RMSSinger: Realistic-Music-Score based Singing Voice Synthesis

Jinzheng He
jinzhenghe@zju.edu.cn
Zhejiang University

Jinglin Liu
liu.jinglin@bytedance.com
ByteDance

Zhenhui Ye
zhenhuiye@zju.edu.cn
Zhejiang University

Rongjie Huang
rongjiehuang@zju.edu.cn
Zhejiang University

Chenye Cui
chenyecui@zju.edu.cn
Zhejiang University

Huadai Liu
huadailiu@zju.edu.cn
Zhejiang University

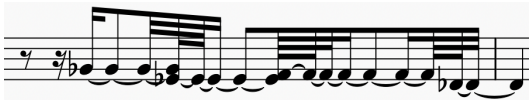
Zhou Zhao *
zhaozhou@zju.edu.cn
Zhejiang University

Abstract

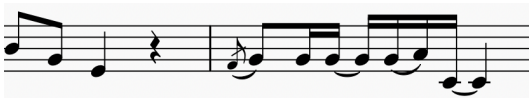
We are interested in a challenging task, **R**ealistic-**M**usic-**S**core based **S**inging **V**oice **S**ynthesis (RMS-SVS). RMS-SVS aims to generate high-quality singing voices given realistic music scores with different note types (grace, slur, rest, etc.). Though significant progress has been achieved, recent singing voice synthesis (SVS) methods are limited to fine-grained music scores, which require a complicated data collection pipeline with time-consuming manual annotation to align music notes with phonemes. Furthermore, these manual annotation destroys the regularity of note durations in music scores, making fine-grained music scores inconvenient for composing. To tackle these challenges, we propose RMSSinger, the first RMS-SVS method, which takes realistic music scores as input, eliminating most of the tedious manual annotation and avoiding the aforementioned inconvenience. Note that music scores are based on words rather than phonemes, in RMSSinger, we introduce word-level modeling to avoid the time-consuming phoneme duration annotation and the complicated phoneme-level mel-note alignment. Furthermore, we propose the first diffusion-based pitch modeling method, which ameliorates the naturalness of existing pitch-modeling methods. To achieve these, we collect a new dataset containing realistic music scores and singing voices according to these realistic music scores from professional singers. Extensive experiments on the dataset demonstrate the effectiveness of our methods. Audio samples are available at https://rmssinger.github.io/.

1 Introduction

Singing Voice Synthesis (SVS) aims to generate high-quality singing given music scores (lyrics, note pitches, and note durations), and has attracted increasing academic and industrial attention. SVS is extensively required in both professional music composing and entertainment industries in real life(Umbert et al., 2015).



(a) fine-grained music scores



(b) realistic music scores

Figure 1: Difference between fine-grained and realistic music scores. Manual adjustment destroys the regularity of note durations, making fine-grained music scores "crushed" and inconvenient for composing.

Though significant progress has been achieved, recent SVS methods(Wang et al., 2022; Zhang et al., 2022b; Liu et al., 2022; Zhang et al.; Huang et al., 2021, 2022a) cannot utilize realistic music scores from composers or websites but require fine-grained music scores. Fine-grained music scores are obtained through a complicated data collection pipeline, which can be mainly divided into three major steps(Wang et al., 2022; Zhang et al.): 1) phoneme annotation step, where the duration of each phoneme is first extracted from singing through Montreal Forced Aligner[1] and then further manually annotated to acquire more accurate phoneme boundaries. 2) note annotation step, where preliminary notes are either created by Logic Pro(Wang et al., 2022) or collected through the word-level average of extracted F0(Zhang et al.)

*Corresponding author.

[1]https://github.com/MontrealCorpusTools/Montreal-Forced- Aligner

Findings of the Association for Computational Linguistics: ACL 2023, pages 236–248
July 9-14, 2023

and then note durations are manually adjusted to the boundaries of vowel phonemes. 3) silence annotation step, where the silence part is annotated as silence (SP) or aspirate (AP).

These steps, especially the first and second steps, require arduous and professional manual annotation(Zhang et al.), which hinders large-scale SVS data collection. Furthermore, since the manual adjustment in the second step destroys the regularity of note durations, fine-grained music scores have a noteworthy difference from realistic music scores (see Figure 1), which inhibits human composers from employing SVS methods for composing.

The primary rationale for adopting such a time-consuming and laborious data collection pipeline is twofold: 1) Existing methods require phoneme-level hard-alignment for duration training. Due to the difficulty in determining phoneme boundaries(Ren et al., 2021), complex manual annotation (phoneme annotation step) is necessary in order to prevent the negative effects of incorrect alignment on model training. 2) Existing methods require a pre-defined phoneme-level mel-note alignment for training and inference. Since one vowel phoneme may correspond to multiple notes(Wang et al., 2022), existing methods directly repeat this phoneme to conform with notes, which requires note boundaries to be aligned with the boundary of each vowel phoneme. However, even professional singers can hardly sing fully conformed to the music score(Zhang et al.), so the note annotation step has to be performed.

To tackle these challenges, we introduce Realistic-Music-Score Singer (RMSSinger), the first RMS-SVS method, which utilizes realistic music scores with different note types (grace, slur, rest, etc.,) for training and inference, alleviating most manual annotations. To alleviate the tedious annotation in the phoneme annotation step, we propose word-level positional attention with word-level hard-alignment and positional attention to avoid the difficulty of determining exact phoneme boundaries. To avoid the note annotation step, we propose the word-level learned Gaussian upsampler to learn the word-level mel-note alignment in training and avoid the phoneme-level mel-note alignment. Furthermore, existing methods mainly adopt simple L1 or L2 loss for pitch modeling, which results in the degradation of expressiveness. To achieve expressive pitch prediction, we propose the first diffusion-based pitch generation method. Due to the existence of both continuous parts (F0) and categorical parts (UV) in pitch contours, we propose the pitch diffusion model (P-DDPM), which models categorical UV and continuous F0 in a single model. Extensive experiments on our collected datasets demonstrate the efficiency of our proposed word-level framework (word-level positional attention and word-level learned Gaussian upsampler) and P-DDPM. The main contributions of this work are summarized as follows:

- We propose the first realistic-music-score-based singing voice synthesis method RMSSinger, which alleviates tedious manual annotation in the current SVS data collection pipeline and achieve high-quality singing voice synthesis given realistic music scores.
- We propose the word-level positional attention and the word-level learned Gaussian upsampler to model lyrics and notes on the word level and avoid phoneme duration annotation and phoneme-level mel-note alignment.
- We propose the first diffusion-based pitch generation model (P-DDPM), which models the continuous F0 and categorical UV in a single model and improves the expressiveness of pitch modeling.
- Extensive experiments demonstrate the performance of our proposed method.

2 Related Works

Singing Voice Synthesis (SVS) aims to generate high-quality singing conditioned on given music scores. With the development of deep learning, SVS has achieved great progress in the network structure and the singing corpus construction. XiaoiceSing(Lu et al., 2020) adopts the non-autoregressive acoustic model inspired by FastSpeech(Ren et al., 2019). ByteSing(Gu et al., 2021) is designed based on the autoregressive Tacotron-like(Wang et al., 2017) architecture. DeepSinger(Ren et al., 2020b) builds a singing corpus by mining singing data from websites and proposes the singing model based on the feed-forward transformer(Ren et al., 2019). More recently, Opencpop(Wang et al., 2022) publish a single-singer Chinese song corpus with manually-annotated fine-grained music scores and propose a Conformer-based(Gulati et al., 2020) SVS method. WeSinger(Zhang et al., 2022b)

adopts a Transformer-like acoustic model and an LPCNet neural vocoder. ViSinger(Zhang et al., 2022a) employs the VITS(Kim et al., 2021) architecture for end-to-end SVS and introduces an F0 predictor to guide the prior network. DiffSinger(Liu et al., 2022) introduces the diffusion-based(Ho et al., 2020) decoder for the high-quality mel-spectrogram generation and proposed the shallow diffusion mechanism for faster inference. M4Singer(Zhang et al.) further publishes a multi-style, multi-singer Chinese song corpus with manually-annotated fine-grained music scores.

3 Diffusion Models

Diffusion models(Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) are a paradigm of generative methods that aim to approximate the end-point distribution (target distribution) of a Markow chain and have achieved impressive results in benchmark generative tasks(Dhariwal and Nichol, 2021). Diffusion models consist of two processes:

Diffusion Process The diffusion process gradually perturbs data $x_0 \sim q(x_0)$ to pure noise with a Markov chain according to the variance schedule $\beta_1, ..., \beta_T$:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}). \tag{1}$$

Reverse Process The reverse process gradually denoises the latent variable $x_T \sim p(x_T)$ to the corresponding real data sample x_0:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \tag{2}$$

where $p_\theta(x_{t-1}|x_t)$ are parameterized with a neural network and learned by optimizing the usual variational bound on negative log-likelihood:

$$\begin{aligned} &E[-logp_\theta(x_0)] \leq \\ &E_q[-logp(x_T) - \sum_{t\geq 1} log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}] = \mathcal{L}, \\ &\mathcal{L}_{t-1} = \mathcal{D}_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \end{aligned} \tag{3}$$

With different perturbation transition $q(x_t|x_{t-1})$ used, different diffusion models are defined:

Gaussian Diffusion: Gaussian diffusion(Ho et al., 2020; Nichol and Dhariwal, 2021) is utilized in continuous data domains. Gaussian diffusion adopts the Gaussian noise for perturbation:

$$\begin{aligned} q(x_t|x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \\ p(x_{t-1}|x_t) &= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)). \end{aligned} \tag{4}$$

With the parameterization introduced in (Ho et al., 2020), Equation 3 can be further simplified and finally optimized with:

$$E_{x_0,\epsilon}[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}||\epsilon - \epsilon_\theta(x_t, t)||], \tag{5}$$

where $\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The neural network is trained to predict the "noise" ϵ from noisy input x_t given timestep t. Gaussian diffusion has been widely utilized for image generation(Nichol and Dhariwal, 2021; Dhariwal and Nichol, 2021) and audio generation(Jeong et al., 2021; Huang et al., 2022b).

Multinomial Diffusion: Multinomial diffusion (Hoogeboom et al., 2021) is utilized in discrete data domains, which replaces Gaussian noise with random walking on discrete data space. The diffusion process can then be defined as:

$$\begin{aligned} q(x_t|x_{t-1}) &= \mathcal{C}(x_t|(1-\beta_t)x_{t-1} + \beta_t/K), \\ q(x_t|x_0) &= \mathcal{C}(x_t|\bar{\alpha}_t x_0 + (1-\bar{\alpha}_t)/K) \end{aligned} \tag{6}$$

where $\mathcal{C}$ denotes a categorical distribution with probability parameters, $x_t \sim \{0, 1\}^K$, β_t denotes the chance of resampling a category uniformly, and $\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Using Equation 6, we can compute the categorical posterior:

$$\begin{aligned} &q(x_{t-1}|x_t, x_0) = \mathcal{C}(x_{t-1}|\theta_{post}(x_t, x_0)), \\ &\theta_{post}(x_t, x_0) = \tilde{\theta}/\sum_{k=1}^{K}\tilde{\theta}_k, \\ &\tilde{\theta} = [\alpha_t x_t + (1-\alpha_t)/K] \odot [\bar{\alpha}_{t-1}x_0 + \\ &(1-\bar{\alpha}_{t-1})/K], \end{aligned} \tag{7}$$

With the parameterization proposed in (Hoogeboom et al., 2021), $p(x_{t-1}|x_t) = \mathcal{C}(x_{t-1}|\theta_{post}(x_t, \hat{x_0}))$ is utilized to approximate $q(x_{t-1}|x_t, x_0)$. And the neural network is trained to approximate $\hat{x_0}$ from noisy sample x_t given timestep t.

Though widely utilized in many data domains, diffusion models have never been utilized for pitch modeling. Furthermore, due to the existence of continuous F0 parts and discrete UV parts in pitch contours(Wang et al., 2018), neither Gaussian diffusion nor multinomial diffusion alone can deal with

pitch modeling. In this paper, we propose the first diffusion-based pitch modeling (P-DDPM), which incorporates Gaussian diffusion and multinomial diffusion in a single model and achieves better pitch modeling.

4 Methdology

4.1 Overview

In this section, we introduce the overall architecture of our proposed RMSSinger. As shown in Figure 2a, RMSSinger is built on one of the most popular non-autoregressive TTS models FastSpeech2(Ren et al., 2020a). Lyrics are encoded through the phoneme encoder and then aligned to the lengths of mel-spectrogram through the word-level positional attention layer (Section 4.4)to obtain the expanded lyric feature. Next, we utilize the note encoder to encode note pitches, note durations, and note types (rest, slur, grace, etc.) and adopt the word-level learned Gaussian upsampler (Section 4.3) for word-level mel-note alignment learning to obtain the expanded note feature. The timbre information of different singers is embedded to obtain the singer embedding. Then, the expanded lyric feature, expanded note feature and singer embedding are summed as the pitch decoder input. The pitch diffusion model (P-DDPM) (Section 4.5) utilizes the pitch decoder input as the condition to generate pitches (F0 and UV). Similar to (Ren et al., 2020a), we obtain the pitch embedding through F0 and UV. Next, the expanded lyric feature, pitch embedding and singer embedding are summed as the input of the mel decoder. Finally, to further improve the quality of the synthesized mel-spectrogram, we introduce a diffusion-based post-net (Section 4.7) to refine the coarse outputs of the mel decoder.

4.2 Encoder

In this subsection, we introduce the phoneme encoder and the note encoder utilized in RMSSinger. The phoneme encoder takes the phoneme sequence as input and outputs the phoneme feature $\mathcal{H}$. We also perform the word-pooling on $\mathcal{H}$ to obtain the word-level feature $\mathcal{H}_w$. The architecture of the phoneme encoder is comprised of a series of Feed-Forward Transformer Blocks (Vaswani et al., 2017), which have proven the effectiveness of long sequences modeling and linguistic information extraction in TTS methods. The input of the note encoder is the realistic music score. As there exist different types of information in music scores, the note encoder includes an embedding layer for note pitches, an embedding layer for note types (rest, slur, grace, etc.), and a linear projection layer for note durations. All information types are summed as the note feature $\mathcal{H}_n$.

4.3 Word-level Learned Gaussian Upsampler

One of the key challenges of SVS is the alignment between word-level mel-spectrogram and notes, that is the actual length[2] of each note. Though the note duration on music scores provides a preliminary estimate of the actual length, even professional singers cannot precisely conform to the music score. Therefore, previous SVS methods manually adjust the note duration to the phoneme boundary, which not only requires time-consuming annotation from experts but also destroys the regularity of the note duration.

The key idea of the proposed word-level learned Gaussian upsampler (see figure 2b), inspired by (Donahue et al., 2020), is to learn the word-level mel-note alignment in training. Given the note feature $\mathcal{H}_n$, the word-level feature $\mathcal{H}_w$ and the singer embedding s, we expand $\mathcal{H}_w$ to the note-level $\mathcal{H}_{wn}$. Next we predict the actual length of each note:

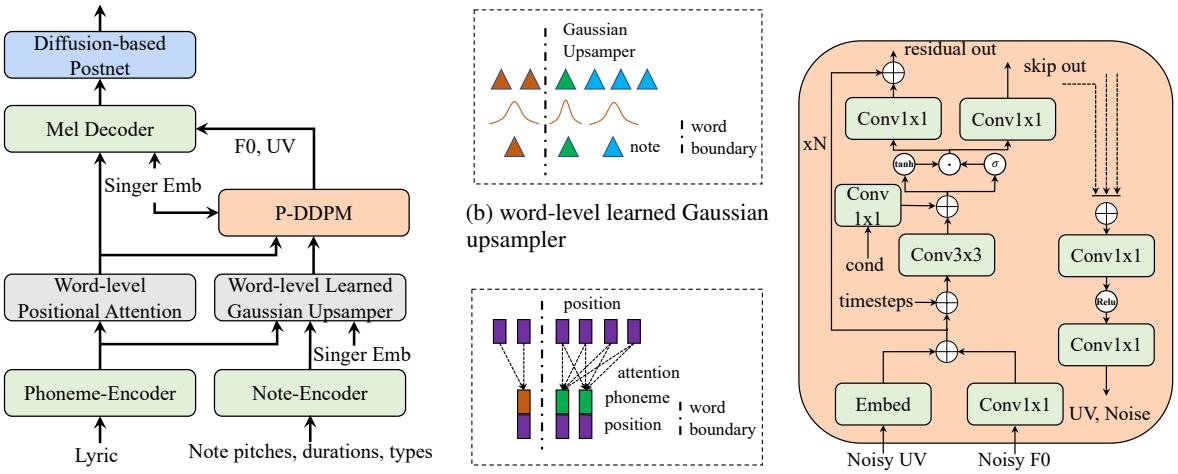
$$\begin{aligned} \mathcal{H}_a &= \mathcal{H}_n + \mathcal{H}_{wn} + s, \\ l_n &= f(\mathcal{H}_a), \end{aligned} \tag{8}$$

with a neural network f. The neural network consists of a stack of 1D-convolution, Relu, and layer normalization. We use a linear projection with ReLU nonlinearity at the output to make l_n non-negative, which ensures the monotonicity and none of the notes can be ignored. Then we upsample the note feature to its corresponding actual length. We introduce the Gaussian distribution to make the upsampling process differentiable and learnable. To be specific, given the predicted actual length, we can find the end position of each note $e_n = \sum_{m=1}^{n} l_m$, and then the center position of each note $c_n = e_n - \frac{1}{2} l_n$. We place a Gaussian distribution with fixed deviation σ at the center c_n of the output segment corresponding to the note n. Then we can define:

$$w_t^n = \frac{exp(-\frac{(t-c_n)^2}{2\sigma^2})}{\sum_m exp(-\frac{(t-c_m)^2}{2\sigma^2})}, t \in (0, T) \tag{9}$$

where T denotes the length of the mel-spectrogram and w_t^n represents the weight of each note for

[2]The number of the mel-frames



(a) overall architecture

(b) word-level learned Gaussian upsampler

(c) word-level positional attention

(d) P-DDPM

Figure 2: The overall architecture of our proposed RMSSinger is shown in subfigure (a). Lyrics are first encoded through the phoneme encoder and then expanded through the word-level positional attention in subfigure (c). Notes are encoded through the note encoder and then expanded through the word-level learned Gaussian upsampler in subfigure (b). The expanded note feature, expanded lyric feature and singer embedding are summed and used as the condition of P-DDPM in subfigure (d).

the output position t. And finally, the expanded note feature at position t can be calculated as $a_t = \sum_n w_t^n \mathcal{H}_n$. We highlight that 1) when calculating a_t, we only consider the contribution from the same word, that is if position t belongs to the range of word i and note n corresponds to the word j and $i \neq j$, then $w_t^n = 0$. 2) During training, we use the ground-truth duration of each word to determine the range of position t and avoid expensive DTW calculation. 2) During inference, we use the sum of the predicted actual length of notes $\sum_m l_m, m \in word_i$ as the predicted duration of $word_i$. 3) We use the ground-truth word duration to constrain the prediction of the actual length, which is computed as:

$$\mathcal{L}_d = || \sum_m l_m - dur_i ||, \tag{10}$$

where dur_i denotes the ground-truth duration of $word_i$.

4.4 Word-level Positional Attention

To align the lyric features (outputs of the phoneme encoder) to the lengths of the mel-spectrogram, previous SVS methods mainly adopt the duration predictor to predict the number of frames of each phoneme. Due to the complex articulation of each phoneme in singing, these methods have to use manually-annotated phoneme duration for training, which increases the cost of data collection. Note that most music scores are word-level and word boundaries are much easier to be determined, inspired by (Ren et al., 2021; Miao et al., 2020), we propose the word-level positional attention (see figure 2c), which avoids the annotation of phoneme duration. To be specific, given the output of the phoneme encoder $\mathcal{H}$, let the word-level phoneme positional encoding which represents the position of each phoneme in a word be $\mathcal{P}_{ph}$, and let the word-level mel-spectrogram positional encoding which denotes the position of each frame in a word be $\mathcal{P}_m$, we introduce the position-to-phoneme attention:

$$\begin{aligned} \mathcal{H}_k &= W(cat(\mathcal{H}, \mathcal{P}_{ph})), \\ \mathcal{H}_{epd} &= Softmax(\frac{\mathcal{P}_m \mathcal{H}_k^T}{\sqrt{d}})\mathcal{H}^T, \end{aligned} \tag{11}$$

where W represents a linear projection, and $\mathcal{H}_{epd}$ represents the expanded lyric-feature. During training, we use the ground-truth word durations to obtain word-level mel-spectrogram positional encoding. During inference, we use the predicted word duration $\sum_m l_m$ introduced in subsection 4.3.

4.5 Pitch Diffusion Model

To generate the pitch contours, previous methods mainly adopt a pitch predictor which predicts the continuous fundamental frequency (F0) and the discrete unvoiced label (UV). The pitch predictor is constrained with simple L1 or L2 loss for F0 and cross-entropy loss for UV. However, due to the complicated pitch variation of the singing voice, the

simple pitch predictor fails to model the variance, resulting in degraded expressiveness. To tackle this challenge, we propose the first pitch diffusion model (P-DDPM) (see Figure 2d), which incorporates both the Gaussian diffusion and multinomial diffusion to generate F0 and UV. During the diffusion process, the Gaussian noise (see Equation 4) and random resampling (see Equation 6) are used to perturb the continuous F0 (represented by x) and the discrete UV labels (represented by y) correspondingly:

$$\begin{aligned} q(x_t|x_{t-1}) &= \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \\ q(y_t|y_{t-1}) &= \mathcal{C}(y_t|(1-\beta_t)y_{t-1} + \beta_t/K). \end{aligned} \tag{12}$$

During the reverse process, following Equation 5 and Equation 7, the neural network is utilized to predict the corresponding $\epsilon_\theta(x_t, t)$ and $\hat{y_0}$. We adopt a non-causal WaveNet (Oord et al., 2016) architecture as our denoiser, which has proven to be effective in modeling sequential data. We design a 1x1 convolution layer for the continuous F0 and an embedding layer for the discrete UV label in order to perform Gaussian F0 diffusion and multinomial UV diffusion in a single model. The neural network is optimized through the corresponding Gaussian diffusion loss $\mathcal{L}_{gdiff}$ and multinomial diffusion loss $\mathcal{L}_{mdiff}$.

4.6 Decoder

In this subsection, we introduce the mel decoder utilized in RMSSinger. The mel decoder takes the expanded lyric feature, singer embedding and pitch embedding as input and outputs the coarse mel-spectrogram. Following previous speech synthesis methods (Huang et al., 2022c; He et al., 2022), we use a stack of Feed-Forward Transformer blocks as the architecture and use the L1 loss function to optimize the mel decoder:

$$\mathcal{L}_{mel} = ||mel_p - mel_g||, \tag{13}$$

where mel_p denotes the predicted coarse mel-spectrogram and mel_g denotes the ground-truth mel-spectrogram.

4.7 Diffusion-Based Post-Net

To achieve high-quality singing voice synthesis, we have to capture the rich and highly dynamic variation in the singing voice. However, the widely-applied transformer-based decoder (mel decoder) is difficult to generate detailed mel-spectrogram samples(Ren et al., 2022; Ye et al., 2023, 2022). To further improve the quality of generated samples, we introduce the diffusion-based post-net, which converts the coarse outputs of the mel decoder into fine-grained ones. In detail, we use the coarse outputs as the condition of the diffusion model for training and inference. We use the Gaussian diffusion loss $\mathcal{L}_{post}$ similar to the previous diffusion-based TTS method (Jeong et al., 2021) to optimize the diffusion-based postnet.

4.8 Training Pipeline

There are two training stages for RMSSinger: during the first stage, we optimize the whole model except the diffusion-based postnet by minimizing the following loss function:

$$\mathcal{L}_1 = \mathcal{L}_{gdiff} + \mathcal{L}_{mdiff} + \mathcal{L}_d + \mathcal{L}_{mel} \tag{14}$$

We obtain coarse mel-spectrogram after the first stage of training. In the second training stage, we freeze the whole model except the diffusion-based postnet and only optimize the diffusion-based postnet by minimizing $\mathcal{L}_{post}$.

5 Experiments

5.1 Experimental Setup

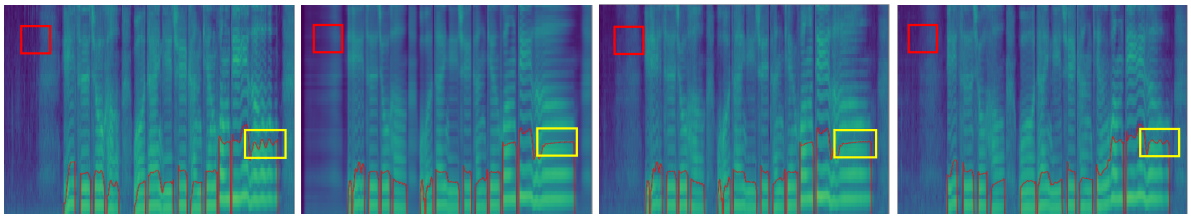
In this section, we first describe our collected dataset for RMS-SVS, and then introduce the implementation details of our proposed RMSSinger. Finally, we explain the training and evaluation details utilized in this paper.

Dataset Currently, there are no public SVS datasets providing realistic music scores, so we collect and annotate a high-quality Chinese song corpus (about 12 hours in total) with realistic music scores. Professional singers are recruited to sing conforming to these realistic music scores. They are paid based on their singing time. Next, word durations are extracted through an external speech-text aligner and then manually finetuned. Since we do not need fine-grained phoneme durations, the finetune process requires much less effort. Finally, we annotate the silence and aspirate parts since these parts are not provided in most realistic music scores. All audio files are recorded in a professional recording studio, which guarantees the high quality of our dataset. All audios are sampled as 48000 Hz with 24-bit quantization, and we randomly select one song from each singer for the testing.

Implementation Details We convert Chinese lyrics into phonemes through pypinyin. We extract mel-spectrogram from raw waveforms and

Method	F0RMSE ↓	VDE ↓	MCD ↓	MOS-P ↑	MOS-Q ↑
GT	-	-	-	4.55 ± 0.04	4.58 ± 0.03
GT(vocoder)	3.77	0.020	1.33	4.10 ± 0.04	4.09 ± 0.05
FFTSinger(Zhang et al.)	14.0	0.092	3.52	3.57 ± 0.08	3.46 ± 0.07
DiffSinger(Liu et al., 2022)	12.4	0.077	3.43	3.63 ± 0.07	3.79 ± 0.07
RMSSinger (ours)	**12.2**	**0.069**	**3.42**	$\mathbf{3.77 \pm 0.05}$	$\mathbf{3.84 \pm 0.06}$

Table 1: Performance comparison with different methods. We use F0RMSE, VDE and MCD for objective evaluation. And we use MOS-P and MOS-Q for subjective measurement.



(a) ground-truth (b) FFTSinger (c) DiffSinger (d) RMSSinger

Figure 3: Visualization of the pitch contour and mel-spectrograms of ground-truth and different methods.

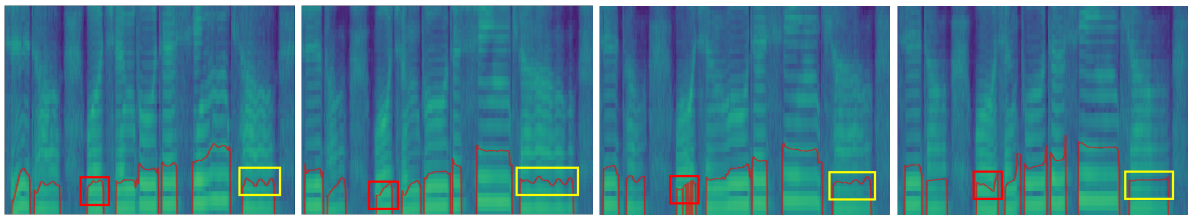
set the sample rate to 24000Hz, the window size to 512, the hop size to 128, and the number of mel bins to 80. In the phoneme encoder and the mel decoder, we adopt a similar setting as that in FastSpeech2(Ren et al., 2020a). In the word-level learned Gaussian upsampler, the kernel size of 1D-convolution is set to 5 and the hidden channel is set to 256. In the word-level positional attention, we set the number of attention heads to 1. In the P-DDPM, we set the number of convolution layers to 12, the kernel size to 3, the residual channel to 192 and hidden channel to 256. And we also set the total number of diffusion steps to 100 and adopt the linear β schedule from 0.0001 to 0.06. The diffusion post-net has a similar architecture and β schedule except that the number of convolution layers is set to 20, and the residual channel is set to 256. More details can be found in Appendix B.

Evaluation Details In our experiments, we employ objective and subjective evaluation metrics to evaluate the pitch modeling and the audio quality of generated samples. For the objective evaluation, we utilize F0 Root Mean Square Error(F0RMSE) to measure the accuracy of F0 prediction and Voice Decision Error(VDE) to measure the accuracy of UV prediction. We use Mean Cepstral Distortion (MCD) for audio quality measurement. For the subjective evaluation, we use Mean Opinion Score (MOS) for main results and Comparision Mean Opinion Score (CMOS) for ablations. For a more detailed examination, we score MOS-P/CMOS-P and MOS-Q/CMOS-Q corresponding to the MOS/CMOS of pitch modeling and audio quality. We utilize the HiFi-GAN(Kong et al., 2020) vocoder published in DiffSinger(Liu et al., 2022) for all experiments. More details can be found in Appendix A.

5.2 Main Results

In this subsection, we conduct extensive experiments to compare the performance of RMSSinger with other baselines. Since RMS-SVS is a new task, none of the existing methods can handle it. Therefore, we implement several representative and state-of-art SVS methods(Zhang et al.; Liu et al., 2022) with our proposed word-level framework to handle realistic music scores. Specifically, FFTSinger(Zhang et al.) adopts a similar architecture to FastSpeech2(Ren et al., 2020a), which uses MSE loss for F0 training and binary cross entropy loss for UV training. Besides, FFTSinger adopts the FFT decoder and uses the L1 loss for mel-spectrogram reconstruction. DiffSinger(Liu et al., 2022) uses the same pitch modeling method but replaces the FFT decoder in FFTSinger with the diffusion-based decoder and uses the Gaussian diffusion for mel-spectrogram training.

The main results are shown in Table 1. From the objective and subjective results, we can see that 1) most methods achieve promising results, which illustrates the feasibility of RMS-SVS and the effectiveness of our proposed word-level framework. 2) RMSSinger achieves better results on F0RMSE, VDE, and MOS-P, which demonstrates



(a) ground-truth (b) RMSSinger (c) w/o UV diffusion (d) w/o F0 diffusion

Figure 4: Visualization of ground truth and generated results of RMSSinger and different ablations

our proposed P-DDPM can improve both the F0 and UV modeling and improve the expressiveness. 3) RMSSinger and DiffSinger achieve similar results on MCD and MOS-Q, which is because the diffusion-based postnet of RMSSinger and the diffusion decoder of DiffSinger have a similar architecture. 4) RMSSinger and DiffSinger outperform FFTSinger in terms of audio quality by a large margin due to the existence of mel-spectrogram over-smooth in FFTSinger.

We then visualize the mel-spectrogram and pitch contour generated by different methods in Figure 3 to show the difference among different methods more intuitively. We can find that 1) RMSSinger can generate more natural pitch contours especially in the vibrato part (yellow box region), which demonstrates our method can achieve better pitch modeling. 2) RMSSinger and DiffSinger can generate more detailed mel-spectrogram, and alleviate the mel-spectrogram over-smooth (see red box region), which explains the higher audio quality.

Method	F0RMSE	VDE	CMOS-P
Full model	**12.2**	**0.069**	**0.0**
w/o UV diffusion	12.6	0.080	-0.58
w/o F0 diffusion	12.8	0.070	-0.50

Table 2: Ablation studies on the effect of P-DDPM.

Method	MCD	CMOS-Q
Full model	**3.42**	**0.0**
w/o diffusion postnet	3.48	-0.91

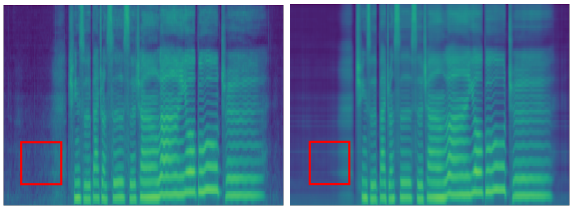
Table 3: Ablation studies on the postnet.

5.3 Ablation Studies

In this subsection, we conduct a series of ablation studies to investigate the effect of key components in our RMSSinger.

P-DDPM To evaluate the effectiveness of our proposed P-DDPM, we design two ablations. In the first ablation, we remove the UV diffusion from P-DDPM and we use a Transformer-based UV predictor, which is constrained by binary cross-entropy loss. In the second ablation, we remove the F0 diffusion from P-DDPM and use a Transformer-based F0 predictor, which is constrained by L1 loss. The results can be found in Table 2. We can see that 1) in the first ablation, VDE increases significantly and CMOS-P degrades, which demonstrates that the UV diffusion in P-DDPM contributes to better UV modeling; 2) in the second ablation, F0RMSE increases, VDE nearly holds the same and CMOS-P degrades which demonstrates the F0 diffusion is essential to natural F0 modeling.

We also visualize the pitch contours of different ablations in Figure 4. We can find that 1) without UV diffusion, there exists unpleasant UV errors (see the red box). 2) without F0 diffusion, the model can not generate natural F0, especially in the vibrato part (yellow box region).



(a) RMSSinger (b) w/o postnet

Figure 5: Visualization of generated results of RMSSinger and the ablation

Postnet To evaluate the effectiveness of our proposed diffusion-based postnet, we remove the postnet and utilize the output of the mel decoder as the generated samples. The results can be found in Table 3. We can find that MCD increases and CMOS-Q degrades, which demonstrates that the diffusion postnet contributes to a better mel-spectrogram prediction. We also visualize the generated mel-spectrograms. As shown in Figure 5, the diffusion-based postnet contributes to alleviating the mel-spectrogram over-smooth (see red box region).

6 Conclusion

In this paper, we propose RMSSinger, the first realistic-music-score-based singing voice synthesis (RMS-SVS) method, which utilizes the word-level modeling framework to avoid most tedious manual annotations. To achieve better pitch modeling, we propose the first diffusion-based pitch modeling method (P-DDPM), which incorporates the Gaussian diffusion and multinomial diffusion in a single model. Extensive experiments conducted on our collected dataset demonstrate the feasibility of our method for RMS-SVS and the superiority of our proposed P-DDPM.

7 Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant No.2022ZD0162000, National Natural Science Foundation of China under Grant No.62222211, Grant No.61836002 and Grant No.62072397.

8 Limitations

There are majorly two limitations: Firstly, we collect a Chinese singing voice dataset and test our method only on this Chinese dataset due to the difficulty of recruiting professional singers in different languages. In the future, we will attempt to collect the singing voices dataset including more languages and test our method in multilingual settings. Secondly, our method adopts the diffusion model in pitch modeling and the postnet, which require multiple inference steps. We will try advanced acceleration methods for diffusion models in the future.

9 Ethics Statement

RMSSinger provides a high-quality realistic-music-score-based singing voice synthesis method, which may cause unemployment for people with related occupations. Furthermore, the possible misuse of realistic music scores from the website may lead to copyright issues. We will add some constraints to guarantee people who use our code or pre-trained model would not use the model in illegal cases.

References

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794.

Jeff Donahue, Sander Dieleman, Mikołaj Bińkowski, Erich Elsen, and Karen Simonyan. 2020. End-to-end adversarial text-to-speech. *arXiv preprint arXiv:2006.03575*.

Yu Gu, Xiang Yin, Yonghui Rao, Yuan Wan, Benlai Tang, Yang Zhang, Jitong Chen, Yuxuan Wang, and Zejun Ma. 2021. Bytesing: A chinese singing voice synthesis system using duration allocated encoder-decoder acoustic models and wavernn vocoders. In *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. IEEE.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Jinzheng He, Zhou Zhao, Yi Ren, Jinglin Liu, Baoxing Huai, and Nicholas Yuan. 2022. Flow-based unconstrained lip to speech generation.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465.

Rongjie Huang, Feiyang Chen, Yi Ren, Jinglin Liu, Chenye Cui, and Zhou Zhao. 2021. Multi-singer: Fast multi-singer singing voice vocoder with a large-scale corpus. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3945–3954.

Rongjie Huang, Chenye Cui, Feiyang Chen, Yi Ren, Jinglin Liu, Zhou Zhao, Baoxing Huai, and Zhefeng Wang. 2022a. Singgan: Generative adversarial network for high-fidelity singing voice generation. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2525–2535.

Rongjie Huang, Max WY Lam, Jun Wang, Dan Su, Dong Yu, Yi Ren, and Zhou Zhao. 2022b. Fastdiff: A fast conditional diffusion model for high-quality speech synthesis. *arXiv preprint arXiv:2204.09934*.

Rongjie Huang, Yi Ren, Jinglin Liu, Chenye Cui, and Zhou Zhao. 2022c. Generspeech: Towards style transfer for generalizable out-of-domain text-to-speech synthesis. *arXiv preprint arXiv:2205.07211*.

Myeonghun Jeong, Hyeongju Kim, Sung Jun Cheon, Byoung Jin Choi, and Nam Soo Kim. 2021. Diff-tts: A denoising diffusion model for text-to-speech. *arXiv preprint arXiv:2104.01409*.

Jaehyeon Kim, Jungil Kong, and Juhee Son. 2021. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pages 5530–5540. PMLR.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033.

Jinglin Liu, Chengxi Li, Yi Ren, Feiyang Chen, and Zhou Zhao. 2022. Diffsinger: Singing voice synthesis via shallow diffusion mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11020–11028.

Peiling Lu, Jie Wu, Jian Luan, Xu Tan, and Li Zhou. 2020. Xiaoicesing: A high-quality and integrated singing voice synthesis system. *arXiv preprint arXiv:2006.06261*.

Chenfeng Miao, Shuang Liang, Minchuan Chen, Jun Ma, Shaojun Wang, and Jing Xiao. 2020. Flow-tts: A non-autoregressive network for text to speech based on flow. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7209–7213. IEEE.

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020a. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*.

Yi Ren, Jinglin Liu, and Zhou Zhao. 2021. Portaspeech: Portable and high-quality generative text-to-speech. *Advances in Neural Information Processing Systems*, 34:13963–13974.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in Neural Information Processing Systems*, 32.

Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu. 2020b. Deepsinger: Singing voice synthesis with data mined from the web. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1979–1989.

Yi Ren, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2022. Revisiting over-smoothness in text to speech. *arXiv preprint arXiv:2202.13066*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Marti Umbert, Jordi Bonada, Masataka Goto, Tomoyasu Nakano, and Johan Sundberg. 2015. Expression control in singing voice synthesis: Features, approaches, evaluation, and challenges. *IEEE Signal Processing Magazine*, 32(6):55–73.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xin Wang, Shinji Takaki, and Junichi Yamagishi. 2018. Autoregressive neural f0 model for statistical parametric speech synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(8):1406–1419.

Yu Wang, Xinsheng Wang, Pengcheng Zhu, Jie Wu, Hanzhao Li, Heyang Xue, Yongmao Zhang, Lei Xie, and Mengxiao Bi. 2022. Opencpop: A high-quality open source chinese popular song corpus for singing voice synthesis. *arXiv preprint arXiv:2201.07429*.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. 2017. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*.

Zhenhui Ye, Rongjie Huang, Yi Ren, Ziyue Jiang, Jinglin Liu, Jinzheng He, Xiang Yin, and Zhou Zhao. 2023. Clapspeech: Learning prosody from text context with contrastive language-audio pre-training. *2305.10763*.

Zhenhui Ye, Zhou Zhao, Yi Ren, and Fei Wu. 2022. Syntaspeech: syntax-aware generative adversarial text-to-speech. *arXiv preprint arXiv:2204.11792*.

Lichao Zhang, Ruiqi Li, Shoutong Wang, Liqun Deng, Jinglin Liu, Yi Ren, Jinzheng He, Rongjie Huang, Jieming Zhu, Xiao Chen, et al. M4singer: A multi-style, multi-singer and musical score provided mandarin singing corpus. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yongmao Zhang, Jian Cong, Heyang Xue, Lei Xie, Pengcheng Zhu, and Mengxiao Bi. 2022a. Visinger: Variational inference with adversarial learning for end-to-end singing voice synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7237–7241. IEEE.

Zewang Zhang, Yibin Zheng, Xinhui Li, and Li Lu. 2022b. Wesinger: Data-augmented singing voice synthesis with auxiliary losses. *arXiv preprint arXiv:2203.10750*.

A More Experimental Details

A.1 Subjective Evaluation

We randomly select 16 sentences from the test set for the subjective evaluation. Each ground-truth audio or generated audio has been listened to by at least 15 professional listeners. For MOS-P and CMOS-P, listeners are told to focus on the naturalness of pitch modeling (e.g., vibrato part, UV part, and so on). For MOS-Q and CMOS-Q, listeners are told to focus on audio quality (e.g., noise, high-frequency details, pronunciation, and so on). For MOS, each listener is asked to evaluate different audio samples on a 1 - 5 Likert scale. For CMOS, listeners are told to compare pairs of audio generated by different systems and indicate which of the two audio they prefer and following the rule: 0 indicating no difference, 1 indicating a small difference, and 2 indicating a large difference. All listeners get equally paid.

A.2 Training Details

We train and evaluate our model on a single NVIDIA 2080Ti GPU. Adam optimizer is used with $\beta_1 = 0.9, \beta_2 = 0.98$. It takes 180000 steps for the first stage of training and 160000 steps for the second stage. It takes about 24 hours for each stage of training on a single NVIDIA 2080Ti GPU.

B More Model Details

B.1 Encoder

Our phoneme encoder consists of 1 phoneme embedding layer and 4 Feed-Forward Transformer (FFT) blocks. Each FFT block consists of 1 multi-head attention layer with 2 attention heads and 1 1D convolution layer with the kernel size set to 5. All hidden channels are set to 256.

B.2 Decoder

Our mel decoder has a similar architecture to the phoneme encoder except that the mel decoder does not consist of the phoneme embedding layer.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☑ A2. Did you discuss any potential risks of your work?
Section 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract, Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5.1, Appendix A.2, Appendix B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5.1, Appendix A.2

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5.2, Section 5.3

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5.1, Appendix A.2

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Section 5.1, Appendix A.1

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Section 5.1, Appendix A.1

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Section 5.1, Appendix A.1

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☒ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
They're completely anonymous, so we don't know