Whiskies : WHISper with Korean fIne-tuning and dEcoding Strategy change

Korea University COSE461 Final Project

Seokhee Kim
Department of Computer Science
Team 4
2018320138

HyeokJae Lee
Department of Computer Science
Team 4
2018320146

Yeahoon Kim
Division of Information Security
Team 4
2020330006

Seunghyun Park
Department of Computer Science
Team 4
2021100003

Abstract

Whisper is an automatic speech recognition pre-trained model and it achieved high robustness on multilingual dataset due to large audio training dataset . Although it shows high performance on many tasks, the model has some limitations such as getting stuck in generating repeated loops, passing the first or last few words, and transcribing text unrelated to the actual audio or hallucinations [1]. When fine-tuned with Korean daily speech (ksponspeech) dataset[2], the resulting model shows better performance than baseline model. When applying decoding strategy or fine-tuned with our Korean repeated dataset, problems such as hallucinations or repeated loop are improved in some cases. In this report, we shows that fine-tuning model on daily speech dataset and additional specialized dataset applying optimized decoding strategies can relieve some of above problems.

1 Introduction

Whisper model, which is OpenAI's Automatic Speech Recognition (ASR) model pre-trained on multilingual large audio dataset, achieved high performance and robustness similar as human [1]. However in some cases Whisper often outputs errors that humans would do not, such as getting stuck in generating repeated loops, omitting some words, transcribing unrelated text, or generating hallucinations. Among those errors, we especially handle the error of getting stuck in repeated loops which producing repeated sequence of words in some section of outputs. This type of error is one of the main problems that makes it difficult to apply the ASR model to real worlds because the model may produce unwanted texts.

In the process of resolving this error of repeating the wrong words/phrases without considering the context in model doing translating and transcribing, we thought that it would be possible to develop a model more suitable for real life through fine-tuning. Existing methods to solve this error are using Voice Activity Detection (VAD) or a decoding strategy.

The method using VAD is already running within Whisper itself. According to the Whisper 's model structure, it is possible to find the part that attempts to eliminate repeated loops and hallucinations by determining whether or not it is a real voice through the internally learned VAD. However, internal voice activity detection does not work in general cases. In order to compensate this problem, we can

use an external VAD module to determine if the voice is actually present. From a practical point of view, this approach is reasonable, but it cannot solve the fundamental problem and it is beyond Whisper's original architecture.

There is also a improvement method through decoding strategies.[1] There is no detailed mention, and there is only a comment it can be achieved in the inference time by adjusting *temperature*, *beam_size*, etc. It is difficult to confirm the tendency that a specific result could be derived through this, and achieve fundamental improvement through this.

In order to compensate for the limitations of these existing studies, we attempted two major methods based on the Korean dataset. Through Whisper fine tuning, we tried to learn Whisper-small model suitable for the Korean conducting below two tasks.

- Fine-tuning of the decoder
- Learning our generated dataset

2 Related Work

2.1 Fine-tuning

Fine-tuning can be performed by changing the value according to the basic description and model structure which is provided or using domain specific dataset. Details can be found in that official paper [1] and code.

2.2 Add Data for training

Adding new datasets can be viewed as a kind of fine-tuning technique in generative language models. When using this approach, it is necessary to additionally define a loss for supervised learning. In our work, since Whisper learns from the beginning with a cross entropy loss with the correct answer label, it was thought that additional loss would not need to be defined.[3]

2.3 Decoding Options for inference

2.3.1 Beam

In general, using *beam_ size*, which follows the FCFS heuristic in a first-come, first-served basis and stops when the size of the sequence set completed according to the step reaches the beam size, it is possible to provide flexibility in search depth and help establish a stopping criterion.[4] The introduction of the patience factor would reduce the inference speed. In addition, the beam option itself improves the generation performance of the model for machine translation and summarization by the patience factor, and at the same time slow down the generation.

2.3.2 Temperature

As one of the decoding options, it provides room to form different probability distributions in generations by changing the value of the temperature variable. The temperature value widely applied to text generation[5][6][7], we can improve the accuracy by this option. To use this options by our intention, top-p sampling and top-k sampling value should be considered together.[8][6]

3 Approach

Whisper is an ASR system learned through multilingual supervised data. It shows excellent performance for audio data including noise and accents through a large dataset. This Whisper model has tiny, base, small, medium, large versions depending on the size of the model. Due to the limitations of the resources, we conducted research on a Whisper-small model which has 244M parameters. Our goal is to create a robustness model for Korean.

According to Whisper's structure, sentences are generated in the decoder based on the results learned through the datasets in the encoder part. The performance can be improved by changing or adding the training dataset or tuning various parameters of the transformer.

3.1 Adding our generated dataset

We attempted additional fine-tuning process through various Korean dataset to improve the model into a Korean-specific model. In particular, in order to distinguish between cases where repeated loops occur and cases where repetition occurs in real data, we trained pre-trained model on Korean daily speech dataset and the dataset which we made. Through this, we hoped that the model could distinguish between repeated loop and repetition.

3.2 Tuning decoder strategies

We modify some decodeing strategies to make more correct results for Korean dataset. We went through the process of tuning the parameters of the decoder in the model, and improved the model by finding and applying an appropriate option at infernece time.

3.3 Relieving repeated loops

There are only discussions among users on official OpenAI/whisper github page, but nothing is clearly known about why the repeated loop occurs in whisper and the conditions for it. In a similar translation part, repeated examples due to the greedy method can be identified[9], so analysis and hypotheses were established based on this. Whisper was used to analyze some cases in which repeated loop occurred, and after establishing several hypotheses, data set was collected and conducted the experiments based on these hypotheses.

The repeated loop mainly occurred when there was a part where no voice in the middle or when an inaccurate voice that the tokenizer could not recognize came out. Since the probability of the next token coming was incorrectly measured, we judged that there was a high possibility that a repeated loops would occur as the token was inserted incorrectly. Based on this, we categorized data that may occurs repeated loops in result:

1. Speech speech with many words that are not considered to be included in the training data
2. Speech voices in which it is difficult to grasp the structure of the sentence due to the incorrect order of words in the sentence or exclamations
3. Spoken voices that actually say repeated words or sentences
4. Voices without specific speech and thought to be noise
5. Speech in a room with background noise or music
6. A voice that begins to speak after not speaking for a certain period of time

Also we considered loops are defined for comparing the performance of the improved model as an objective indicator. We suggests a new metric that measures the degree of loops and performance was compared between models.

Our generated dataset has examples that contains sentences that does not make sense, sentences that does not make sense and have repeated loops, sentences that makes sense and have repeated loops. Through fine-tuning these datasets we expected that the model can discriminate between false repeated loops and true repeated loops like training false knowledge on large language model [3].

4 Experiment

In this section, we experimented with three things: fine-tuning, finding suitable decoder options for inference, and learning a new dataset. As a baseline for comparison and experimentation, OpenAI's Whisper small model with 244M parameters was used for the efficiency of learning. WER and CER, which are often scales for evaluating the ability of a language model to transcribe, were used to check the degradation and improvement of the model's performance. For the repeated loop targeted in this experiment, the methods performed were evaluated by measuring error improvement through the proposal of a new metric.

4.1 Dataset

For selecting the most appropriate dataset, we tested several fine-tuned models using single or combined public Korean dataset including zeroth, kssdataset, common_voice_13, etc. However after investigating the output of those models, we concluded that those dataset are suitable for only news or official speech situation but not for daily situations. To overcome this problem, we used ksponspeech train dataset that has several subjects including various daily situations [2]. In addition to this, in the process of performing additional adjustments on a fine-tuned model, we used our dataset (Appendix B). Our dataset is constructed with special patterns that any other public dataset does not includes. After creating golden texts, a single person recorded it with correct pronunciation in a noise-free place and use it for fine-tuning.

During the evaluation, the widely used metrics, WER and CER, were used and for dataset ksponspeech validation dataset and zeroth-korean test dataset were used. This evaluation dataset was not used in the learning mentioned above. In this dataset, consisting of less than 30 seconds of repeated loop evaluation, repeated loops do not occur enough to experiments, so we need to find additional appropriate voice data and measured the part related to the repeated loop. Since in this experiment, golden text is not requied so videos are randomly selected from YouTube based on six types as we suggested above. Those videos can be found in the Appendix C

4.2 Evaluation method

A text normalizer preprocesses data for reliable training. This prevents the WER from fluctuating by removing special symbols, converting numbers to letters and letters to numbers, etc. The normalizer provided by Whisper paper was designed for multilingual data as stated in the appendix of the Whisper paper, so it is not suitable for Korean. For our experiment, we developed the zeroth-normalizer python package based on the codes in "Project: Zeroth" and used it as a measure.

4.2.1 Word Error Rate & Character Error Rate

Word Error Rate (WER) and Character Error Rate (CER) is a general metric for measuring speech recognition and machine translation system performance. WER is calculated based on words and CER is calculated based on characters. Especially, CER is useful for comparing different models for tasks for multilingual dataset where the diversity of languages makes WER unsuitable.

4.2.2 Repeated Loop Rate (RLR)

We defined loop as 'repetition of characters, words, or substrings that do not exist in actual voice data' in the absence of golden text.

$$\text{Repeated Loop Rate (RLR)} = \frac{\text{number of loops}}{\text{number of characters}} \times 100 \tag{1}$$

This is inversely proportional to the length of the character, and is proportional to the number of occurrences of the loop. Our goal is to reduce the number of loop occurrences, and the longer the total sentence length, the more repeated loops occur, so we divided it by the total length of the loop. If the number of characters in the two sentences is the same, the larger the number of loops, the larger the result value. If the number of loops is the same, the larger the total number of characters, the smaller the result value.

Although this is not an accurate metric. It has the disadvantage that the result value is measured well if there's no predicted text - no written text. In our experiment, we determined that the metric had sufficient persuasiveness, since most of experiment results does not have errors in which words were not written. See Appendix D for detail pseudo-code.

4.3 Experiment Details

For simplified notation, we will state **baseline** model as Whisper-small pre-trained model. In order to analysis our approach's result, we fine-tuned **baseline** model in two steps. **Step1** model is fine-tuned with ksponspeech dataset on **baseline** model and **step2** model is fine-tuned with our generated dataset on **step1** model.

4.3.1 Experiment with decoding option

To find out the best decoding strategy that can achieve best WER score, we tests several combinations of options with **step1** model on unseen dataset (zeroth-korean). The options we tested are related to decoding strategies of the transformer architecture: *num_beams*, *num_beam_groups*, *penalty_alpha*, *early_stopping*, *temperature*, *repetition_penalty*, *diversity_penalty*, *epsilon_cutoff*, *eta_cutoff*

4.3.2 Model fine-tuning

All fine-tuning training was conducted with a single Nvidia RTX2080 GPU. The code used for fine tuning was modified based on the this *huggingface*'s article.

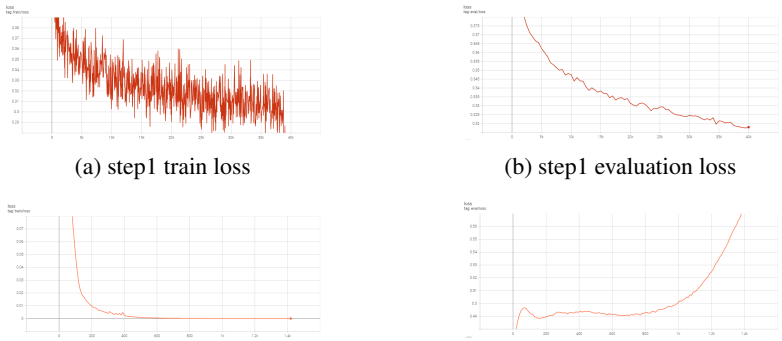
In **step1** fine-tuning step which used large dataset, we set relatively large learning rate on training as $1e-5$. For optimizing factors, we used PyTorch's AdamW optimzer with $\beta_1 = 0.9, \beta_2 = 0.98$. For regularization factor, we set weight_decay as $1e-5$. In **step2** fine-tuning step which used our generated dataset, we set relatively small learning rate on training as $1e-6$. For optimizing factors, we used PyTorch's AdamW optimzer with $\beta_1 = 0.9, \beta_2 = 0.98$. For regularization factor, we set weight_decay as $1e-6$. In both fine-tuning processes, to prevent the model from overfitting to the characteristics of the dataset, the encoder weights are frozen and only the decoder weights were trained.

Among the checkpoints obtained as a result of each steps, the checkpoint that shows the best result combining WER score and inference results which measured by heuristically was used for each subsequent experiments.

4.4 Experiment Results

4.4.1 Fine-tuning

See Figure 1.



(a) step1 train loss

(b) step1 evaluation loss

(c) step2 train loss

(d) step2 evaluation loss

Figure 1: fine-tune results of **step1** and **step2**

According to the result combining WER score and output texts, we choose checkpoint 9500 for **step1** and checkpoint 150 for **step2**.

4.4.2 Decoding Options

See Table 1.

4.4.3 Evaluation Metrics

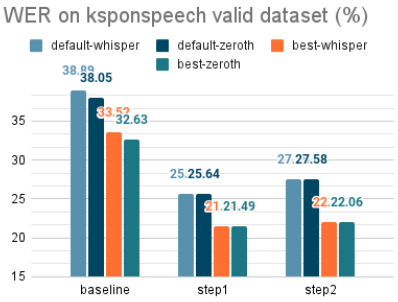
See Figure 2.

4.4.4 Repeated Loop Metrics

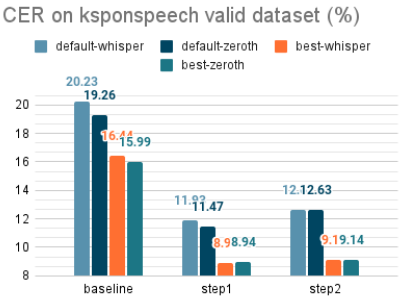
See Figure 4.

	Default	Best
num_beams	1	4
num_beam_groups	1	2
penalty_alpha	None	0.0
early_stopping	False	True
temperature	1.0	0.0
repetition_penalty	1.0	1.4
diversity_penalty	0.0	0.0
epsilon_cutoff	0.0	0.0
eta_cutoff	0.0	0.0

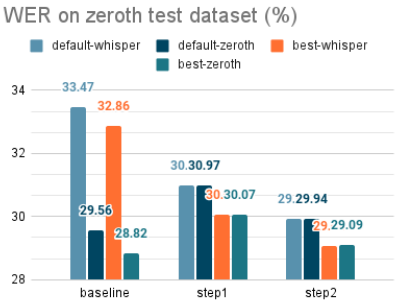
Table 1: Decoding Stategies



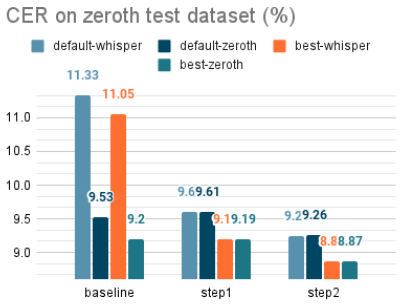
(a) WER on ksponspeech validation dataset



(b) CER on ksponspeech validation dataset



(a) WER on zeroth test dataset



(b) CER on zeroth test dataset

Figure 2: Evaluation Metrics of baseline, **step1** and **step2** models

(a) *default* and *best* indicates the decoding strategy and *ksponspeech* and *zeroth* indicates the dataset name on which evaluation executed

4.4.5 Performance in Real World Examples

Case A. Performance Improved See Table 2. For compact view, the models' results are redacted with following rules: the repeated sequences are replaced with **<REPEAT>** and long texts with no importance are replaced with **<SKIP>**. See also Appendix E for more results.

Case B. Performance Worsen Overall performance worsen with fine-tuned models with some examples in Type6. See Appendix E for more details.

Case C. Performance Unimproved There is no noticeable differences in result with examples in Type1. See Appendix E for more details.

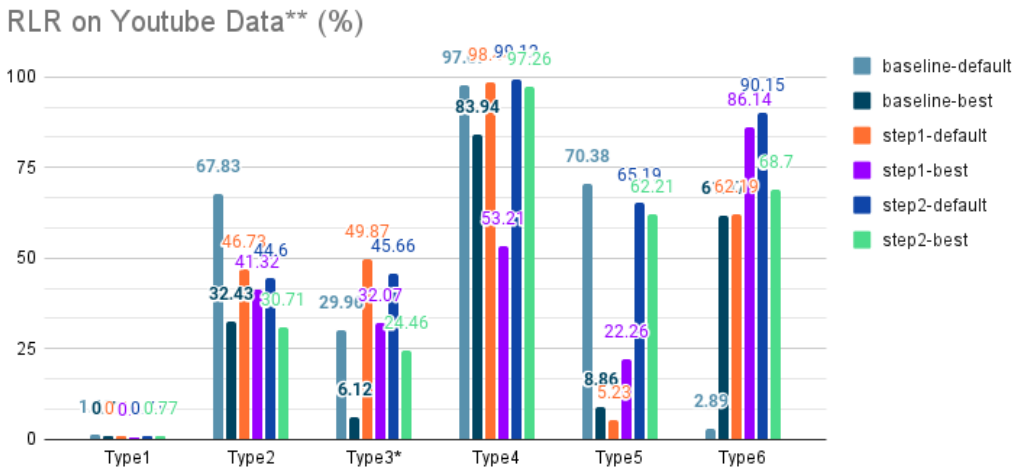


Figure 4: RLR of each model with various decoding strategies

(a) *default* and *best* indicates the decoding strategy

(b) *Type3 consisted with 'Spoken voices that actually say repeated words or sentences', so in this type, higher is better

(c) **See Appendix C, E for more details

Type	baseline-default	baseline-best	step1-default	step1-best	step2-default	step2-best
Type3.6	낯선자가 나가자	낯선자가 나가자	여자 친구 있어요? 아니 없어요. 없어요? 없어요. 없어요? 아니 없어요. 아 있었는데 아니 없어요 그냥. **<SKIP>** 낯선 자가 나가자 난 얘기는	여자 친구 있어요? 아니 없어요. 없었어요? 없어요. 없었어요? 아니 없어요. 아 있었는데 아니 없어요 그냥. **<SKIP>** 낯선 자가 나가자 난 얘기는	여자 친구 있어요? 아니 없어요. 없어요? 없어요. 없어요? 아니 없어요. 아 있었는데 아니 없어요 그냥. **<SKIP>** 낯선 자가 나가자 난 이 근	여자 친구 있어요? 아니 없어요 없어요 없어요 없어요 아니 없어요 아 있었는데 아니 없어요 그냥 **<SKIP>** 낯선 자가 나가자 난 얘기는
Type4.1	아오케! **<REPEAT>** 7번 달아보소! **<SKIP>** **<REPEAT>** 1번째 선수는? **<REPEAT>** 감사합니다! **<REPEAT>** 랩	아오케! 아오케! 아아 아 7번 달아보세요! 2라운드 **<REPEAT>** **<REPEAT>** 감사합니다! 아 아 아 아 아 아 아	뭐야. **<REPEAT>** 어유크룸 크룸뭐야 **<REPEAT>** 어너끝났어.	뭐야.어 유쿠쿠어. 어어어야끝났어.	뭐야 뭐지? **<REPEAT>** 어유급급 **<REPEAT>** 어어 **<REPEAT>** 어끄사	아이 어 어 어 어 **<REPEAT>** 유급급 **<REPEAT>** 어어 **<REPEAT>** 어야끝내라 **<REPEAT>**

Table 2: Performance improved cases

5 Analysis

As a result of fine-tuning in **step1**, both WER and CER were improved compared to the **baseline** model. In addition, for the six types of data, the prediction performance was improved for some types compared to the **baseline** model. In particular, the **step1** model predicted Type3 and Type4 data more accurately than the **baseline** model, but for Type6 data, the results were worse than the **baseline**. This suggests that the accuracy of words and sentences increases as we train more Korean data, but the performance of data without speech decreases as it is not trained.

After fine-tuning in **step2**, the WER and CER improved compared to the **baseline** model, but the performance was not as good as the **step1** model. The reason for the decrease in performance is that the original dataset was trained with normal sentences, but since the model was trained with abnormal sentence data, the prediction performance for normal sentences may have decreased. Therefore, looking at the Type2 data, the repeated loop metric of **step2** is lower than that of other models, but it shows good recognition of abnormal sentences with many exclamations.

We can also see that the accuracy of the model increases by changing the decoding option, as the values of WER and CER decrease. This is because the model is able to check a wider range of cases by increasing the number of beams to 4, whereas before it was greedily predicting with 1 beam. Also, by increasing the value of *repetition_penalty* and setting *early_stopping* to True, we can see that the repeated loop metric decreases in most cases. However, we can see that the performance is worse for the Type5 and Type6 data in the **step1** model. The decoding model considers many cases as the number of beams increases, so it seems that the prediction performance decreases because it predicts background music or noise as speech that was not previously treated as speech. Since the Whisper model processes the MEL spectrogram as input along with the previous output, unlike the existing language models, there seems to be a limit to the variance of decoding options that only affect the output.

6 Conclusion

In this paper, the repeated loop was improved in some situations through fine-tuning and decoder option change based on the baseline model of whisper. Despite the existence of similar errors in language models of various types and sizes, the persistence of inconvenient conditions, such as users arbitrarily adjusting options or attaching additional modules, was especially noticeable. It is presumed that related research is too difficult or that it is difficult to find examples that apply to actual models. We couldn't expect breakthrough performance, but we pointed out that where we could improve the robustness by focusing a little more on the problem discussed in the github discussion. At the same time, when a problem such as repeated loop occurs in 'Korean Daily Conversation', other users may have experienced the above problem by suggesting a method for changing transformer options and fine tuning, and adding opinions on the relevance of functions. It can be said that one method that can be tried is presented more clearly. Of course, the change in WER and CER showed that the improvement was not large compared to the existing model, but It can be seen that a great achievement was achieved by improving the repeated loop, which we tried to focus on.

It is estimated that better experiments and results can be obtained in situations that are more free from resource constraints, such as learning on a sufficient Korean data set and experiments combining encoders and decoders. Our approach, such as the presentation of metrics that evaluate trivial but potentially fatal errors, presents a new direction for evaluating the performance of large language machine models, and at the same time, the process of fine-tuning the model for users' convenience would be a stepping stone to find.

References

[1] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. 2022.

[2] J.-U. Bang, S. Yun, S.-H. Kim, M.-Y. Choi, M.-K. Lee, Y.-J. Kim, D.-H. Kim, J. Park, Y.-J. Lee, and S.-H. Kim. Ksponspeech: Korean spontaneous speech corpus for automatic speech recognition. *Applied Sciences (Switzerland)*, 10(19):1–17 – 17, 2020.

[3] Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. 2019.

[4] Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. Beam decoding with controlled patience. 2022.

[5] Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. 2017.

[6] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. 2018.

[7] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. 2018.

[8] Alec Radford, Rewon Child Jeffrey Wu, Dario Amodei David Luan, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[9] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. 2019.

A Appendix: Team contributions

We worked to ensure that our contributions to the code and report were comparable, although we each had different areas of focus, such as choosing topics and fine-tuning whisper-small, changing decoder options, and summarizing references. There were also (unsuccessful) learnings and attempts that were not covered in the paper, but we each shared our progress at each meeting, and then divided up the roles for the next tasks. The parts that each person mainly played are as follows.

Seokhee Kim: Developed 'zeroth-normalizer' python package based on 'Project-zeroth' codes. Recorded 'Our generated dataset'. Fine-tuned various models especially on ksponspeech dataset. Provided cloud storages for model and experiment result sharing. Write mainly experiment detail and result parts of this paper.

HyeokJae Lee: Presenting the need to freeze the encoding. Show how to adjust the parameters and share the changed performance. Providing resources for model experimentation.

Yeahoon Kim: Created an algorithm to find the appropriate decoding option and compared the performance between models by creating and implementing an RLR Metric. Wrote mainly approach part of this paper.

Seunghyun Park: Tried fine-tuning by establishing a decoding strategy by referring to various papers and references. Tried tokenizer transformation and inference. Wrote overall basic format of the report (draft of Abstract, 1 4.2, 6).

B Appendix: Our generated dataset

For simplified view, it shows one example per types.

Type	**Transcript**
Sentences that makes sense but has repeated sequences	간장 공장 공장장은 강 공장장이고 된장 공장 공장장은 공장장이다.
Music lyrics that has repeated sequences	난 너를 사랑해 난 너를 사랑해 난 너를 사랑해 난 너를 사랑해 난 너를 사랑해 난 너를 사랑해 따스한 그대의 손길로
Sentences that contains a character repeated	아 아
Sentences that does not make sense and has repeated sequences	감사합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다 합니다

Table 3: Our generated dataset examples

C Appendix: Categorized Youtube Datasets

Category	Youtube Link
1. Speech speech with many words that are not considered to be included in the training data	https://youtu.be/fY3SWqG_QO4 until 01:00 https://youtu.be/eNLJVwO8X7U https://youtu.be/IiEFOjTFEKk https://youtu.be/T7h8O7dpJIg
2. Speech voices in which it is difficult to grasp the structure of the sentence due to the incorrect order of words in the sentence or exclamations	https://youtu.be/Bxm22mho9Ps https://youtu.be/rLueTjLWVCc https://youtu.be/VM49h7CsawA until 04:20 https://youtu.be/WGGEBz2DzXk
3. Spoken voices that actually say repeated words or sentences	https://youtu.be/8lCDRJxK-H0?t=37 from 00:37 until 01:10 https://youtu.be/of9ZfPKQiDA from 09:50 until 10:10 https://youtu.be/f2TrOKUUuVQ https://youtu.be/ptCIhrBnWn8 https://youtu.be/ISgdFbko82U https://youtu.be/NhCkgYPbwj4 from 01:08 https://youtu.be/qFV-eZYZZyk
4. Voices without specific speech and thought to be noise	https://youtu.be/Msro-YJGFtk until 03:00 https://youtu.be/4aXuJ2omHvY until https://youtu.be/TbxCpMJR0iM from 03:00 until 06:00
5. Speech in a room with background noise or music	https://youtu.be/M8HU5qJhJ1Q https://youtu.be/oR2WgpuTtbc until 00:46 https://youtu.be/QFYuz9Buwrk https://youtu.be/3IZOf7KVJDw
6. A voice that begins to speak after not speaking for a certain period of time	https://youtu.be/XUuSxAC5tx8 from 00:08 until 01:00 https://youtu.be/XUuSxAC5tx8 from 02:50 until 03:19 https://youtu.be/XUuSxAC5tx8 from 08:23

Table 4: Categorized Youtube Datasets

D Appendix: Pseudo-code for RLR

Algorithm 1 Find number of loops

```
procedure FINDNUMBEROFLOOPS(inputStr)
    cnt ← 0
    for i from 0 to length of inputStr do
        for j from 1 to min((i+1)/2, 100)+1 do
            flag ← True
            for idx from 1 to j+1 do
                if inputStr[i-2*j+idx] == inputStr[i-j+idx] then
                    flag ← False
                    break
                end if
            end for
            if flag then
                cnt ← cnt + 1
                break
            end if
        end for
    end for
    return cnt
end procedure
```

E Appendix: Results in Real World Examples

You can see the results in this github repository.