Supplementary Material for TokenSplit: Using Discrete Speech Representations for Direct, Refined, and Transcript-Conditioned Speech Separation and Recognition

Hakan Erdogan[1], *Scott Wisdom*[1], *Xuankai Chang*[2], *Zalan Borsos*[1], *Marco Tagliasacchi*[1], *Neil Zeghidour*[1], *John R. Hershey*[1]

[1]Google Research
[2]Carnegie Mellon University

{hakanerdogan, scottwisdom, zborsos, mtagliasacchi, neilz, johnhershey}@google.com, xuankaic@andrew.cmu.edu

1. Second-Stage Processing

We experimented with adding a second stage similar to AudioLM's [1] third stage to refine coarse acoustic tokens into fine acoustic tokens. Another encoder-decoder EncDec_2 is used to generate the fine acoustic tokens given the coarse acoustic tokens (and semantic tokens, if available) from the first stage model for each source i. The computation can be shown as follows, and is shown in Figure 1:

$$\hat{\mathbf{A}}_i^{>Q''} = \text{EncDec}_2(\hat{\mathbf{S}}_i, \hat{\mathbf{A}}_i^{\leq Q''}).$$

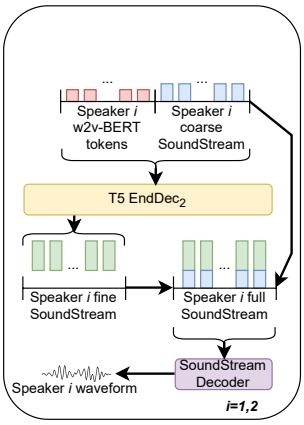

Figure 1: *Illustration of stage 2 processing.*

If semantic tokens are not available during inference, we replace them with mask tokens. During training, we used only the first source to train the second stage model and semantic tokens were replaced with mask tokens with 50% probability. We found that our results did not improve or improved little by using coarse+fine tokens instead of coarse tokens only. It seemed first $Q'' = 6$ levels of SoundStream model we trained produced high enough quality for our purposes. We present results from using fine tokens in Section 3.

2. Masking Probabilities

We used input masking patterns with certain probabilities of masking certain combination of inputs. A mask pattern has a value of 1 for an input that is not masked and 0 for an input that is masked for the inputs $W_1+W_2+S_m+A_m$ that is transcript1, transcript2, semantic tokens and acoustic tokens of the mixture signal. As an example, 0011 means two target transcripts are masked and semantic and acoustic tokens are not masked. For our basic separation models (variants 1, 2 and 3 below), the probabilities of mask patterns were 0.55 for 0011, 0.1 for 1100, 1011, 0111 and 1111 and 0.05 for 0000. In addition to the full masking of each kind of tokens, we also masked a random contiguous segment of each kind with the segment length to full length ratio randomly chosen between 0 and 0.5. The second stage model which uses $S_1 + A_1$ as input is also trained with masking pattern probabilities of 0.5 for patterns 11 and 01, that is input semantic tokens are masked with 0.5 probability. For stage 2 model, whenever we do not have the semantic tokens predicted by stage 1 during inference, we just replace the semantic token part with mask tokens.

3. TokenSplit Variants

Here are the TokenSplit variants we considered and experimented with, shown as (input tokens) -> (output tokens).

1. $W_1 + W_2 + S_m + A_m$ -> $W_1 + W_2 + S_1 + S_2 + A_1 + A_2$
2. $W_1 + W_2 + S_m + A_m$ -> $S_1 + S_2 + A_1 + A_2 + W_1 + W_2$
3. $W_1 + W_2 + S_m + A_m$ -> $A_1 + A_2 + S_1 + S_2 + W_1 + W_2$ (default)
4. $W_1 + W_2 + A_m$ -> $W_1 + W_2 + A_1 + A_2$
5. $W_1 + W_2 + A_m$ -> $A_1 + A_2 + W_1 + W_2$
6. $S_m + A_m$ -> $S_1 + S_2 + A_1 + A_2 + W_1 + W_2$
7. A_m -> $A_1 + A_2 + W_1 + W_2$

The performance metrics for each variant is provided in Table 1. As mentioned in the main text, variant 3 performed the best. The reason this variant did better than others that first generate text (variant 1) or semantic tokens (variant 2) could be that when transcripts or semantic tokens are generated first, the autoregressive token generation model can rely on them being accurate and produce acoustic tokens that match them. However, if there is an error when sampling the transcript or semantic tokens, the model ends up generating audio that does not match with the ground truth input contents. The model is not great in its ASR performance (as compared to a dedicated ASR model), so it is expected that this causes higher word error rates.

We see that we obtain much better performance in DNSMOS with TokenSplit models as compared to a TDCN++ model, and we get close performance in DWER and DCER metrics. VISQOL score is not better than TDCN++ but it is not too much lower and VISQOL may not be the best metric for evaluating generative model output quality due to possible frame-level misalignments between ground truth and generated

Table 1: *TokenSplit separation results using 7 different variants on first 3 seconds of the Libri2Mix test set compared to TDCN++.*

Model	SI-SNRi	DNSMOS	VISQOL	DWER	DCER
Mixture	0.0	3.39	1.50	86.6	62.2
TDCN++	12.3	2.96	2.11	25.1	14.9
v1 coarse	-3.7	3.51	2.03	27.1	15.9
v1 coarse+fine	-3.7	3.53	2.03	26.7	15.7
v2 coarse	-3.8	3.51	2.01	29.0	17.3
v2 coarse+fine	-3.8	3.53	2.01	29.1	17.5
v3 coarse	-3.5	3.50	2.04	26.6	15.4
v3 coarse+fine	-3.5	3.52	2.05	25.8	14.8
v4 coarse	-4.4	3.49	1.84	42.4	26.2
v4 coarse+fine	-4.9	3.48	1.73	43.1	26.5
v5 coarse	-3.6	3.49	2.03	27.9	16.1
v5 coarse+fine	-4.0	3.48	1.90	29.4	17.0
v6 coarse	-4.1	3.51	1.98	31.6	19.1
v6 coarse+fine	-4.2	3.53	1.98	31.8	19.3
v7 coarse	-3.9	3.49	1.98	30.3	17.7
v7 coarse+fine	-4.2	3.48	1.86	31.2	18.2

signal. The variant that does not use semantic tokens (variant 5) achieved quite close performance to the best variant. This may show that semantic tokens are not that important for this task even though they end up helping a little. Listening tests also verify the superiority of the TokenSplit generated output as shown in the paper.

4. Using multiple samplings

In this section, we test how the results change when we use multiple samplings and choose among them. In addition to the one sampling we had from the models, we generated two more samplings from the models and picked the best output based on DNSMOS metric which does not use any reference signal. The results are given in Table 2. It can be observed that choosing best of 3 samplings improve DNSMOS, but does not change other metrics much, so we just reported the result from a single sampling.

Table 2: *Results comparing single sampled output versus picking the best of three samplings from the models using DNSMOS.*

Model	DNSMOS	ViSQOL	DWER	DCER
TokenSplit	3.50	2.04	26.6	15.4
TokenSplit bestof3	3.58	2.05	26.6	15.3
TokenSplitRefine	3.51	2.29	17.2	9.5
TokenSplitRefine bestof3	3.58	2.29	17.2	9.5

5. ASR

We present ASR comparison results for different variants of the model. The model output transcript is compared with running the standalone ASR model on the model separated audio signals in Table 3. We can see that using the model as an ASR model for mixed speech does not work as well as applying a standalone ASR model on separated audio signals. We plan to explore and improve this result in future work.

Table 3: *Comparison of ASR performance in terms of differential WER using the model's output transcripts and running ASR on the model's coarse output sources.*

Model	DWER	DWER(model output)
v1 coarse	27.1	74.4
v2 coarse	29.0	74.6
v3 coarse	26.6	71.9
v4 coarse	42.4	89.2
v5 coarse	28.4	75.5
v6 coarse	31.6	76.3
v7 coarse	30.3	76.0

6. References

[1] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, O. Teboul, D. Grangier, M. Tagliasacchi, and N. Zeghidour, "AudioLM: a language modeling approach to audio generation," *arXiv preprint arXiv:2209.03143*, 2022.