# Appendices

## A Details of Models

In this section, we describe details about the dictionary construction, the architecture of the semantic encoder and linguistic encoder, and multi-length discriminator. For further information about the variational generator, one could refer to PortaSpeech [41]. We also describe our modifications to model architecture to support the pre-training task of Dict-TTS with large-scaled ASR datasets.

### A.1 Dictionary Construction

The Chinese dictionary used in the experiments is obtained from `https://github.com/yihui/zdict`, which is crawled from `https://www.zdic.net/zd/zb/ty/`. The Japanese dictionary is crawled from `https://dictionary.goo.ne.jp` and the Cantonese dictionary is crawled from `https://humanum.arts.cuhk.edu.hk/Lexis/lexi-can`. The crawled dictionaries are processed into the format described in SubSection 3.3.

We use a pre-trained cross-lingual language model [10] to extract the semantic context information $k$ of every gloss entry $e$ and store them in the disk for computational efficiency[8]. Note that our Dict-TTS does not need the heavy BERT model in the inference stage. As is pointed out by the previous works, lower layers of BERT are found to perform broad attention across all pairs of words or encode local syntax [9, 39] and middle layers are found to mostly capture transferable syntactic and semantic knowledge [17, 51]. But the upper layers are specifically tuned towards the pre-training tasks of BERT [56]. Thus, we use the average of the first-layer word embeddings and the later 8-layer contextual BERT representations as the extracted knowledge.

### A.2 Architecture of the Semantic Encoder and Linguistic Encoder

As shown in Figure 6, the semantic encoder and linguistic encoder of our Dict-TTS are both stacks of feed-forward Transformer layers with relative position encoding [44] following PortaSpeech [41]. And there are four layers of the semantic encoder and four layers of the linguistic encoder in our Dict-TTS.

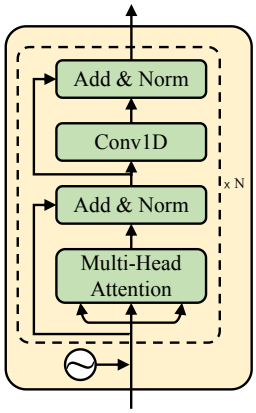



Figure 6: The detailed architecture of the semantic encoder and linguistic encoder.

### A.3 Modifications for Pre-training Task

The vanilla architecture of PortaSpeech is tested on LJSpeech (a single-speaker dataset). By introducing a group of learnable speaker embeddings to represent the speakers' timbre, pronunciation habits, and other features, it can be trained on multi-speaker datasets. However, since large-scaled ASR datasets may not have explicit speaker information, the following modifications should be made: we extract the speaker embeddings from audio samples using resemblyzer[9] and feed them into the variational generator and duration predictor. We pre-train our DictTTS on the WenetSpeech dataset [61], which is a multi-domain corpus for speech recognition. It takes 600k steps for pre-training until convergence. As is shown in Table 2, the semantic comprehension and the generalization capacity of our Dict-TTS are significantly improved by our pre-training process.

## B Detailed Experimental Settings

In this section, we describe more model configurations and details in subjective evaluation.

[8]The model used in our experiments can be downloaded from `https://huggingface.co/xlm-roberta-base`. One could use monolingual BERT to enhance the performance.

[9]`https://github.com/resemble-ai/Resemblyzer`

### B.1 Dict-TTS Model Configurations

We list the model hyper-parameters of Dict-TTS in Table 5.

Table 5: Hyperparameters of Dict-TTS models.

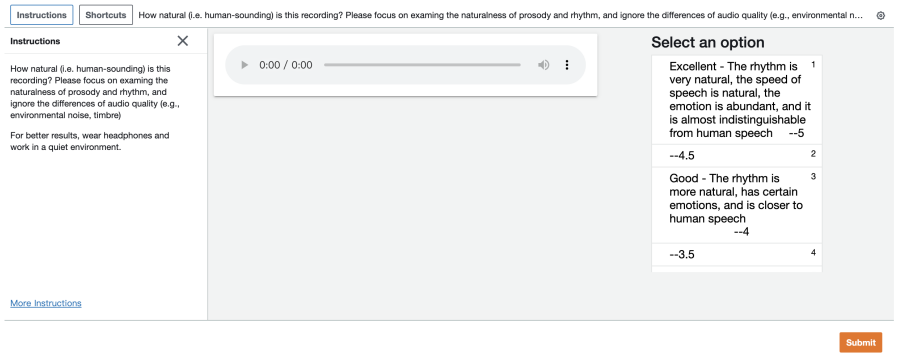
| Hyper-parameter | | Dict-TTS | Number of parameters |
|---|---|---|---|
| Semantic Encoder | Character Embedding | 192 | 4.907M |
| | Semantic Encoder Layers | 4 | |
| | Hidden Size | 192 | |
| | Conv1D Kernel | 5 | |
| | Conv1D Filter Size | 768 | |
| S2PA Module | Hidden Size | 192 | 0.404M |
| Linguistic Encoder | Linguistic Encoder Layers | 4 | 3.371M |
| | Hidden Size | 192 | |
| | Conv1D Kernel | 5 | |
| | Conv1D Filter Size | 768 | |
| Variational Generator | Encoder Layers | 8 | 7.516M |
| | Decoder Layers | 4 | |
| | Encoder/Decoder Kernel | 5 | |
| | Encoder/Decoder channel size | 192 | |
| | Latent Size | 16 | |
| | Prior Flow Layers | 4 | |
| | Prior Flow Conv1D Kernel | 3 | |
| | Prior Flow Conv1D Channel Size | 64 | |
| Multi-Length Discriminator | Number of Discriminators | 3 | 0.927M |
| | Window Size | 32, 64, 128 | |
| | Conv2D Layers | 3 | |
| | Hidden Size | 192 | |
| Total Number of Parameters | | | 17.125M |

### B.2 Baseline Model Configurations

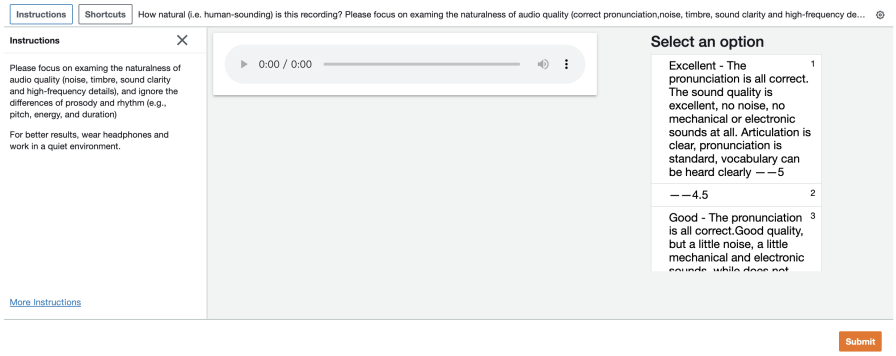
The baseline systems in our experiments can be divided into: 1) phoneme-based systems; 2) character-based systems. For phoneme-based systems, we use the mixture alignment proposed in PortaSpeech [41], which takes phoneme sequence as inputs but utilize both soft phoneme-level and hard word-level duration for mixture alignment. We use a 4-layer phoneme encoder and a 4-layer word encoder following the implementation of PortaSpeech [41]. For character-based systems, we use an 8-layer character encoder for fair comparisons. The encoders above are stacks of feed-forward Transformer layers with relative position encoding [44] following PortaSpeech [41]. Other parts of the architecture are kept the same as Dict-TTS to exclude other interference factors.

### B.3 Details in Subjective Evaluation

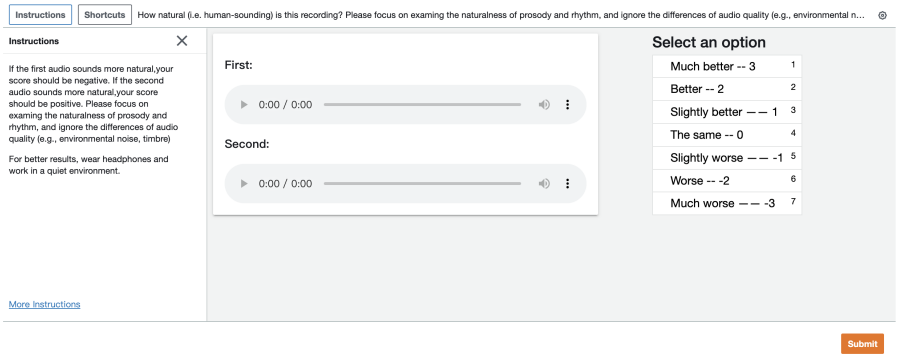
**Audio Quality and Prosody** We perform the audio quality and prosody evaluation on Amazon Mechanical Turk (MTurk). For each dataset, we randomly select 50 texts from the test set and use the TTS systems to generate the audio samples. Each audio has been listened to by at least 20 listeners. For MOS, each tester is asked to evaluate the subjective naturalness of a sentence on a 1-5 Likert scale. For CMOS, listeners are asked to compare pairs of audio generated by systems A and B, indicate which of the two audio they prefer, and choose one of the following scores: 0 indicating no difference, 1 indicating a slight difference, 2 indicating a significant difference and 3 indicating a very large difference. For audio quality evaluation (MOS-Q and CMOS-Q), we tell listeners to "*focus on examining the naturalness of audio quality (e.g., noise, timbre, sound clarity, and high-frequency details), and ignore the differences of prosody and rhythm (e.g., pitch, energy, and duration)*". For prosody evaluations (MOS-P and CMOS-P), we tell listeners to "*focus on examining the naturalness of prosody and rhythm (e.g., pitch, energy, and duration), and ignore the differences in audio quality (e.g., noise, timbre, sound clarity, and high-frequency details)*". The screenshots of instructions for



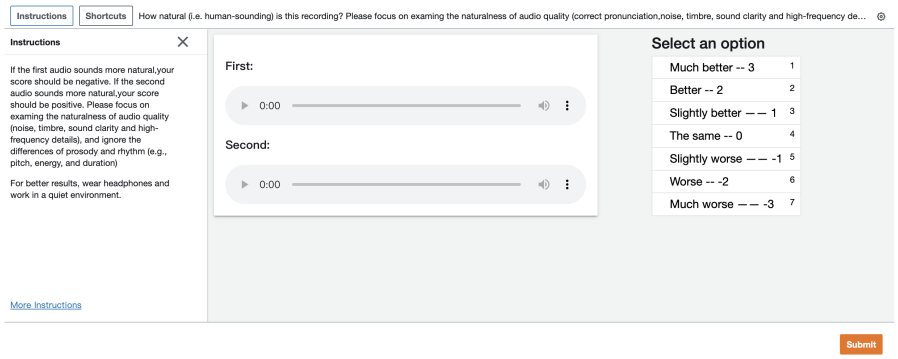

(a) Screenshot of MOS-P testing.




(b) Screenshot of MOS-Q testing.




(c) Screenshot of CMOS-P testing.




(d) Screenshot of CMOS-Q testing.

Figure 7: Screenshots of audio quality and prosody evaluations.

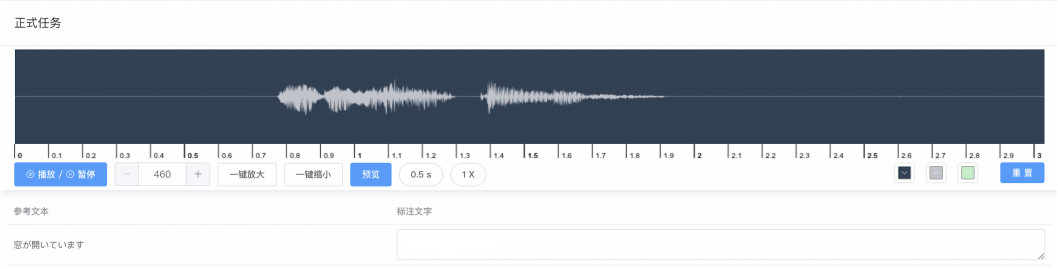



Figure 8: Screenshot of pronunciation accuracy evaluation.

testers are shown in Figure 7. We paid $8 to participants hourly and totally spent about $500 on participant compensation.

**Pronunciation Accuracy** We perform the pronunciation accuracy evaluation on MolarData[10]. For each dataset, we use all texts in the test set to generate the audio samples. Each audio has been listened to by at least 4 language experts. Each tester is asked to carefully listen to the audio multiple times, write down the mispronounced phonemes, and discuss with each other until a conclusion is reached. The screenshots of instructions for testers are shown in Figure 8. We paid $15 to participants hourly and totally spent about $800 on participant compensation. A small subset of speech samples used in the test is available at `https://dicttts.github.io/DictTTS-Demo/`.

### B.4 Details of the G2P tools used in the experiments

We use *pypinyin (0.46.0)*[11] and G2PM[12] in Biaobei dataset. Until 20 May, 2022, the latest version of pypinyin is 0.46.0. And it is worth noting that the pronunciation accuracy is greatly improved when we update the version from 0.36.0 to 0.46.0. In JSUT dataset, we use *pyopenjtalk (0.2.0)*[13]. And in Common Voice (HK) dataset, we use *pycantonese (3.4.0)*[14].

### B.5 Details of Datasets

In this subsection, we mainly describe the distribution characteristics of polyphones in the datasets used in our experiments, including: 1) Biaobei [3]; 2) JSUT [47] and 3) Common Voice (HK) [1]. As shown in Figure 9, we calculate the number of polyphones in every sentence. The average sentence lengths of the Biaobei, JSUT, and Common Voice (HK) datasets are 18, 29, and 10. Among these datasets, the most frequent number of polyphones appearing in a sentence is three to five and some sentences even include eight (or more) polyphones, indicating the importance of polyphone disambiguation.

### B.6 Error Bars and Random Seeds

For the experiments of the audio quality and prosody, we report confidence intervals of subjective metric results in Table 3. For the experiments of the pronunciation accuracy, we ran the experiments 10 times with 10 different random seeds ($[1234, 1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, 9999]$) and obtained the averaged results.

## C Visualization of Attention Weights

We put some semantics attention visualizations in Figure 10. We can see that Dict-TTS can create reasonable text-to-dictionary alignments, which improves the performance of polyphone disambiguation and helps the semantics comprehension for end-to-end TTS systems.

[10] `https://www.molardata.com/en.html`
[11] `https://github.com/mozillazg/python-pinyin`
[12] `https://github.com/kakaobrain/g2pM`
[13] `https://github.com/r9y9/pyopenjtalk`
[14] `https://github.com/jacksonllee/pycantonese`

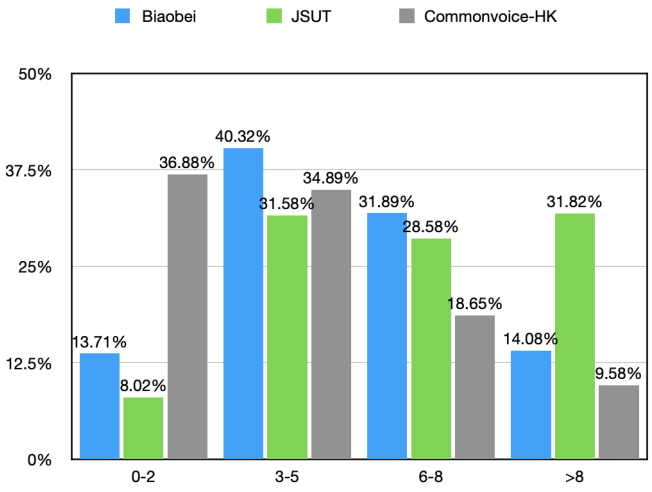
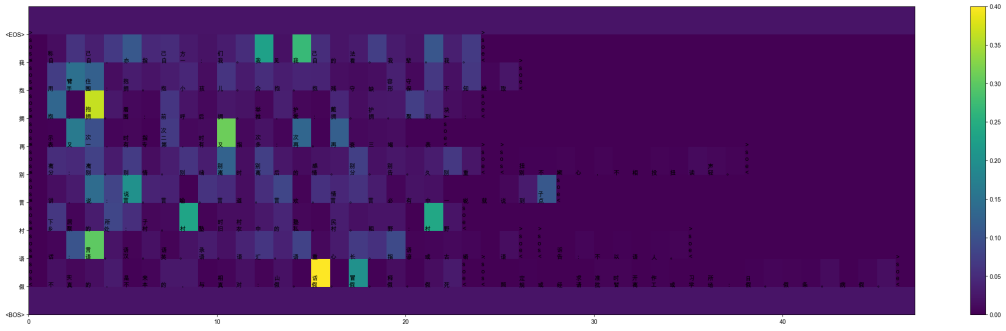
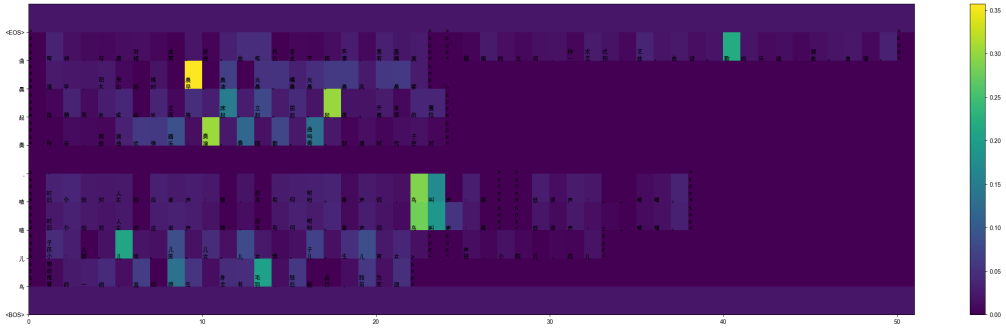



Figure 9: The illustration of the number of polyphone in a sentence. The horizontal axis is the frequency of polyphones in a sentence and the vertical axis is the number of sentences shown by percentage.



(a) Biaobei-000001



(b) Biaobei-000002

Figure 10: Visualizations of the semantics attention weights in Biaobei dataset.

## D The Definition of Polyphone and Heteronym

Polyphones are letters or characters having more than one phonetic value, while heteronyms are words that have two (or more) different possible pronunciations that are associated with two (or more) different meanings [35]. According to the previous research [63], the difficulty of polyphone disambiguation in logographic languages mainly lies in heteronyms. Thus, the previous works [11, 49, 38] usually use "polyphone disambiguation" to refer to "heteronym disambiguation" for those languages. In this work, we use "polyphone disambiguation" to refer to "heteronym disambiguation" following the previous works.

## E Analyses on Prosodic Realization

The DTW measure would be impacted by both lexical tone and prosodic realization. In order to further analyse the naturalness of prosody and rhythm for Dict-TTS and baseline systems, we evaluate the duration errors and character-level average pitch errors. For duration errors, we calculate the MSE of character-level durations. For character-level average pitch errors, we firstly calculate the mean pitch for each character's region in the mel spectrogram according to the Montreal Forced Aligner (MFA) to remove the influence of lexical tone, and then we calculate the MSE of the mean pitch sequences. The results on the Biaobei dataset are shown in Table 6. It can be seen that the duration error and the average pitch error of Dict-TTS are significantly lower than those of the baseline systems, demonstrating the effectiveness of the extracted semantics from prior dictionary knowledge.

Table 6: Duration error and average pitch error comparisons on the Biaobei dataset.

| Method | Duration Error (ms) | Average Pitch Error |
|---|---|---|
| Character | 36.2 | 1424.6 |
| BERT Embedding | 35.7 | 1312.1 |
| NLR | 36.4 | 1414.3 |
| Phoneme (G2PM) | 35.8 | 1341.7 |
| Phoneme (pypinyin) | 35.3 | 1308.8 |
| Dict-TTS | **34.4** | **1232.3** |

## F Adding Rules to the Pronunciation Weights

There are some pronunciation rules (like "sandhi rules") that can not be learned from the dictionary. For example, in Mandarin, "一" before tone4 should be "Y I2" (e.g., "一段") and when "一" is an ordinal word, it should be "Y I1" (e.g., "一四九五年"). According to these pronunciation rules, we can obtain the correct pronunciation labels for some specific characters based on the part-of-speech (POS) tags of the input character sequence. After we obtain the correct pronunciation labels for these specific characters, we can directly force the pronunciation weights of these characters to be the ground truth values.

## G Polyphone Disambiguation for Various Languages

The polyphone disambiguation problem is critical in logographic languages such as Chinese, but is less problematic in phonograms like English.

For logographic languages like Chinese, although the lexicon can cover nearly all the characters, there are a lot of polyphones that can only be decided according to the context of a character. Thus, G2P conversion in this kind of languages is mainly responsible for polyphone disambiguation, which decides the appropriate pronunciation based on the current word context. Therefore, polyphone disambiguation is crucial in these languages and our method is an effective solution for the polyphone disambiguation problem in these languages. For alphabetic languages like English, lexicon cannot cover the pronunciations of all the words. Thus, the G2P conversion for English is mainly responsible for generating the pronunciations of out-of-vocabulary words [50]. Although the polyphone disambiguation is less problematic in these languages, our methods can still be used as the modules

to retrieve the correct pronunciation for polyphones and heteronyms in their G2P process (e.g., the Algorithm step 2 in `https://github.com/Kyubyong/g2p`).

In our experiments, JSUT dataset is a mixture of phonograms and logograms, which is different from Biaobei and Common Voice (HK). Japanese writing system consists of two types of characters: the kanji (漢字) and the syllabic kana – hiragana (平仮名) and katakana (片仮名). In our analysis, 32.42% of the characters in JSUT dataset are kanji. The pronunciations of a part of the kanji can not only be specified by the semantic information and should be specified by empirical pronunciation rules. For example, most kanji (漢字) can be pronounced multiple ways: on-yomi (音読み) and kun-yomi (訓読み). Although the compound kanji usually uses on-yomi and one kanji probably uses kunyomi, the different readings are largely just chosen empirically in practice. Our Dict-TTS has the potential to work only for the kanji whose pronunciation should be specified based on the semantic meaning. Due to the characteristics of Japanese writing systems, in Table 1, although Dict-TTS surpasses the character-based system, it does not show comparable performance with the open source G2P module in Japanese. But as shown in Section 3.4, our method is compatible with the predefined rules from language experts by directly adding specific rules to pronunciation weight. We are sure that the performance of our method can be further improved by introducing the pronunciation rules in Japanese (e.g., the rules in the rule-based G2P baseline "pyopenjtalk").

## H Potential Negative Societal Impacts

Dict-TTS improves the pronunciation accuracy and prosody of the synthesized speech voice and lowers the requirements for G2P conversion, which may cause unemployment for people with related occupations. Besides, the production of fake speeches may cause voice security issues. Further efforts in automatic speaker verification should be made to improve voice security.