INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

CVTE-Poly: A New Benchmark for Chinese Polyphone Disambiguation

Siheng Zhang[1,2], *Xingjun Tan*[2], *Yanqiang Lei*[2], *Xianxiang Wang*[2], *Zhizhong Zhang*[2*], *Yuan Xie*[2]

[1]Guangzhou Shiyuan Electronic Technology Company Limited, China
[2]School of Comuter Science, East China Normal University, China

{zhangsiheng,tanxingjun,wangxianxiang,leiyanqiang}@cvte.com,
{zzzhang,yxie}@cs.ecnu.edu.cn

Abstract

Conversion from graphemes to phonemes is an essential component in Text-To-Speech systems, and in Chinese, one main challenge is polyphone disambiguation—to determine the pronunciation of characters with multiple pronunciations. In this task, the benchmark dataset **C**hinese **P**olyphone disambiguation with **P**inyin (CPP) suffers from two main limitations: Firstly, it contains some wrong labels in contrast to the newest official dictionary. Secondly, it is imbalanced and hence models learned from it show a learning bias towards frequently-used pronunciations and polyphones.

In this paper, we refine CPP and release a new dataset named **CVTE-poly**, containing 845254 samples, nearly ten times the size of CPP and is more balanced. Besides, we propose a comprehensive measurement for polyphone disambiguation task, against the data imbalance problem. Experiments show that our simple but flexible baseline trained on CVTE-poly outperforms existing models, which demonstrate the benefit of our dataset.

Index Terms: text-to-speech, Chinese graphemes to phonemes, polyphone disambiguation, deep learning

1. Introduction

Chinese characters represent the meanings but not the sounds, and hence in order to pronounce a Chinese character in a Text-To-Speech (TTS) system, it is essential to use graphemes to phonemes (G2P) conversion which transforms a character into ‘Pinyin’ (the Romanization system of Chinese). The major challenge in Chinese G2P conversion is how to disambiguate the pronunciation of polyphones—characters having different pronunciations according to their semantic and syntactic usage within context.

There have been many academic efforts tackling this task. Park’s work [1] was the first to release a benchmark dataset called CPP (Chinese Polyphone disambiguation with Pinyin) with train/dev/test split. Since then, researchers followed this dataset[2, 3]. However, there are some problems with CPP, which turn out to be the main restrictions of polyphone disambiguation. The problems can be summarized as follow:

- First, the CPP dataset does not strictly obey the 7th edition of ‘Modern Chinese Dictionary’ (MCD-7) [4], which is the newest version of the official Chinese dictionary. For example, the CPP dataset labels the character ‘会’ in ‘会稽’ (a city name) as ‘hui4’, but the correct label is ‘kuai4’. Besides, language develops over time, and some pronunciations also change. For example, the character ‘骑’ used to have two pronunciations,‘qi2’ (ride in English) and ‘ji4’ (rider in English). But MCD-7 only preserves the first one.
- Second, the CPP dataset is not comprehensive. It does not contain some frequently-used polyphones such as ‘壳’ . Besides, some polyphone has only one pronunciation in the dataset. Take the character ‘识’ (‘shi2’ or ‘zhi4’) as an example. CPP contains 200 sentences regarding with ‘shi2’ but none for ‘zhi4’. This causes models to have what we called **learning bias** and hence are not applicable.
- Third, the data imbalance problem naturally exists in Chinese, including character imbalance and pronunciation imbalance. This problem results in **over-estimation** of models. There are some work addressing on this problem [5, 3, 6]. However, this field needs a more balanced dataset, together with fairer quantitative metrics against data imbalance.

In this paper, we release a new dataset called **CVTE-poly** with a larger corpus and more balanced labels, as well as refinements to CPP including wrong label corrections and non-polyphone removals. Besides, instead of complicated models, we propose a simple baseline model which can be flexibly assembled with tokenization and Part-Of-Speech (POS) tagging. The contributions of this work are threefold:

- We improve the benchmark dataset both in quality and in quantity. CVTE-poly is about ten times the size of CPP, covering more polyphones and pronunciations. Besides, CVTE-poly has a more balanced label distribution.
- We propose a more comprehensive measurement to avoid over-estimation for polyphone disambiguation task.
- Experiments show that with CVTE-poly, simple baseline models outperform the state-of-the-art, demonstrating the effectiveness of the new dataset.

All codes and data are available to the public [1].

2. Related Work

Early stage work on Chinese polyphone disambiguation included rules-based models [7, 8], which require large amount of linguistic knowledge and hence are not applicable. Later, researchers used statistical models like Maximum Entropy [9] and LSTM [10].

However, there was no public benchmark until Park’s work [1]. It released the benchmark dataset CPP and proposed g2pM model base on bi-directional LSTM. In g2pW [2], the authors used weighted softmax to concentrate on candidate pronunciations. This work also designed a conditional weight layer to learn auxiliary POS tagging task. Similarly, Zhang’s work [3] used weighted softmax as well as focal loss to reduce learning bias. Li’s work [11] uses a mix-pooling mechanism to get richer semantic representation for a character.

[1]https://github.com/NewZsh/polyphone

In this paper, we argue that it is more urgent to pay attention to **learning bias** caused by uneven distributions of polyphones and pronunciations, instead of designing complicated models. The learning bias is due to two reasons: First, Chinese characters' occurrence forms a long-tailed distribution[12, 7]. Take the character '行' as an example. It has two pronunciations 'xing2' and 'hang2', and has rich meanings is Chinese; Second, it is also observed that many polyphonic characters take one pronunciation as an overwhelming majority [6]. Take character '遂' as an example. It only pronounces 'sui2' in word '半身不遂' (means hemiplegia in English), and pronounces 'sui4' in other usages.

There is some work aiming at this problem from different perspectives, including data augmentation and weighted sampling [5], distant supervision [6] and so on. However, these work did not come up with a quantitative measurement for learning bias.

3. CVTE-poly Dataset

In this section, we describe the refinements to CPP dataset, as well as our dataset **CVTE-poly**.

3.1. Refinements to CPP

The problematic samples in CPP fall into two categories:

- Wrong labels. For the testing set, we list the 70 wrongly labelled sentences and the reasons to correct them. For the training and validation sets, we list the corrections in several categories.
- Non-polyphone characters. In the testing set, there are 1319 sentences which take a non-polyphone as a target character. These sentences should be removed. In the training and validation sets, we also delete such sentences.

The refined CPP dataset contains 69094, 8640 and 8935 sentences for the training, validation and testing sets respectively. From here on, the CPP dataset refers to the refined one.

3.2. CVTE-poly

To collect CVTE-poly, here is our main pipeline:

1 Collect all polyphones from MCD-7. Then, collect all words containing the polyphones with the pronunciations [2]. There are 612 polyphones in total. Since some of them are rare and do no form words, the result here is 19755 words, covering 425 polyphones with different pronunciations;

2 Crawl the webpage of each word from 'Baidu Baike' [3], preserve the sentences that contain the target word. As some words do no appear in 'Baidu Baike', the result covers 403 polyphones with different pronunciations (see Table 1);

3 Manually confirm some labels. The polyphones in a word sometimes have more than one pronunciations. For example, the character '行' in the word '同行' can be pronounced as 'xing2' (travelling together) or 'hang2' (peer). For these words, we check the sentences and labels manually.

Note that although CPP contain 540 polyphones, just 344 of them appear with more than one pronunciation, and the samples with the remaining 196 of them are biased for training. Table 1 also shows that the size of CVTE-poly is nearly ten times that of CPP. In short, CVTE-poly enriches the **size** and **diversity** to a large extent.

[2]This can be done from the electronic copy: https://github.com/CNMan/XDHYCD7th/blob/master/XDHYCD7th.txt. Note that the uploaded version is crowd-sourced and hence has some errors.

[3]https://baike.baidu.com/, a Chinese wiki-like website.

Table 1: *Number of sentences and polyphones: CPP v.s. CVTE-poly*

	#sentences	#poly	#poly with >1 labels
1. CPP	86629	540	344
2. CVTE-poly	845254	414	403
1+2	931883	552	429

To study the balance property, we sum up the counts of majority pronunciation of each polyphone and the counts of minorities. As shown in Table 2, in CPP, the ratio between them is nearly 10 *v.s.* 1, while that of CVTE-poly is about 6.66 *v.s.* 1.

Table 2: *Sum of pronunciations counts: majority v.s. minority*

	#majority	#minority	ratio
1. CPP	78821	7808	10:1
2. CVTE-poly	817300	122695	6.66:1
1+2	893173	133451	6.69:1

4. Baseline configuration

In this work, we use a very simple model by adding just a fully-connected layer and softmax layer after the pre-trained language models, which have been demonstrated to be effective for many down-stream tasks, including language understanding and generation. Here, we choose ERNIE [13]. It is pre-trained on large scale of Chinese corpus and has a larger vocabulary for Chinese characters compared with BERT [14].

Note that the context is only used to push the characters through ERNIE and get the context-sensitive embedding of the polyphone. Denote the embedding as $x = x_{\text{ernie}}$, our model then predicts the label distribution p by

$$p = \text{softmax}(Wx + b)$$

in which W, b are the weights and biases of fully-connected layer.

During training, we use cross entropy to measure the loss between predicted label distribution p and the one-hot encoding of the label y:

$$L = -\sum_{i=1}^{C} y_i \log p_i$$

in which C represents the total number of all pronunciations collected from all polyphones.

During inference, to restrict a polyphone to choose from its candidate pronunciations, we use a hard mask $m = [m_1, \cdots, m_C]$ with all zeros except for the candidates to be ones. Then we determine the final pronunciation according to the maximum of $p \odot m$, in which $\odot$ represents element-wise multiplication.

4.1. Tokenization and POS tagging

It has been demonstrated that POS tagging can serve as auxiliary supervised learning task to improve the accuracy of polyphone disambiguation [2]. In our model, we use it as part of feature by concatenation. This allows our model to flexibly choose whether to assemble this feature or not.

Besides, we observe that tokenization can be a reflection of the semantic or syntactic usage of a character, and hence affects pronunciations. For example, in the sentence '如何/学会/计算机' (it means 'how to master information technology' in English, we use '/' to represent the tokenization), '会' is pronounced as 'hui4', and in the sentence '他/是/学/会计/的' (it means 'he studies accounting'), '会' is pronounced as 'kuai4'. The different tokenizations affect the pronunciations.

Tokenization can be represented as sequential labelling by 'B' (**B**egin of a word), 'M' (**M**iddle of a word), 'E' (**E**nd of a word) and 'S' (**S**ingle character as a word). The labelling can also be transformed into a vector by embeddings. For each character, we denote its vector for tokenization information as x_{token}. Then we concat it with the character vector before the final prediction, i.e. $x = [x_{\text{ernie}}; x_{\text{token}}]$.

After tokenization, POS tagging assigns a POS tag for each word. For each character, we assume that its tag inherits from the word it belongs to. The labelling can also be transformed into a vector by embeddings. For each character, we denote its vector for POS tagging information as x_{pos}. Then we concat the character vector before final prediction $x = [x_{\text{ernie}}; x_{\text{token}}; x_{\text{pos}}]$.

5. Experiments

5.1. Evaluation protocol

We use CVTE-poly as part of training, and evaluate models on the CPP testing set.

As shown in Table 1, 196 polyphones appears in CPP dataset with just one pronunciation. We extract the subset which does not contain these polyphones from the CPP testing set , naming '**test(small)**'. Evaluations conducted on it ensure that models will not learn the data set bias. We use accuracy averaged in three ways as evaluation metrics (it is similar to but more comprehensive than f1-score):

- *Acc.*, overall accuracy averaged by cases;
- *Acc. avg.p*, accuracy averaged by each polyphone. As mentioned above, some characters are commonly used, this metric can help us to understand whether the model biases toward some characters;
- *Acc. avg.pp*, accuracy averaged by each polyphone and pronunciation. As mentioned above, some polyphones have dominated pronunciation, this metric can help to check whether the model biases on some pronunciations.

5.2. Comparison models

To ensure a fair comparison with SOTA models, we choose two with official released code: g2pW [2] [4] and g2pM [1] [5].

Note that by adopting the multi-task learning framework, g2pW aims to fit pronunciation and POS tagging together. Hence, its training requires POS tag labels, which is annotated manually on the CPP dataset according to its original paper. When training g2pW on the CPP dataset only, we use its POS tag labels. When training g2pW on CVTE-poly together with CPP training set, we use the POS tags predicted by 'jieba' (a python NLP toolkit)[6] as labels.

For g2pM, we train it with 100 epochs, with learning rate 1e-5, and 1 hidden layer with hidden size 128 for bi-directional LSTM. For g2pW, we train it with 100 epochs, with learning rate 5e-5 and the weight of loss from POS tagging being 0.1, as suggested in the original papers.

[4]https://github.com/GitYCC/g2pW/

[5]https://github.com/kakaobrain/g2pM/

[6]https://pypi.org/project/jieba/

5.3. Training protocol

The ERNIE-3.0 series have several models with difference sizes of parameters. Here we choose two models for Chinese to verify our methods: the base model 'ERNIE-3.0-base-zh' (118 million parameters in total) and the nano model 'ERNIE-3.0-nano-zh' (18 million parameters in total) [7]. In the nano model, word embeddings are 312-dimensional, and we set the embeddings of tokenization and POS tagging to the same shape. In ERNIE-base, this dimension becomes 768.

For both models, we train 100 epochs with a learning rate of $5e^{-5}$, and evaluate on the CPP validation set after every epoch to choose the best one for testing. For ERNIE-nano, we set the batch size to be 300, while for ERNIE-base, due to the GPU memory constraint, we set the batch size to be 64.

We choose 'jieba' for tokenization and POS tagging. The experiments are conducted on a machine with single 12G Titan Xp GPU using PyTorch.

5.4. Experiment results and analysis

Table 3: *Comparison with SOTA and our models (trained only on CPP, best result is in bold, second best is with underline)*

	CPP test			CPP test(small)		
	Acc.	Acc. *avg.p*	Acc. *avg.pp*	Acc.	Acc. *avg.p*	Acc. *avg.pp*
g2pM	.9728	.9652	.9011	.9580	.9453	.8658
g2pW	**.9897**	<u>.9845</u>	<u>.9577</u>	**.9841**	<u>.9757</u>	<u>.9426</u>
our	<u>.9891</u>	**.9854**	**.9596**	<u>.9834</u>	**.9786**	**.9461**

Table 4: *Comparison with SOTA and our models (trained with CVTE-poly, best result is in bold, second best is with underline)*

	CPP test			CPP test(small)		
	Acc.	Acc. *avg.p*	Acc. *avg.pp*	Acc.	Acc. *avg.p*	Acc. *avg.pp*
g2pM	.9780	.9673	.9232	.9621	.9521	.9196
g2pW	**.9890**	**.9842**	<u>.9664</u>	<u>.9833</u>	**.9755**	<u>.9545</u>
our	**.9890**	<u>.9837</u>	**.9670**	**.9841**	**.9755**	**.9559**

Table 3 and 4 summarize the experiment results of SOTA together with our best results. From the tables, our baseline models, though very simple, achieve comparable or even higher performance compared with SOTA. More specifically,

- **Compare the performance cross the two tables**, all models trained with CVTE-poly show higher *Acc. avg.pp* compared with only trained on CPP. This demonstrates the effectiveness of CVTE-poly. On the other two metrics, training with CVTE-poly and only on CPP show comparable performance. We suggest that the reason is the performance bottleneck of each model on the CPP testing set has been reached.
- **Compare the performance of three models in two tables**, our model shows highest *Acc. avg.p* and *Acc. avg.pp*, both trained only on CPP and with CVTE-poly, though our model shows a negligible decrease in the overall accuracy compared

[7]Official repository of ERNIE 3.0: https://paddlenlp.readthedocs.io/zh/latest/model_zoo/transformers/ERNIE/contents.html

with g2pW. This demonstrates our proposed model is more balanced.

- **Compare the performance tested on test and test(small)**, we can see that metrics on the former are all higher than that on the latter. This verifies that the samples not in test(small) cause the performance to be over-estimated. It is worth mentioning that all models trained with CVTE-poly also show the same pattern. This indicates that training with samples with high diversity will not harm the models' performance on the highly-biased polyphones, which can also be a strong evidence for the effectiveness of CVTE-poly.

5.5. Ablation study and analysis

To fully understand whether using tokenization and POS tagging information does make an effect on polyphone disambiguation, we conduct an ablation study on only CPP and with CVTE-poly, with models of different capacities: ERNIE-base and ERNIE-nano. The results are summarized in tables 5 and 6, which reveal the following interesting findings:

- **CVTE-poly shows a substantial help for both improving accuracy and reducing bias.** Comparing across Table 5 and Table 6, with CVTE-poly, both models show great improvements, especially on the *Acc. avg.pp* (ERNIE-nano model improves from 94.80% to 95.88% on the whole testing set, from 92.95% to 94.42% on the small testing set, and ERNIE-base model improves from 95.96% to 96.70% and 94.61% to 95.59%).
- **Only using tokenization brings no improvement.** From the tables, we can not conclude that only using tokenization embeddings brings improvement. Refer to Table 5, ERNIE-nano trained only with CPP benefits a bit from tokenization. However, on other cases, model assembled with only tokenization get even a bit worse. We suggest that the reason is that tokenzation is too correlated to pronunciations and hence cannot provide extra information for learning.
- **POS Tagging helps improving accuracy and reducing bias.** This improvement is obvious especially on smaller language model with fewer training samples (see Table 5 for ERNIE-nano trained only on CPP). Besides, with tokenization and POS tagging embeddings together, *Acc. avg.pp* has greater promotion than the overall accuracy. This indicates that utilizing tokenization and POS tagging information can reduce the bias for polyphone disambiguation task. It is also worth mentioning that external tools for tokenzition or POS tagging are not perfect, which indicates the potential of the models against some noise.
- **Models with higher capacity enjoy higher accuracy with lower bias.** From each table, we can see that ERNIE-base leads ERNIE-nano over all metrics, even when ERNIE-base trained only on CPP without extra information (see line 5 in Table 5) is comparable with ERNIE-nano trained with CVTE-poly with both tokenization and POS taggings. Besides, using tokenization and POS tagging in ERNIE-base just boost the performance a bit, while in ERNIE-nano, the improvement is much higher. This phenomenon suggests that large model tends to learn the underlying semantic and syntactic of language.
- **Small models also can be effective in applications.** With CVTE-poly, and with tokenzation and POS taggings, we can see that ERNIE-nano shows comparable performance to g2pW as well as our best model. Note that since ERNIE-nano costs nearly 1/3 of the memory of ERNIE-base and enjoys half of time in inference, it is an applicable choice when the actual application requires low time or memory cost.

Table 5: *Ablation study on models just trained on CPP (best result is in bold)*

		CPP test			CPP test(small)		
		Acc.	Acc. *avg.p*	Acc. *avg.pp*	Acc.	Acc. *avg.p*	Acc. *avg.pp*
nano	-	.9837	.9758	.9344	.9747	.9620	.9110
	+token	.9839	.9787	.9350	.9751	.9666	.9117
	+POS	.9860	.9798	.9455	.9784	.9683	.9260
	+token,POS	.9864	.9818	.9480	.9789	.9715	.9295
base	-	.9887	.9853	.9571	.9831	.9773	.9420
	+token	.9880	.9845	.9547	.9825	.9765	.9400
	+POS	**.9891**	.9847	.9578	.9828	.9776	.9433
	+token,POS	**.9891**	**.9854**	**.9596**	**.9834**	**.9786**	**.9461**

Table 6: *Ablation study on models trained with CVTE-poly (best result is in bold)*

		CPP test			CPP test(small)		
		Acc.	Acc. *avg.p*	Acc. *avg.pp*	Acc.	Acc. *avg.p*	Acc. *avg.pp*
nano	-	.9867	.9795	.9531	.9796	.9680	.9364
	+token	.9846	.9811	.9506	.9761	.9703	.9329
	+POS	.9871	.9819	.9535	.9761	.9703	.9329
	+token,POS	.9877	.9834	.9588	.9810	.9740	.9442
base	-	.9880	.9804	.9581	.9818	.9695	.9434
	+token	.9871	.9791	.9567	.9810	.9690	.9418
	+POS	**.9891**	.9819	.9613	.9820	.9720	.9517
	+token,POS	.9890	**.9837**	**.9670**	**.9841**	**.9755**	**.9559**

6. Conclusions

This work releases a new benchmark '**CVTE-poly**' for Chinese polyphone disambiguation . The new dataset is better than CPP in all perspectives, including size, diversity and balance. Besides, this work proposes a comprehensive measurement to address the learning bias problem. Experiments show that simple baseline models can outperform SOTA models with the help of CVTE-poly, which demonstrate the effectiveness of our new dataset. Last but not the least, this work makes detailed comparisons with models of different capacity, as well as the effect of extra syntactic features.

Future work in Chinese polyphone disambiguation contains the following: (1) predict all polyphones in a sentence at once by modelling the latent connections among them. For example, word '重创' can pronounce as 'zhong4 chuang1' (means 'huge damage' in English) or 'chong2 chuang4' (means 're-establish' in English). In this example, both characters are polyphones, and there exists a strong reliance between their pronunciations; (2) ancient Chinese texts have a phenomenon called adulteration, and some idioms or expressions coming from ancient Chinese are still used today. Hence, the character may have special pronunciations in these cases.

7. Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 62006139).

8. References

[1] K. Park and S. Lee, "g2pM: A Neural Grapheme-to-Phoneme Conversion Package for Mandarin Chinese Based on a New Open Benchmark Dataset," in *Proc. Interspeech*, 2020, pp. 1723–1727.

[2] Y. C. Chen, Y. C. Steven, Y. C. Chang, and Y. R. Yeh, "g2pW: A Conditional Weighted Softmax BERT for Polyphone Disambiguation in Mandarin," in *Proc. Interspeech*, 2022, pp. 1926–1930.

[3] H. Zhang, H. Pan, and X. Li, "A Mask-Based Model for Mandarin Chinese Polyphone Disambiguation," in *Proc. Interspeech*, 2020, pp. 1728–1732.

[4] Dictionary Editorial Office of Institute of Language, Chinese Academy of Social Sciences, *Modern Chinese Dictionary (7th Edition)*. The Commercial Press, 2016.

[5] Y. Zhang, H. Zhang, and Y. Lin, "Data augmentation for long-tailed and imbalanced polyphone disambiguation in mandarin," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7137–7141.

[6] J. Zhang, Y. Zhao, J. Zhu, and J. Xiao, "Distant Supervision for Polyphone Disambiguation in Mandarin Chinese," in *Proc. Interspeech*, 2020, pp. 1753–1757.

[7] Z.-R. Zhang, M. Chu, and E. Chang, "An efficient way to learn rules for grapheme-to-phoneme conversion in Chinese," in *Proc. International Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2002, pp. 59–63.

[8] F.-L. Huang, "Disambiguating effectively chinese polyphonic ambiguity based on unify approach," in *International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 6, 2008, pp. 3242–3246.

[9] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, 2007, pp. IV–705–IV–708.

[10] 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 *Proc. Interspeech*, 2019, pp. 2110–2114.

[11] J. Li, Z. Zhang, M. Chen, J. Ma, S. Wang, and J. Xiao, "Improving Polyphone Disambiguation for Mandarin Chinese by Combining Mix-Pooling Strategy and Window-Based Attention," in *Proc. Interspeech*, 2021, pp. 4104–4108.

[12] L. Xie, H. Fang, and Y. Jin, "Statistical analysis for standard chinese common-words, syllables and phoneme system," *Journal of Northwest University for Nationalities*, vol. 51, no. 11, pp. 35–39, 2012.

[13] Y. Sun, S. Wang, and et al., "Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation." *arXiv preprint*, no. arXiv:2107.02137, 2021.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova., "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint*, no. arXiv:1810.04805, 2018.