending on the context.

Mandarin Chinese is a tonal language, and pitch is an essential part of Chinese pronunciation. One syllable can be pronounced with four main tones and one neutral tone (or, as some say, five tones), leading to rise and fall in Chinese cadence. Therefore, when disambiguating Chinese characters, tone must be considered.

The tone of a character can differ from its canonical pinyin depending on the characters placed around it. For example, if there are two consecutive 3rd tone pinyin, native speakers pronounce the first 3rd tone pinyin with 2nd tone. In sentence 2, the pinyin of 只好 is written as *zhi3 hao3*, but is spoken as *zhi2 hao3*. Another case is in which the tone can be 5th tone, ignoring its own tone [1]. In addition to the aforementioned two situations, there may exist a gap between canonical pronunciation and spoken pronunciation depending on the context in which the character is located (see Sentence 3).

- Sentence 2: 只好认真工作
  Canonical Pron.: **zhi3** hao3 ren4 zhen1 gong1 zuo4
  Spoken Pron.: **zhi2** hao3 ren4 zhen1 gong1 zuo4
- Sentence 3: 几乎一模一样
  Canonical Pron.: ji1 hu1 **yi1** mu2 **yi1** yang4
  Spoken Pron.: ji1 hu1 **yi4** mu2 **yi2** yang4

In summarization, polyphone disambiguation is a task used to 1) understand the context and 2) select an appropriate pinyin among several pinyin candidates considering the natural tone. This is a challenging part of the Chinese G2P that needs to be addressed.

Several studies have been conducted on the Chinese G2P. In recent studies, a classification model using a bidirectional long short-term memory (BiLSTM) encoder to capture contextual information has been demonstrated to be an effective approach[2, 3, 4, 5]. In this study, we introduce the label embedding approach and negative sampling with a triplet loss for polyphone disambiguation and demonstrate its effectiveness with the BiLSTM model. This approach converts labels into dense label embeddings rather than one-hot vectors. Therefore, the classification problem is changed to identifying the nearest label embedding, also referred to as pinyin embedding in this paper.

*Work performed as an intern in Clova Voice&Avatar, NAVER Corp.

Our method achieves a better performance with higher polyphone disambiguation accuracies than general one-hot label classification, with improvements of 4.50% and 1.74% on two datasets. Our BiLSTM label embedding model also outperforms BERT [6] in the two experimental settings. To the best of our knowledge, this study is the first attempt to apply label embedding and negative sampling to Chinese G2P conversion.

2. Related work

Chinese G2P: Studies on Chinese G2P have generally used rule-based and data-driven approaches. Handcrafted rule-based G2P conversion systems require the knowledge and labor of linguistic experts to manually produce complicated rules [7]. Furthermore, there exists a dictionary-based method that determines pinyins utilizing information from a dictionary, but word segmentation and part-of-speech (POS) tagging must be preceded a priori [8].

The data-driven approaches that have been studied extensively in recent years utilize statistical methods or neural networks. Statistical algorithms like maximum entropy and decision trees are investigated to disambiguate polyphones [9, 10]. Feature selection is important for statistical algorithms, and n-grams, segmented words, and POS tags are used as features. Recent approaches use neural models such as BiLSTM or BERT as encoders to extract character-, word-, or sentence-level (or multi-level) features and classify them into appropriate pinyin with the following feed-forward layers [2, 3, 4]. These studies used cross-entropy as the loss function to train the models.

Label embedding for classification: Label embedding has been demonstrated to be effective [11, 12] in providing additional information to the classification model as an additional input. One study embedded the labels in the same latent space as the words embedded and leveraged the compatibility between the words and the labels to derive the attention score [11]. In another study, the compatibility between images and labels was measured by embedding each label into the attribute space [12].

Some studies [13, 14] treated labels as dense embeddings rather than one-hot vectors to leverage potential label information. The salient point of this approach is that it transforms a classification task into a vector matching task, which is a task searching for the closest label embedding. The text recognition problem is transformed into searching for the closest word label embedding to word image embedding by projecting the image embedding into the word label embedding space [13]. The dot product operation is used to compute the distance between the representations. A study implemented a fully connected (FC) layer, Matcher, to produce matching scores between word and label embeddings [14]. In our polyphone disambiguation work, we embedded Chinese characters and pinyins into the same space and determined the closest pinyin embedding to a character embedding by computing the cosine distance.

Negative sampling and Triplet Loss: The triplet loss as an error function causes the output of the model to get closer to positive samples and away from negative samples. Negative sampling and triplet loss are frequently used in various tasks [15, 16, 17, 18]. These methods were typically used in training word embeddings [15]. In a previous study [16], triplet loss was applied for natural language processing, vision, recommendation, and relational graphs. These methods were also applied for multivariate time series representation learning and image similarity learning [17, 18].

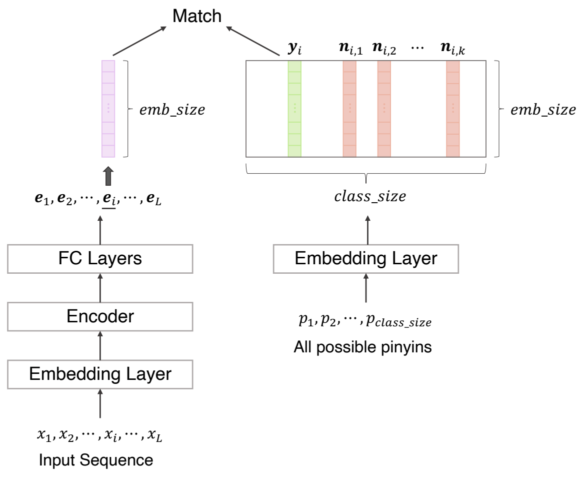
Match
y_i
$n_{i,1}$ $n_{i,2}$ ⋯ $n_{i,k}$
emb_size
emb_size
$e_1, e_2, \cdots, e_i, \cdots, e_L$
class_size
FC Layers
Embedding Layer
Encoder
$p_1, p_2, \cdots, p_{class\_size}$
All possible pinyins
Embedding Layer
$x_1, x_2, \cdots, x_i, \cdots, x_L$
Input Sequence

Figure 1: *Overall structure of the label embedding polyphone disambiguation system. Main workflow involves properly matching the characters and pinyins after they are embedded.*

3. Proposed model

3.1. Architecture

The overall structure of the method proposed in this study is shown in Figure 1. The input character sequence is denoted as $x = (x_1, x_2, \dots, x_L)$, where L denotes the character sequence length. In the character encoding process, shown in the left part of Figure 1, each word $x_i (1 \leq i \leq L)$ is converted into an embedding vector via the embedding layer. The vector corresponding to x_i contains contextual information from the encoder and becomes the final character representation e_i by the FC Layers. We denote the representations using $e = (e_1, e_2, \dots, e_L)$.

In the label encoding process, all possible pinyins $p_1, p_2, \dots, p_{class\_size}$ become vectors of the same size as character embedding through the pinyin embedding layer. Finally, the model is trained and the pinyin is inferred by calculating the matching scores between the character embedding e_i and the candidate pinyins' embeddings. These candidate pinyin embeddings include y_i, representing the correct pinyin, and $n_{i,1}, n_{i,2}, \dots, n_{i,k} (k \geq 1)$, representing the wrong candidate pinyins.

3.2. Method

Bidirectional LSTM: Long short-term memory (LSTM) [19], as an encoder, effectively extracts features of variable-length sequences. However, the unidirectional property of LSTM causes it to use only the context before the current time step t. BiLSTM [20] comprehends the forward and backward contexts to generate bidirectional contextual information, which is more meaningful. Based on the bidirectional context, BiLSTM consistently outperformed the unidirectional LSTM [2].

Negative sampling: Polyphone disambiguation is the task of selecting an appropriate pinyin among candidate pinyins for which a Chinese polyphone can be pronounced. The difference from typical classifications is that the candidate list is determined by each character; the candidates are not the entire class. The number of candidate pinyins of a polyphone is mostly two or three, and our method utilizes this to our advantage. We utilize the k-negative sampling strategy [15] for training, where k varies depending on the character. According to the context, one appropriate pinyin in the candidate list is used as a positive sample, whereas the remaining k candidate pinyins are used as negative samples. Negative sampling improved the model significantly by keeping the model's output embeddings away from negative label embeddings (the specific experimental details are described in Section 4).

Triplet Loss: We implement cosine similarity to evaluate the similarity between two embeddings. To train the model with k-negative samples, we define the loss function as the following equation:

$$L = 1 - \cos(e, y) + \frac{1}{k} \sum_{j=1}^{k} \max(0, \cos(e, n_j)) \qquad (1)$$

3.3. Training

Before training the model, candidate pinyin lists were constructed from the training and development dataset. The set of all pinyins that appear in the training and development dataset corresponding to a Chinese character become the character's candidate pinyins. In addition, to handle characters or pinyins that did not appear, we use the Chinese open-source dictionary CC-CEDICT[1] to collect extra character pinyin mapping information and reinforce the candidate pinyin lists.

During training, Chinese sentences were fed to the model. The BiLSTM encoder then generates character embeddings containing contextual information of the input sentences. The character embedding of a polyphone, positive pinyin embedding, and $k(k \geq 1)$ negative pinyin embeddings obtained through negative sampling comprise a training unit in the training process. The loss is calculated using the triplet loss, and the weights of the model, including the two embedding layers, are updated through backpropagation. The character embedding is learned to resemble the positive pinyin embedding and differ from the negative pinyin embeddings by repeatedly reducing the loss.

3.4. Inference

When a Chinese sentence is provided to the model during inference, for each character, if it is a monophone, the one-to-one mapping result is returned; if it is a polyphone, polyphone disambiguation is conducted. The character embedding by the encoder was compared to the embeddings of the candidate pinyins. Thereafter, the matching algorithm selects the most similar pinyin as the answer. There is no substantial load in the calculation, as we only have to compare few pinyins (mostly two or three) belonging to the candidate pinyin list of the polyphone without comparing it with all pinyins. All pinyins can be used iteratively once they are embedded. Finally, cosine similarity measures the degree of resemblance to the embeddings.

4. Experiments and results

4.1. Dataset

We evaluated the performance of the label embedding method using two datasets.

Data Baker Ltd Dataset: Data-Baker Science and Technology Ltd. provides an open benchmark dataset [2] for the Chinese polyphone disambiguation task. The dataset contains 10,000 samples, each consisting of a Chinese sentence and pinyins for each character. We selected 9,746 samples whose sentence length equal to the label length and split them into training, development, and test sets at a ratio of 8:1:1. There are 4,110 unique Chinese characters, of which 1,007 are unique polyphones. The ratio of polyphones is small, but 59% of all characters that appear in the dataset are polyphones. In addition, 1,534 unique pinyins appeared as labels.

[1]https://cc-cedict.org/wiki/

[2]https://www.data-baker.com/open#/data/index/source

Table 1: *Hyperparameters*

Hyperparameters	**One-hot Classifier**	**Label Embedding Classifier**
Char embedding size	300	128
Label embedding size	-	128
LSTM hidden size	64	300
Batch size	128	256

Our dataset: We constructed a Chinese G2P dataset by labeling from speech. The voice actor read a Chinese script naturally, and taggers dictated the pinyins exactly as they heard. The results of labeling could be different from the canonical pronunciations. This is mainly due to the tone of the pinyin, as native speakers do not necessarily follow a fixed tone.

The following examples are the pinyin candidate lists in the dictionary and our dataset for the same character. The 6th tone means cases in which 3rd tone was pronounced as 2nd tone. The examples show that a monophone can become a polyphone and a polyphone can have more candidate pinyins, according to our dataset construction method.

- 老, canonical: [lao3] ours: [lao3, lao6]
- 一, canonical: [yi1] ours: [yi1, yi2, yi4, yi5]
- 少, canonical: [shao3, shao4] ours: [shao3, shao4, shao5, shao6]

The entire dataset has a total of 1,968 unique pinyins and 4,662 unique Chinese characters. Among these Chinese characters, 31.6% of the characters are unique polyphones, whereas 68.4% are unique monophones. However, the appearance rate of polyphones is about twice that of monophones. G2P task is more challenging in our dataset because of the increased proportion of the polyphones and the increased number of candidate pinyins. However, our dataset is expected to train G2P models for natural speech synthesis. We randomly divided 45,558 sentences into training, development, and test sets at a ratio of 8:1:1.

4.2. Baseline and hyperparameters

We implemented the BiLSTM model [2, 3], which performs one-hot label classification as a baseline. The hyperparameters of baseline, one-hot classifier, and our proposed model, label embedding classifier, are defined in Table 1. Both models use a single-layer BiLSTM and two fully connected layers. The forward and backward outputs of BiLSTM are concatenated. The dropout rate is set to 0.1, and ReLU [21] is used as the activation function. We useds Adam optimizer [22] with a learning rate 1e-4 to train models.

4.3. Evaluation

The following metrics were used to evaluate the performance of the label embedding: polyphone accuracy (Poly Acc), character

Table 2: *Results on the Data Baker Ltd Dataset and our dataset.*

Dataset	Model	Poly Acc (%)	Char Acc (%)	Sent Acc (%)
Data Baker Ltd Dataset	One-hot Classifier	89.11	92.12	29.47
	Label Embedding Classifier	**93.61**	**95.34**	**47.54**
Our Dataset	One-hot Classifier	93.88	95.79	55.87
	Label Embedding Classifier	**95.62**	**96.99**	**65.64**
	Label Embedding Classifier w/o Negative Sampling	89.30	92.64	35.76

Table 3: *Results of the BERT models on our dataset.*

Model	Poly Acc (%)	Char Acc (%)	Sent Acc (%)
BERT One-hot Classifier	94.03	95.90	56.03
BERT Label Embedding Classifier	94.95	96.52	61.38
Label Embedding Classifier w/ BERT embedding	**95.98**	**97.23**	**67.66**
Label Embedding Classifier	95.62	96.99	65.64

accuracy (Char Acc), and sentence accuracy (Sent Acc). Polyphone accuracy is polyphone disambiguation accuracy, which indicates the proportion of correctly classified polyphones. Character accuracy represents the character unit G2P accuracy for all Chinese characters (monophones and polyphones). Sentence accuracy refers to the proportion of sentences in which all polyphones are accurately disambiguated.

4.4. Result

Table 2 summarizes the performance of two models on the Data Baker Ltd dataset and our dataset. The results demonstrate that the label embedding classifier outperformed the one-hot classifier in all metrics. It achieved 4.50%, 1.74% polyphone disambiguation improvements in both datasets compared to the one-hot classifier.

We estimate that the label embedding approach outperforms one-hot label classification as it concentrates only on candidate pinyins rather than calculating the scores for the entire pinyins. In addition, as shown in the last row of Table 2, negative sampling significantly improves label embedding performance by utilizing candidate pinyins. This is effective because of the characteristics of Chinese, which has a large number of pinyins (approximately 2,000), but a small number (mainly two or three) of pinyin candidates for each character.

5. Experiments using BERT

We applied the label embedding method to the pre-trained BERT [3] model to investigate the effect of contextual information in character embedding. BERT was used in three ways as shown in Table 3: BERT for one-hot label classification, BERT for label embedding, and BERT for additional character embedding. The BERT one-hot classifier and BERT label embedding classifier replaced the character embedding layer and BiLSTM of one-hot classifier and label embedding classifier with BERT respectively. Label embedding classifier with BERT embedding model, based on the BiLSTM, concatenated BERT embedding to its character embedding.

The BERT models were tested using our dataset. According to Table 3, the BERT label embedding classifier underperformed our BiLSTM label embedding classifier. We assume this is because BERT, which focuses on the meaning of the characters, exhibits weakness when different characters with the same meaning have different pinyins. Characters with the same meaning will have similar representations when using BERT, but may have completely different pinyins. Therefore, BiLSTM label embedding classifier, where character embeddings are distinct from each other and contextualized by BiLSTM, can outperform the BERT label embedding model. Rather than using BERT embedding as a direct character embedding, the best performance was achieved when using it as a feature that augmented contextual information.

[3]https://huggingface.co/bert-base-chinese

6. Conclusion

We proposed a novel label embedding approach for the Chinese grapheme-to-phoneme conversion task. We achieved 4.50% and 1.74% improvements in polyphone disambiguation accuracy on the two datasets compared to the one-hot label classification approach. This indicates that our label embedding approach with the negative sampling strategy is effective in the Chinese G2P task, which has a large pinyin class size (approximately 2,000), but a small candidate pinyin size (mainly two or three) for each character. In addition, our label embedding demonstrated a competitive performance that surpassed BERT. In future work, investigating methods to speed up inference of the label embedding approach might prove important for real-time TTS without delay.

7. References

[1] R. T.-H. Tsai and Y.-C. Wang, "A maximum entropy approach to chinese grapheme-to-phoneme conversion," in *2009 IEEE International Conference on Information Reuse & Integration*. IEEE, 2009, pp. 411–416.

[2] C. Shan, L. Xie, and K. Yao, "A bi-directional lstm approach for polyphone disambiguation in mandarin chinese," in *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2016, pp. 1–5.

[3] K. Park and S. Lee, "g2pm: A neural grapheme-to-phoneme conversion package for mandarinchinese based on a new open benchmark dataset," in *INTERSPEECH*, 2020.

[4] Z. Cai, Y. Yang, C. Zhang, X. Qin, and M. Li, "Polyphone disambiguation for mandarin chinese using conditional neural network with multi-level embedding features," in *INTERSPEECH*, 2019.

[5] D. Dai, Z. Wu, S. Kang, X. Wu, J. Jia, D. Su, D. Yu, and H. Meng, "Disambiguation of chinese polyphones in an end-to-end framework with semantic features extracted by pre-trained bert," in *INTERSPEECH*, 2019.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[7] Z. Min and C. Lian-hong, "A new rule-based method of automatic phonetic notation on polyphones," *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.*, vol. 1, pp. 671–674 vol.1, 2004.

[8] 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.

[9] F. 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.

[10] X. Mao, Y. Dong, 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'07*, vol. 4. IEEE, 2007, pp. IV–705.

[11] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint embedding of words and labels for text classification," *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 2321–2331, 2018.

[12] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 1425–1438, 2016.

[13] J. A. Rodríguez-Serrano and F. Perronnin, "Label embedding for text recognition," in *BMVC*, 2013.

[14] H. Zhang, L. Xiao, W. Chen, Y. Wang, and Y. Jin, "Multi-task label embedding for text classification," in *EMNLP*, 2018.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[16] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "Starspace: Embed all the things!" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[17] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *NeurIPS*, 2019.

[18] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1386–1393.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[20] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, pp. 2673–2681, 1997.

[21] A. F. Agarap, "Deep learning using rectified linear units (relu)," *ArXiv*, vol. abs/1803.08375, 2018.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.