VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Machine Learning Assignment

Bahnaric Phoneme Segmentation

Semester: 232

Advisors: Dr. Nguyen Duc Dung
Students: Vo Hoang Nhat Khang - 2152646
Ta Gia Khang - 2152642

HO CHI MINH CITY, MAY 2024

Contents

1 Motivation

The Bahnaric language stands as a vital cultural cornerstone within its ethnic community, encapsulating centuries of heritage, identity, and shared experiences. Yet, despite its intrinsic significance, the linguistic landscape of Bahnaric remains underrepresented in modern communication technologies. This disparity not only impedes the seamless exchange of ideas and knowledge within the Bahnaric ethnic group but also restricts meaningful interactions with other linguistic communities.

Recognizing the transformative potential of technology, our objective is to empower Bahnaric language speakers by bridging this digital divide. Our mission extends beyond mere communication; it seeks to foster a sense of belonging and empowerment within the Bahnaric community, while also facilitating cross-cultural dialogue and understanding.

2 Background

2.1 Decision Tree

A decision tree is a supervised learning algorithm used for both classification and regression tasks. It recursively splits the data into subsets based on the most significant attribute at each node, resulting in a tree-like structure where each internal node represents a decision based on an attribute, and each leaf node represents the outcome.

2.1.1 Formulation

Let's consider a decision tree for a binary classification problem. Suppose we have a dataset consisting of N samples with D features. The decision tree seeks to partition the feature space into regions $R_1, R_2, ..., R_J$ such that the majority of samples in each region belong to a particular class.

At each node of the decision tree, a splitting criterion is used to determine the best feature and threshold for partitioning the data. One common splitting criterion for classification is the Gini impurity, defined as:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

where t represents a node, c is the number of classes, and $p(i|t)$ is the probability of class i at node t.

The Gini impurity is minimized when the node is pure (i.e., contains samples from only one class). Therefore, at each node, the algorithm searches for the feature and threshold that minimize the weighted sum of the Gini impurities of the child nodes.

The splitting criterion can also be based on other metrics, such as entropy or information gain. For example, the entropy of a node t is given by:

$$\text{Entropy}(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t)$$

where $p(i|t)$ is the probability of class i at node t.

Once the tree is built, it can be used to make predictions by traversing the tree from the root node to a leaf node based on the values of the input features. Decision trees are prone to overfitting, especially when they are deep and complex. Regularization techniques such as tree pruning, limiting the maximum depth, or setting a minimum number of samples per leaf can help mitigate overfitting.

2.2 Gradient Boosting

Gradient Boosting is a powerful machine learning technique used for both classification and regression tasks. It is an ensemble learning method that combines the predictions of several weak learners

(typically decision trees) to produce a strong learner. The core idea behind Gradient Boosting is to iteratively train new models that predict the residuals (the differences between the actual and predicted values) of the previous models and then combine these models to make the final prediction.

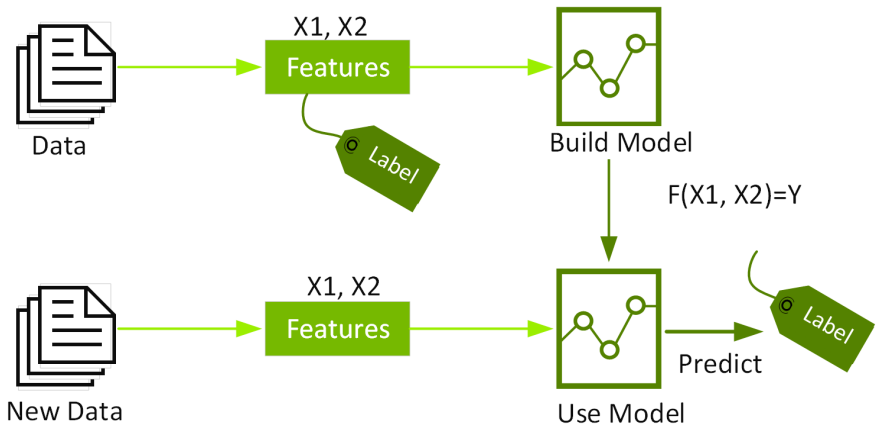


Figure 1: Finding patterns in dataset with labels

2.2.1 Algorithm Overview

1. Initialize the model with a constant value, usually the mean of the target variable for regression or a probability for classification.
2. For each iteration:
   (a) Train a new weak learner on the residuals of the previous model.
   (b) Compute the predictions of this new model and add it to the ensemble, with a weight determined by a learning rate parameter.
3. Repeat the above steps until a predefined number of iterations is reached or until a certain threshold of improvement is achieved.

2.2.2 Key Components

- **Loss Function:** The loss function measures the discrepancy between the predicted and actual values and guides the optimization process. For regression tasks, a common loss function is the mean squared error (MSE), defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N is the number of samples, y_i is the true target value, and $\hat{y}_i$ is the predicted target value.

For classification tasks, the cross-entropy loss is often used. For binary classification, it is given by:

$$\text{Cross-Entropy} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right)$$

where p_i is the predicted probability of class 1 for sample i.

- **Weak Learner:** Gradient Boosting typically employs decision trees as weak learners. A decision tree predicts the target variable by recursively partitioning the feature space into regions and assigning a constant value to each region. The output of a decision tree can be represented as:

$$f_m(x) = \sum_{j=1}^{J} c_{mj} \cdot I(x \in R_{mj})$$

  where $f_m(x)$ is the prediction of the m-th decision tree, R_{mj} is the j-th region of the feature space, c_{mj} is the constant value assigned to that region, and $I(\cdot)$ is the indicator function.

- **Learning Rate:** The learning rate controls the contribution of each new model to the ensemble. It is denoted by η and typically ranges between 0 and 1. The update rule for the prediction at each iteration m is given by:

$$\hat{y}^{(m)} = \hat{y}^{(m-1)} + \eta \cdot f_m(x)$$

  where $\hat{y}^{(m)}$ is the ensemble prediction after adding the m-th weak learner $f_m(x)$.

- **Regularization:** Regularization techniques are essential for preventing overfitting in Gradient Boosting. One common approach is tree pruning, which involves removing branches from the trees that do not significantly improve performance. Another regularization technique is shrinkage, which scales the contribution of each weak learner by a small factor γ, reducing the impact of each individual tree on the final prediction.

2.3 Bahnaric phoneme

- The phoneme sample comprises individual words, with each word pronounced by a native speaker. The onset and offset times of each phoneme are denoted by the labels '`ov`' and '`op`' respectively.
- Our problem can be summarized as:
  - **Input:** An audio clip and a list of given phonemes in that file.
  - **Output:** Predict the start/end point for the phonemes's region, also using statistic to cut that region to each phoneme's region.

2.4 Feature engineering

From the input audio file, we choose a suitable frame size and then cut that audio into frames, the following features will be extracted from the audio frames[1]:

- **MFCC (Mel Frequency Cepstral Coefficients):** These coefficients collectively form an MFC. They are obtained from a cepstral representation of the audio clip, which is a nonlinear "spectrum-of-a-spectrum".
- **Zero Crossings:** This refers to the rate at which the signal transitions from positive to negative or vice versa.
- **Mel Spectrogram:** A Mel Spectrogram is a type of spectrogram where the frequencies are converted to the Mel scale.
- **Harmonics:** These are integer multiples of the base frequency in a sound. They play a role in the perceived timbre of the sound.

[1] sprocket: Open-Source Voice Conversion Software Kazuhiro Kobayashi, Tomoki Toda

- **Spectral Centroids:** These indicate the "center of mass" of the spectrum, helping to identify the brightness of a sound in digital signal processing.
- **Chromagram:** A chromagram visually represents the chroma (pitch class) of a signal. In music, the chroma of a note refers to its position within the octave of the twelve-note chromatic scale.
- **Tempo BPM (Beats Per Minute):** This measures the tempo of music, indicating the number of beats occurring in one minute.
- **Spectral Bandwidth:** This is the difference between the highest and lowest frequencies within a continuous band of frequencies. It can be utilized to identify the smoothness of a sound.

2.5 LightGBM

- LightGBM is a gradient boosting ensemble method that is used by the Train Using AutoML tool and is **based on decision trees**. As with other decision tree-based methods, LightGBM can be used for both classification and regression. LightGBM is optimized for high performance with distributed systems.
- LightGBM creates decision trees that grow **leaf-wise (best-first)**, which means that given a condition, only a single leaf is split, depending on the gain.

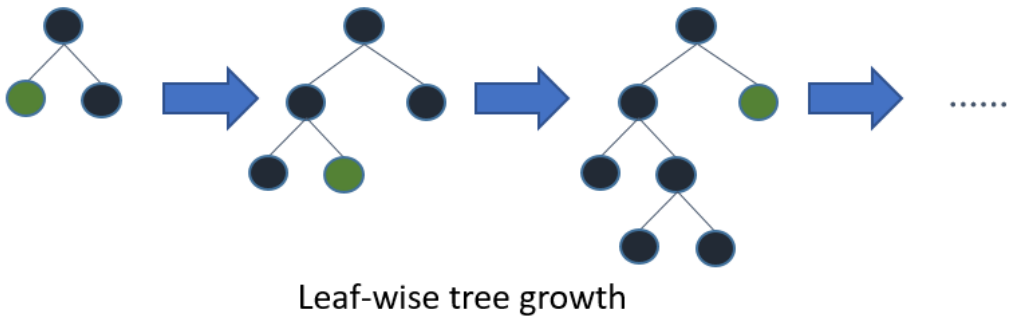


Figure 2: Leaf-wise growing

- Leaf-wise trees can sometimes overfit especially with smaller datasets. Limiting the tree depth can help to avoid overfitting. This method is differ from our tradition decision tree, which is **level-wise growing (depth-first)**.

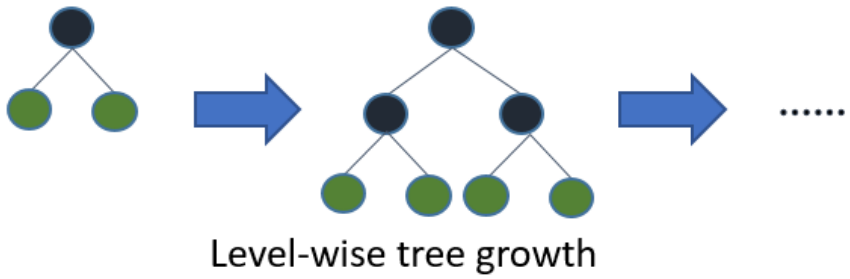


Figure 3: Level-wise growing

- With this problem, we proposed a training phase using LightGBM Classifier, as our result is just classifying if that audio frame is `ov/op` or not.

So as LightGBM gets trained much faster but also it can lead to the case of overfitting sometimes.

2.6 XGBoost

XGBOOST stands for Extreme Gradient Boosting. This algorithm is an improved version of the Gradient Boosting Algorithm. The base algorithm is Gradient Boosting Decision Tree Algorithm. Its powerful predictive power and easy to implement approach has made it float throughout many machine learning notebooks. Some key points of the algorithm are as follows:

- It does not build the full tree structure but builds it greedily.
- As compared to LightGBM it splits level-wise rather than leaf-wise.
- In Gradient Boosting, negative gradients are taken into account to optimize the loss function but here Taylor's expansion is taken into account.
- The regularization term penalizes from building complex tree models.

3 Method

3.1 Feature engineering

Pseudo code for feature engineering:

Algorithm 1: Feature Extraction from Audio Clips

```
foreach audio_clip in audio_clips do
    audio_clip_features = [];
    foreach frame in audio_clip do
        frame_features = [];
        foreach feature in acoustic_features do
            foreach window_length in range(85, 126, 10) do
                windowed_audio = get_window(frame, window_length);
                mean_value = calculate_mean(windowed_audio, feature);
                frame_features.append({"feature_name_window_length": mean_value});
            end
        end
        audio_clip_features.append({"audio_clip_name": frame_features});
    end
end
```

- We save each audio file's features in parquet format for later usage.
- An example of a audio clip's feature can be seen at: **Example features**

3.2 Classification Models

Gradient Boosted Machines and their variants offered by multiple communities have gained a lot of traction in recent years. This has been primarily due to the improvement in performance offered by decision trees as compared to other machine learning algorithms both in products and machine learning competitions.

Two of the most popular algorithms that are based on Gradient Boosted Machines are **XGBoost** and **LightGBM**. Our task is to classify if an audio frame is an `ov/op` or not, so we have experimented with both of them as a classifier, then have a look at the result and choose the one which is more suitable.

3.3 Phoneme Segmentation by Statistics

After getting `ov/op` label, we now have the length of the phonemes's region, our next step is using some statistic to get the time period of each phoneme in the given phoneme list by the following steps:

- From our dataset, we can extract the mean length of each phoneme, which is stored **Here**.
- From the given input, we will extract:
  1. Length of this word: T
  2. Mean length of each phoneme i in list of this word: t_i
- Then, the length of time period is given by:

$$L_i = \frac{t_i^* T}{\sum t_i}$$

4 Experiments

4.1 Training

- Each frame is treated as a data point, and the label is either 0 or 1.
- The extended labels are the neighboring frames of the `ov` or `op` labels.
- `LGBMClassifer` is used to train **two separate models** for the `ov` and `op` labels with the following parameters.

Table 1: Parameters for Gradient Boosting Classifier

Parameter	**Value**
Boosting Type	`gbdt`
Objective	`binary`
Metric	`auc`
Num Leaves	30
Learning Rate	0.01
Feature Fraction	0.9
Bagging Fraction	0.8
Bagging Frequency	10
Verbose	0
Num Threads	No. CPU cores

4.2 Ov/Op Prediction

- The trained model is used to predict the probability of each frame being labeled as `ov` or `op` .
- The only frame with the highest probability is selected as the `ov` or `op` label, we will get some neighbor frame of that frame to be the region of start/end of phoneme.

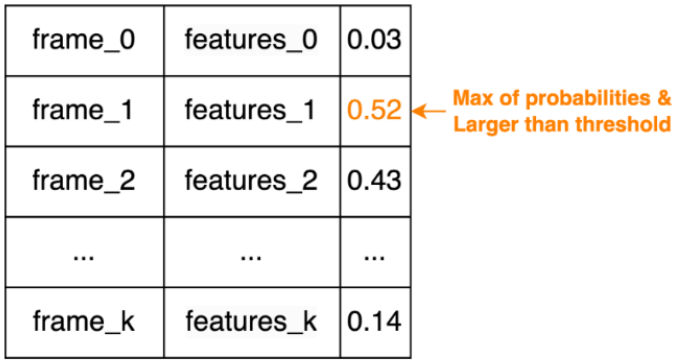


Figure 4: Prediction method

- From this frame index, to get the real time in time domain, we just multiply it with our chosen frame size, which is 5ms.

4.3 Phoneme Segmentation using Statistics

- The proposed formula is given in Section 3.3, and now we will perform an example on calculation for a specific word.
- Suppose phoneme b has mean length of 0.5s, and phoneme a has mean length of 0.4s. If the region time length of word with phoneme sequence `ba` is 1s, we can conclude that:

$$\text{Length}_b = \frac{0.5*1}{0.5+0.4} = 0.555$$
$$\text{Length}_a = \frac{0.4*1}{0.5+0.4} = 0.445$$

- Note that the mean length for each phoneme in the Bahnaric phoneme set is calculated based on a dataset comprised of multiple samples of each phoneme. This dataset is used for statistical analysis.

5 Experimental results

5.1 Ov/Op Prediction

The model is trained on 80% of the data and tested on the remaining 20%. The model achieves an accuracy of 0.67 and 0.79 for the `ov` and `op` labels respectively.

	Precision	**Recall**	**Accuracy**
`op`	0.78	0.38	0.67
`ov`	0.69	0.43	0.72

To compare two models: LightGBM and XGBoost for specific task as `ov` classifier, we got the result as below:

	Precision	**Recall**	**Accuracy**
`LightGBM`	0.69	0.43	0.72
`XGBoost`	0.79	0.35	0.66

5.2 Phoneme Segmentation

This is some experiment results that we have tried in our dataset, we use Praat application to visualize the result as below:

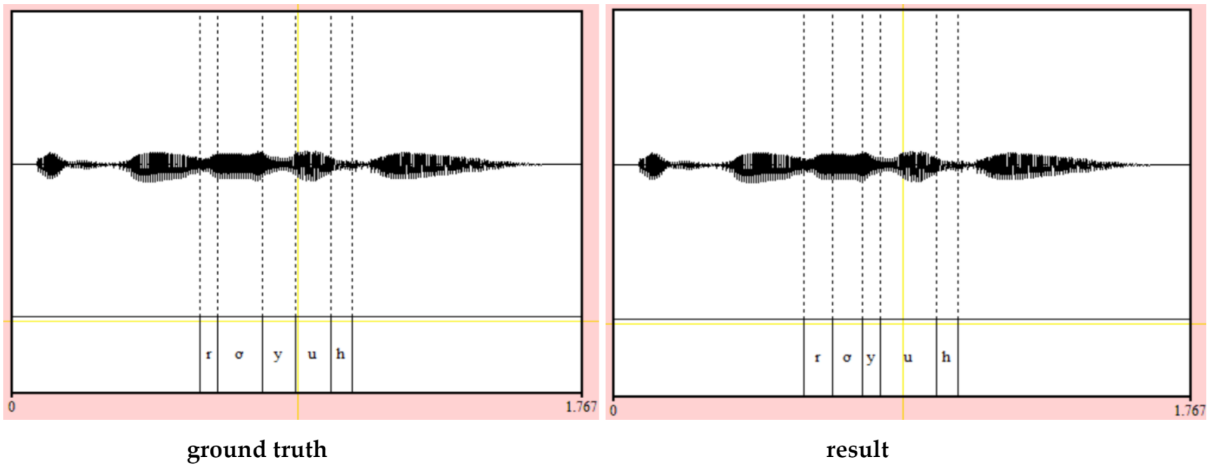


Figure 5: Word: `royuh`

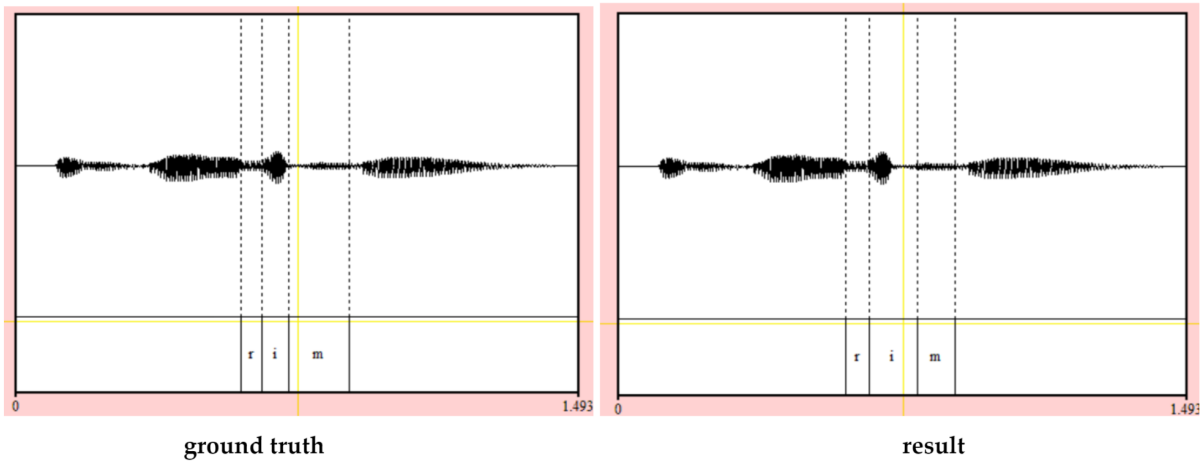
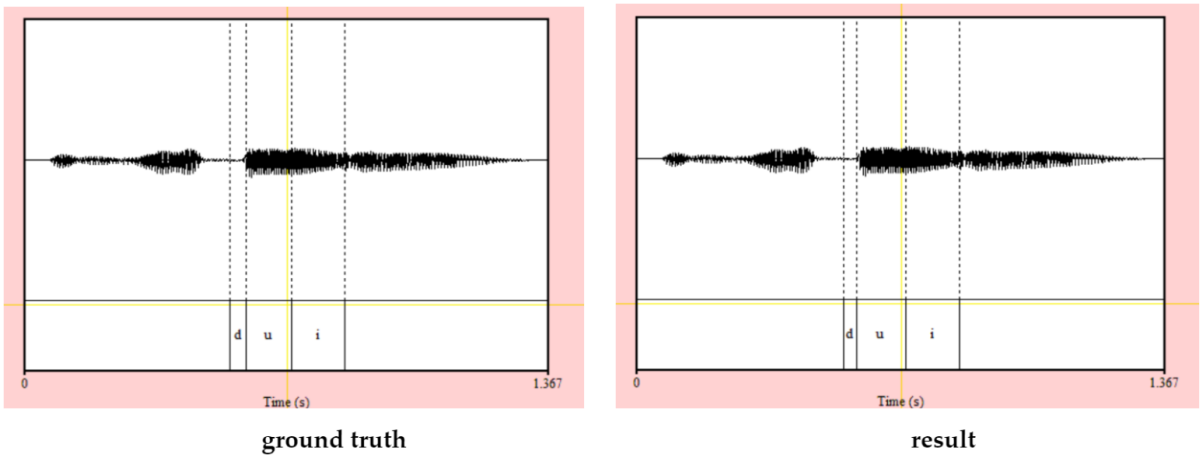
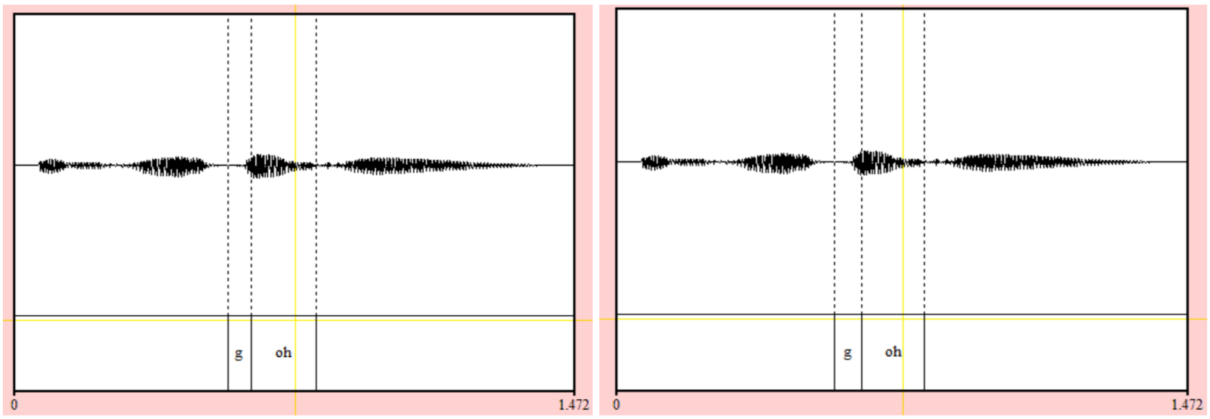


Figure 6: Word: `rim`



ground truth **result**

Figure 7: Word: `dui`



ground truth **result**

Figure 8: Word: `goh`

For more specific result, you can view more via this **Link**.

6 Conclusion

In conclusion, the Bahnaric language represents more than just a mode of communication; it embodies the rich tapestry of heritage, identity, and collective experiences of its ethnic community. However, despite its profound significance, the Bahnaric linguistic landscape has been largely overlooked in the realm of modern communication technologies. This omission not only hampers intra-community discourse but also obstructs meaningful interactions with other linguistic groups.

Throughout our experiment, we could developed a simple tool to take an Bahnaric word audio and return the audio zone for each phoneme, which can support other research activity in phoneme level. By leveraging technological advancements, we aim not only to facilitate seamless communication but also to foster a sense of belonging and empowerment within the Bahnaric community. Furthermore, our efforts are directed towards promoting cross-cultural dialogue and understanding, thereby fostering a more inclusive and interconnected society.

As we move forward, we would like to play with other more complicated algorithms for phoneme segmentation, such at DTW or trying with HMM. We hope that our experiment will be the beginning of the Bahnar phonological language research movement, also for Speech and Phoneme-level research.

For implementation, our repository for source code is public **Here**

References

[1] Kazuhiro Kobayashi and Tomoki Toda. "sprocket: Open-Source Voice Conversion Software ". In: *Proc. The Speaker and Language Recognition Workshop (Odyssey 2018)*. 2018, pp. 203–210. DOI: 10.21437/Odyssey.2018-29.