COMPUTER-ASSISTED JAPANESE PITCH ACCENT TRAINING

Nicolas Guillemot
University of Victoria

Honghu ”Ray” Lin
University of Victoria

Pablo Garcia Ledo
University of Victoria

ABSTRACT

Our project consists of implementing a computer system that helps those studying Japanese to practice their speaking skills. Specifically, we aim to help people improve their mastery of the Japanese pitch accent. The user will be prompted to say a word using a microphone, and this audio will be compared to the dictionary definition of the word’s pitch accent pattern. Using this analysis, our software will help the speaker understand how accurate their pitch accent is. This document further explains the details of this project, and explains our initial plan to execute it.

1. INTRODUCTION

Spoken Japanese associates one or more dialect-specific pitch accent patterns to each word, and these pitch accent patterns must be respected for the words to sound natural to native speakers. In some cases, the pitch accent can also affect the meaning of the word. Thus, practicing correct use of pitch accent is an important part of learning the Japanese language. Readers interested in further explanation of Japanese pitch accent are recommended to watch Dogen’s YouTube video series on the topic [7].

The most common way in which non-native speakers practice their spoken Japanese (assuming they do so at all) is by listening to native Japanese speakers and trying to imitate them as closely as possible. For example, one such specific technique is called ”Shadowing”, as proposed by Alexander Arguelles [6]. Shadowing suggests the student to repeat lines of dialogue in close synchronization with recordings of a native speaker, while simultaneously taking a brisk stroll.

Although speech practice techniques based on repetition are certainly helpful, they don’t give a way for the student to objectively verify the correctness of their pronunciation. For example, the student may not be aware of a nuance in the pronunciation of a word. Furthermore, humans perceive their own voice differently than outside listeners do, due to factors like the vibrations in the head caused by their voice, or due to wishful thinking about the sound of their voice. For such reasons, it can be important to record and listen to oneself to appraise one’s own pronunciation skills. However, in this project, we intend to go one step further, and use a computer system to perform this appraisal.

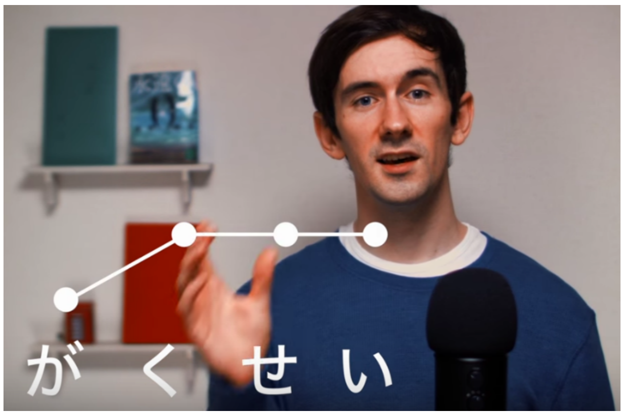

Figure 1. Dogen [7] illustrating the pitch accent for the word ”gakusei”, meaning ”student”. The word is split into four Japanese phonetic letters: ”ga-ku-se-i”. The ”ga” has a low pitch accent, while the ”ku”, ”se”, and ”i” have a high pitch accent. The diagram shown in this video frame capture serves as an inspiration for our user interface.

In this project, we seek to create a computer system which allows users to practice the correct use of Japanese pitch accent, which gives users feedback about their own performance. As feedback, the user sees how closely their own voice matches the correct pitch accent pattern, thanks to our program’s analysis of their voice. Using this system, we plan to make a proof of concept of training software for the Japanese pitch accent, allowing users to engage in highly productive practice sessions.

“Practice does not make perfect. Only perfect practice makes perfect.” - Vince Lombardi

2. RELATED WORKS

As well as the previously mentioned sources, there exist in-depth books on the topic of Japanese pitch accent, which are conveniently available in English [14] [17] [5]. Furthermore, information about pitch accent patterns of Japanese words are documented in several Japanese dictionaries [3] [1]. Conveniently, the ”Japanese Pronunciation / Pitch Accent” add-on [12] for the Anki flashcard software contains an XML dump of pitch accents for all words in NHK’s Japanese dictionary, which serves as the basis for the dataset of our project.

There exists previous research in the analysis of spoken Japanese and its use in developing Computer-Aided Language Learning (CALL) tools. For example, in 1984, Hiroya and Keikichi published an analysis of using the contour of the fundamental frequency (”F0”) to analyze Japanese sentences [8]. Using the contour seems like a promising approach, which we need to investigate further.

Analysis based on the fundamental frequency has been developed over many years in the domain of Japanese CALL research, leading to various extensions. For example, has also been much work in improving the sensitivity of error detection in the pitch accent analysis [13] [16] [9], and this research may help us understand the bounds of accuracy of our own system. Another interesting extension was shown in [10], where the speaker's mispronounced input audio is automatically corrected by the system and played back to the student in their own voice.

In article [4], Mitsuhiko Ota did autosegmental pitch analysis for Tokyo Japanese. He also talks about both lexical accent and phrasing in fundamental frequency. Moreover Mitsuhiko provides a fundamental frequency excursion formula which measures global pitch excursion. For our project we wish to understand when pitch accent would change in various speeches, and it is necessary to create frequency models for each pitch accent.

3. PROJECT DESCRIPTION

As an inspiration for the user interface, we consider the diagrams shown in Dogen's YouTube videos (see figure 1.) We aim to give the user a similar interface, which illustrates the correct pitch accent that the Japanese learner must try to emulate.

The learner makes an attempt to pronounce the word by speaking into a microphone, and our software performs analysis on this inputted audio, ideally in real-time. Through this analysis, the software determines the pitch accent pattern that the speaker used, and matches the speaker's audio sample to the word on the screen. The closer the speaker gets to the dictionary definition of the pitch accent, the higher their performance is scored.

3.1 Generation of Exercises

Pronunciation exercises could be automatically generated from the dataset available to our project. We aim to generate exercises using the pitch accent patterns that come from the NHK Japanese dictionary (downloaded as an XML file from the ”Japanese Pronunciation / Pitch Accent” Anki add-on [12].)

To use the NHK dictionary pronunciation dataset, we need to be able to interpret the various text qualifiers contained therein. For example, some regions of the text are marked as ”overline”, or ”nasal”, or ”nopron”. These qualifiers indicate transitions in pitch between high and low, as well as other nuances of the pronunciation. Somehow, we need to be able to interpret these specifications formally, then generate a model of the speech for the user to try to match.

3.2 Analysis of the User's Speech

After generating a model of the speech that the user should target, we need to perform some analysis on the user's voice. This analysis will be used to convert audio samples into a model that can be compared for accuracy with respect to the formal specification that was generated by parsing the text.

In Japanese language, pitch accent is categorized as High(H) and Low(L). In essence, it is significant for us to learn how pitch accent behaves in music retrieval perspective. Anne Cutler has a table of acoustic measures when syllables are spoken at different position in word for both H and L pitch [2].

3.3 Reference Audio Samples

Ideally, we should also have a reference audio sample for the user to try to emulate. This might be an audio sample of a native Japanese speaker pronouncing the word, or it might be the output of a Japanese text-to-speech program. Audio samples of native Japanese speakers can easily be found on Japanese dictionary websites, but it may be very difficult to match audio samples to the syllables in the text without resorting to manual work. On the other hand, text-to-speech software probably internally has this information, but we may not be able to easily access it. It will be necessary to do a deeper feasibility analysis to know how much of this feature is feasible.

3.4 Project Tasks

From the requirements described above, we can break down this project into tasks. This project has both a research component and an implementation component.

The research and requirements gathering component of this project consists of answering the following interrelated questions:

1. What are the formal rules for Japanese pitch accent?
2. Can dictionary forms be used to do audio analysis?
3. Can user audio be matched to the dictionary form?
4. What are the limitations of these procedures?

The implementation component of this project consists of implementing the following systems:

1. Exercise generation from dictionary definitions.
2. User speech analysis and accuracy scoring.
3. Stretch goal: Playing a reference audio sample.

Before engaging in the implementation, we should be able to translate the results of our initial research into a more precise specification for the software system.

Once we finish the implementation, we will do an evaluation of how well we achieved our initial goals. We could evaluate the accuracy of our system by testing it using audio clips of native Japanese speakers and audio clips of intentionally bad Japanese. In theory, the system should assign high scores to native speakers, and low scores to the intentionally wrong pronunciations.

As further evaluation of its usefulness, the system should be able to capture the nuances of an intermediate speaker. A user who only occasionally makes mistakes or is only slightly wrong in pitch should be able to identify these errors and improve their pronunciation. Determining whether the exercises have actually improved the user's pronunciation might require the subjective opinion of a native Japanese speaker.

We can predict several risks in this project that might negatively impact the final result. For example, one potential risk of this project is that we build a system that only works for the voices of a few individuals, or that requires difficult fine-tuning for each dictionary word we want to include in our exercises. Although such a system might be useful in limited situations, it would not be good enough for broad adoption among people studying Japanese.

A production-ready system should ideally work for a variety of voices, supporting both masculine and feminine voices. Various vocal ranges should be supported, whether tenor, bass, soprano, or alto. It would also be ideal to avoid hard-coding the software to assume that the user has e.g. an American English accent. It might be acceptable to omit the handling of childrens' voices, since people who study foreign languages are likely adult. However, that would be an interesting area of potential extension.

3.5 Tools

In early phases of the project, we will prototype our algorithms using productive scripting languages like MATLAB and Python. Once we develop a satisfactorily accurate algorithm, we may have to switch to a different language (eg: C# or C++) to develop a user interface that implements the algorithm in real-time with satisfactory visuals.

It may also be necessary to collect data from websites (eg: audio samples), in which case special-purpose tools may be developed.

3.6 Project Timeline

The project's timeline is split into the following phases. The details of these phases were explained in the previous sections.

1. Requirements gathering
2. Design
3. Implementation
4. Evaluation
5. Writing the final report

We have the following weeks to work on this project:

1. Week of October 16 (current day)
2. Week of October 23
3. Week of October 30
4. Week of November 6
5. Week of November 13
6. Week of November 20
7. Week of November 27

Examinations end on December 18th, so the exact final submission date depends on the grace of the professor. As such, this schedule may need to be reviewed once final project dates are determined.

We will allocate weeks 1 and 2 to the requirements gathering and design phase. This will give everybody the chance to study the relevant theory in-depth, and hopefully identify potential solutions to the challenges we face.

From there, we'll allocate weeks 3, 4, and 5 to the implementation and evaluation of the project. The implementation and evaluation phases are tied together, since we need a testing method to guide the design decisions in the implementation.

Finally, weeks 6 and 7 will be used to polish the project and write the final report. We probably don't need two weeks to write a report, but this extra time exists in anticipation of the project running late due to unanticipated challenges.

3.7 Role Assignment

3.7.1 Nicolas Guillemot

Nicolas is actively studying Japanese, which gives him some background to understand relevant research. Furthermore, as a graduate student, Nicolas probably has the most time to invest in this project (since this is his only course this term.) For these reasons, Nicolas will have an all-around focus in the project.

3.7.2 Pablo García Ledo

As a software engineering student, without previous experience in voice recognition or specific Japanese knowledge, Pablo will focus mainly on the implementation and the software design aspects of the project rather than research or the definition of the system capabilities.

3.7.3 Honghu "Ray" Lin

Honghu Lin came from China, whose native language and culture are quite similar to Japan. Honghu transferred from electrical engineering to computer science in fourth year, so he has a bit understanding of the material. In the future, he will mainly focus on design and evaluate project, and he will be doing implementation for project as well.

4. MIDTERM PROGRESS REPORT

At the time of the midterm, a basic user interface for the program has been written, and a way to design exercises from dictionary entries has been implemented. Work on the pitch accent detection is still underway. Although the infrastructure for the microphone recording is in place, the code that performs the pitch analysis is still in the research phase.

4.1 Generating Exercises

A file is used to specify all the words that should be used for exercises (as shown in Table 4.1), and they are displayed in the program along with their pitch-accent pattern (as seen in Figure 4.2.)

五つ	イツツ	
昨日	キノウ	0
匂い	ニオイ	
あなた	アナタ	1
明日	アシタ	0

Table 1. Sample data used to generate exercises. Each row has three elements: The word in kanji/kana, in kana-only, and optionally an index that disambiguates which entry in the dictionary should be used, for words with more than one entry.

4.2 Graphical User Interface

In the user interface, the same word is displayed in three different ways:

- In roman characters.
- In kana (Japanese phonetic alphabets.)
- In typical form, like with kanji (Chinese alphabet.)

The roman alphabet is for convenience of users who don't know how to read Japanese. Users who can read kana can get a more accurate representation of the pronunciation by reading the second line. Finally, the word is displayed in its natural written form, which might include a mix of hiragana and katakana (Japanese-origin phonetic alphabets) and kanji (Chinese-origin ideographic alphabet.)

At the top of the user interface, we can see the pitch-accent pattern of the current word. Each mora of the word is associated with one circle, placed either low or high depending on whether a low pitch or a high pitch should be used when speaking the word. At the tail, a hollow circle follows the word, to indicate the pitch that the following word should use.

As an example of the pitch being specified for the next word, consider the following common way to begin a sentence in Japanese: "⟨ X ⟩ **wa**". In this sentence, "**wa**" indicates that "⟨ X ⟩" is the topic of the sentence. The pitch of "**wa**" is defined by position of the hollow circle.

As a temporary feature, one can switch the current word using the left and right arrow keys.

4.3 Pitch-Accent Analysis

Other than the graphical user interface, the second half of the project is the pitch-accent recognition, and this work has yet to be integrated in the training software. It is currently being researched.

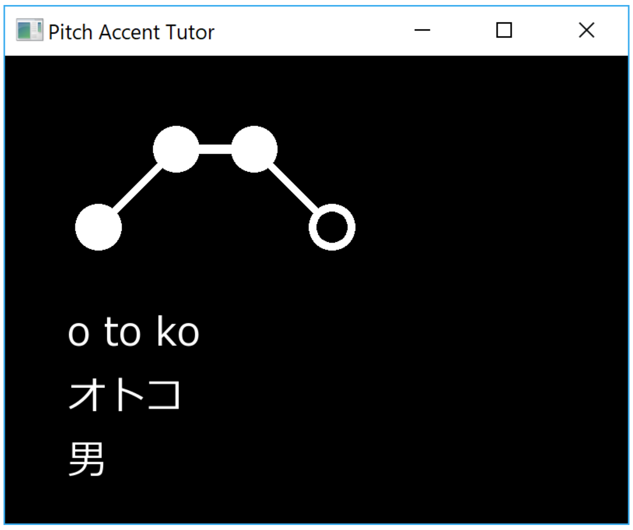

Figure 2. Sample image of our user interface for the pronunciation guide.

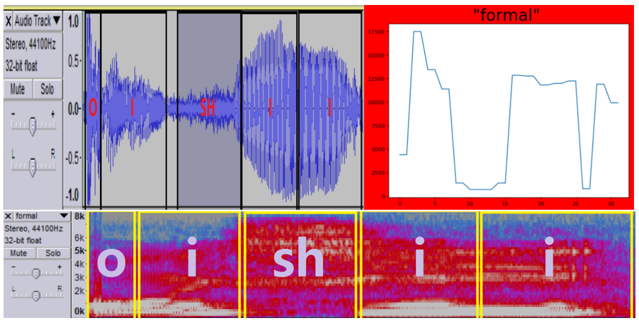

Figure 3. Top left image: Audacity graph for the Japanese word 'oishii'. Top right image: Autocorrelation for same audio. Bottom image: Audacity spectrogram graph for 'oishii'.

As discussed previously, Japanese pitch accents are formally classified into high and low pitches, and one of the main challenges of this project is to be able to map a user's speech to the corresponding pitch. To better understand how this can be done, we tried recording ourselves saying Japanese words, and analyzed the audio using autocorrelation with Audacity and Python code. Some results were within expectations, whereas some other results were surprising.

As an example of analyzing the pitch of a word, consider the following word: "おいしい" ("oishii" - "delicious"). The pitch accent pattern for this word is "o(L)-i(H)-shi(H)-i(H)". In Figure 4.2, the results of analyzing an audio sample from a native speaker are shown. By listening by ear in Audacity, we labeled different parts of the audio sample with the hiragana characters that are pronounced over time. It is noticeable that vowels tend to lie within lower frequencies.

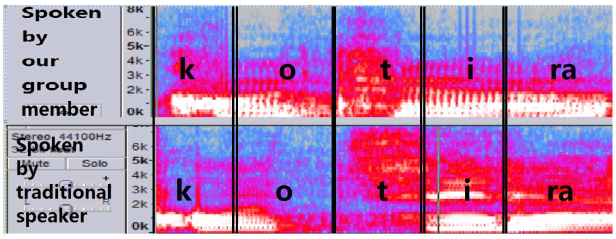

Figure 4. Spectrograms of Japanese word "kotira" for a native speaker and of one of our team members imitating it. Generated using Audacity.

Theoretically, the frequency of the vowel "o" should be lower than "i", yet if we take a look at the autocorrelation diagram of the top right image, "o" seems to have a higher frequency in the spectrogram. The reason for this phenomenon is due to a characteristic of Japanese phonetics, where vowels are sometimes pronounced in a "voiceless" manner [15], referred as devoicing. Hence, sometimes a hiragana that is labeled as high pitched, might actually be pronounced at a low pitch and in a very short period of time.

On the other hand, not only are individuals' voices diverse in timbre, frequency, and loudness, but their personal style of speaking also substantially affects the analysis of audio. For instance, words are usually pronounced with pauses and fermatas, but the duration for fermatas and pauses differs between people. For example, fermatas are often spoken with a higher pitch. Some people also occasionally swallow sounds at the end or in the middle of a word, which is troublesome for speech analysis.

To better understand the nuances between a native speaker and a non-native speaker, consider the spectrograms in figure 4.3. These spectrograms show the Japanese word "こちら" ("ko-ti-ra" or "ko-**chi**-ra") spoken by a native speaker and a non-native speaker imitating the voice. It can be observed that there exists some common frequencies and few slight differences by comparing the spectrograms. By listening to the audio and observing the spectrogram simultaneously, we learned what happen to "ko" is that the non-native speaker shortened the sound when they spoke. Therefore, as can be seen in the spectrogram, the vowel "o" has a different appearance compared to the pronunciation of the native speaker.

4.4 Post-Midterm: Next Steps

For the graphical user interface, the most important thing to do next is to give some visual feedback to the user about how well their pronunciation matches the official pitch accent pattern. From there, it's mostly polish.

For the audio analysis part of the project, we need to take what we learned about Japanese pitch analysis and use it to write C++ code that does pitch analysis in real-time based on the input from the microphone. We already have code that receives the microphone input, so it's mainly a matter of adding the code that does analysis on that data.

5. FINAL PROGRESS REPORT

Based on feedback from the midterm progress report, and based on the feedback from the final presentation, we decided to pursue research and development in the following areas:

- Improved pitch detection system.
- Initial pitch-corrected replay system.
- Improved visual feedback.
- Initial accuracy score system.
- Informal user studies.

In what follows, we describe the progress we made prior to writing this final report. The progress we made in each area of research and development we approached will be described and discussed, one-by-one.

5.1 Pitch-Detection

On the side of pitch detection, we undeniably acknowledged that doing a basic Japanese pitch analysis is not extremely difficult, however it is quite challenging to analyze the pitch accent with high accuracy. Therefore, we concluded that studying the underlying problem as well as understanding various methodologies and theories would be useful before moving further with the implementation.

We theorized that one of the main difficulties of analyzing Japanese pitch, in contrast to typical music or sounds, comes from the fact that most pitch detection algorithms stereotypically evaluate the frequency of a quasi-periodic signal. Throughout our pitch analysis, the program inputs are Japanese words that are composed and spoken shortly with various phonemes. Since these sounds themselves do not have repetitive periodicities (unless the user speaks multiples times into the microphone), the results might not be optimistic for some of the typical pitch-detection algorithms.

5.1.1 Chosen Solution

At the beginning of detection process, we implemented a simple discrete Fourier transform, just to observe various pitch behaviors when the program takes input. However, this algorithm is usually inaccurate, since the running time of algorithm is too slow to work correctly in real-time, resulting in outputs with a large delay. We saw greatly improved performance by upgrading our pitch detection from discrete Fourier transform to a fast Fourier transform (FFT). Naturally, this is due to the fact that the fast Fourier transform has a running time proportional to $O(nlog(n))$, in contrast to the DFT's running time proportional to $O(n^2)$.

5.1.2 DFT and FFT

Before using the FFT algorithm in practice, we sought to improve our own understanding of how the FFT algorithm works. In short, the FFT executes DFT algorithms on both odd and even samples of the input signal, and iterates in both parts. Even after integrating an FFT algorithm, we saw that there were still some inaccuracies in the results. The reason might be that the FFT can derive the spectrum only at certain frequencies (Multiples of Fs/N), yet the DFT can calculate spectrum at arbitrary frequencies.

5.1.3 Future Work

For future improvement, we would like to introduce more sophisticated approaches for pitch-detection. Namely, we would like to apply some autocorrelation algorithms, such as the Average Magnitude Difference Function (AMDF), or the Average Squared Mean Different Function (ASMDF).

Furthermore, with the feature of pitch-corrected replay in mind, we could test the effectiveness of the pitch-corrected replay by closing the feedback loop and running the pitch-detection algorithm on it. This could allow both systems to be used to mutually test each other. Running the FFT and inverse FFT on the audio might be another way to test the accuracy of our results. As future work, it would also be interesting to do study the differences of the effectiveness of pitch-accent detection for both native speakers and learners.

5.2 Pitch-Corrected Replay

Typical Japanese speech practice is done by repeating example sentences after a native speaker, but this approach has one main flaw: It forces you to try to imitate somebody else's voice, rather than teaching you how to modify your own voice. Naturally, this problem also affects pitch accent. To address this problem, we followed the lead of some previous research [10] by implementing a pitch-correcting replay system.

5.2.1 Chosen Solution

The pitch-corrected replay system is an addition to our software that records the user's voice, and attempts to synthetically correct the user's voice to make it sound *as though* they had used the correct pitch accent pattern. It then plays back the user's voice, to help the user notice the corrections they should make by changing the pitch of their own voice. This way, they may naturally better understand the difference between their pronunciation and the correct one.

Based on advice by George, we theorized that the pitch-corrected replay system should be possible to implement by using a phase vocoder. A phase vocoder performs FFT on overlapping windows with a smooth windowing function on the original recording, and synthesizes audio back using that information.

The phase vocoder can be used to modify pitch or speed of some audio independently, either by shifting FFT information some bins up or down, or by synthesizing more samples from each FFT frame than were contained by the window used to obtain that FFT frame.

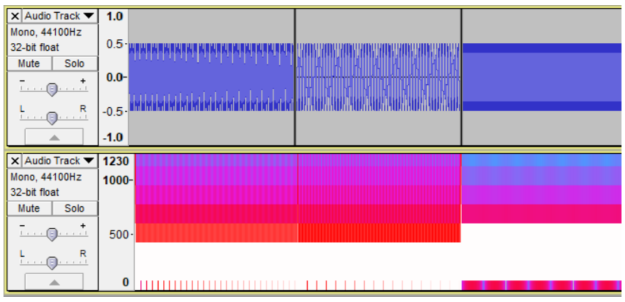

Figure 5. Sample of the way re-synthesized sounds look without interpolation or a windowing function. The continuous sudden changes cause a noticeable clicking noise.

5.2.2 Main Challenges

Given the educational nature of a project, we refrained ourselves from just finding an implementation on the Internet, which would have simplified the problem.

Since that the real-time FFT on recorded audio was already in-place and took a significant amount of processing time, we decided to reuse that information instead of taking more parallel FFT readings or increasing their frequency to get overlapping windows.

The use of overlapping windows with a smooth windowing function allows for much better synthesis quality; the smooth windowing function causes a smooth interpolation between frames of FFT information when the contents of the window overlaps are summed. For this reason, the audio playback could still be improved, since there is clicking between the results of applying the inverse Fourier transform to each FFT frame (see figure 5.2.2 with a sample of the clicking effect). We planned on solving this by manually interpolating between frames during synthesis. This hasn't worked so well so far, and causes recorded voice to be synthesized as an alien-like sci-fi sound, reminiscent of that produced by a Theremin.

The decision of the correct pitch to use is made by taking measures of the maximum and minimum pitch of non-silent FFT frames belonging to a given word, and linearly interpolating between them similarly to the way they are shown on-screen, allowing some space for constant pitch at the beginning of each syllable (see figure 5.2.2 with a visual example of the used pitch pattern). Unfortunately, this solution seems to find too low and high pitches which are frequently filtered and ignored by the pitch shifting algorithm, resulting in the recorded voice being played at erroneous pitches.

5.2.3 Future Work

Playing the synthesized voice without clicking noises can probably be achieved by either fixing the interpolation currently performed during it, or by introducing some other form of audio smoothing like a low-pass filter.

The decision of the pitch to use as a corrected example is also an important problem. Instead of picking the maximum and minimum pitch of the recording, something like

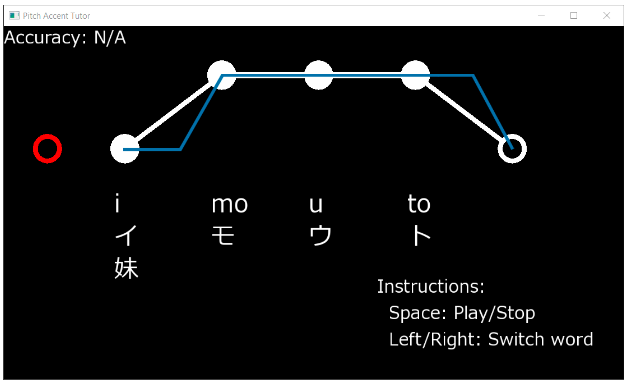

Figure 6. Example of correct pitch pattern used by the pitch correction algorithm. Plotted in blue over the application interface.

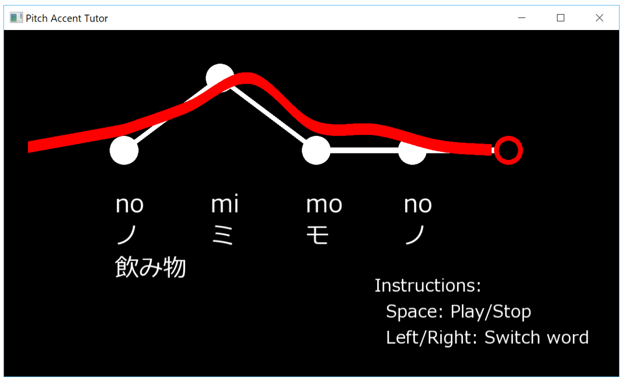

Figure 7. Sample image of our improved user interface with visual feedback.

the limits of the interquartile range of the pitches should be used.

Additionally, the algorithm is currently assuming that you are pronouncing the word in the exact expected amount of time that word should take to be pronounced at a given constant speed, but your voice could be slowed down or sped up to match that expected time.

5.3 Visual Feedback

One of the most valuable things about our pitch accent tutoring software is that it gives users a way to get feedback about their pronunciation. For example, a user might have a bad habit of putting an accent on the wrong part of the word, or omitting the accent for a part of the word that should be accented. If this user says this word using our software, they should become aware of their error, and the software should be able to guide them towards an improved pronunciation. In fact, our software tries to go one step further than that by giving users feedback in real-time, which gives the user a low-latency feedback loop about their performance.

5.3.1 Chosen Solution

To implement real-time visual feedback about the user's pitch accent, we added to the interface an overlay that allows the user to compare their voice's pitch-accent with the word's dictionary-form pitch-accent pattern. As the user speaks, a red circle travels across the current word from left to right. The height of the red circle conveys the relative pitch of the user's voice, and the radius of the circle conveys the volume of the user's voice. Furthermore, as the red circle travels from left to right, it leaves a red trail. Once the user finished pronouncing the word, the red trail remains, which allows the user to inspect the final results of their performance. See figure 5.3.1 for an example of what this user interface looks like.

5.3.2 Main Challenges

The implementation of this visualization encountered several main challenges. For example, plotting the user's speech over the dictionary form requires some calibration of the user's speech. The pitch range of the user's voice must be analyzed to define what constitutes a high pitch and what constitutes a low pitch, and this analysis serves as a calibration in the Y axis. Furthermore, the plot must be calibrated in the X direction based on when the user started speaking, to try to line up the user's speech with the beginning of the word. Finally, because the raw pitch data is quite noisy, some beautification of the data visualization was necessary.

To calibrate the speech visualization in the Y direction, we compute the minimum and maximum pitch seen so far in the current recording. Using this approach runs the risk of interpreting sharp consonant sounds as extreme pitch changes. For example, if the user says "**pa**", the popping noise of the letter "p" might mis-calibrate the audio. To solve this problem, we simply ignore pitch data that goes beyond the reasonable ranges of human speech, thus keeping only pitch data between 40Hz and 600Hz [11].

To calibrate the speech visualization in the X direction, we measure the average volume of the samples used to compute the pitch, and assume that the beginning of the word comes when we see the start of non-silent pitch data. This is one of the reasons why the red circle begins slightly before the beginning of the word, since it gives the user the chance to start speaking slightly more at their convenience.

To beautify the visuals, we smoothen the curve that represents the user's speech using a Catmull-Rom spline. The Catmull-Rom spline creates a smooth bend from one point on the curve to the next by defining a cubic Hermite spline using the 2 previous points and the 2 next points. This smooth interpolation method is used for both the height of the line (controlled by the pitch) and the thickness of the line (controlled by the volume.)

5.3.3 Future Work

One of the main problems with the current approach is that the spline being plotted in real-time tends to move a little erratically as the user speaks. This happens because the spline is being calibrated in real-time based on what the user says so far, so the spline can suddenly change shape many times as it adapts itself to the user. As-is, this

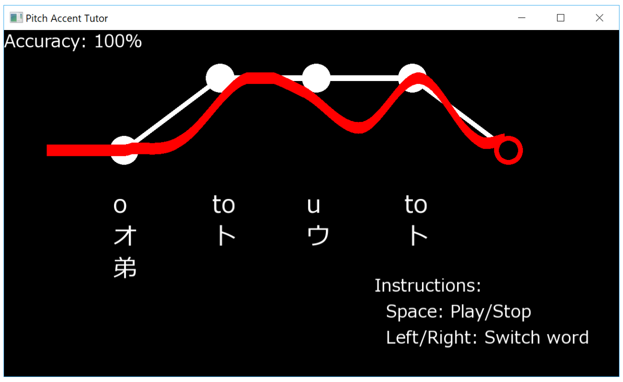

Figure 8. Example word with several consecutive high-pitched morae.

procedure happens every time the user says a word. This problem could be improved by retaining some calibration information from one word to the next, by assuming that the user's voice pitch should be relatively similar. Alternatively, the user could perform an initial calibration step at the program's initialization time.

5.4 Scoring System

Part of the appeal of our pitch-accent training software is the *game-ification* aspect, which turns what would be a tedious academic exercise into an engaging and enjoyable activity. The main way in which we game-ify the exercises is by challenging the user to match their voice to the dictionary-form of the pitch-accent pattern, since overcoming this challenge posed by an external source can be a source of personal satisfaction. Going one step further, we aimed to give the user a tangible score as well, which gives them a numerical assessment of their performance.

5.4.1 Chosen Solution

Two main methods to score the user's performance were considered:

1. Correlation of the red line and white line.
2. Pattern-detection in the pitch history.

Using the first method, one would measure the distance of the red line (see figure 5.3.1) from the white line drawn by the dictionary-form of the pitch-accent pattern. The better the red line matches, the higher the score.

Using the second method, one would define a series of expected features in the user's spoken pitch, and check how well the user's speech matches those patterns.

Although the first solution seems like an obvious one, and would probably give very accurate results in cases like figure 5.3.1, it has some problems in more general scenarios. For example, consider the case of the word ”otouto”, which has the pitch-accent pattern ”o(L)-to(H)-u(H)-to(H)”. In this word, there are several consecutive high-pitched morae, as shown in figure 5.4.1. Although the dictionary-form draws a straight high-pitched line between them, this is not a realistic representation of how a human would say the word.

To further elaborate, the pitch pattern for the word ”otouto” tends to have two peaks when said by a human, one for each ”to” of the word, with a brief dip in the middle. (Japanese note: When the letter ”u” follows ”o” in Japanese, it extends the sound of the ”o”, rather than making two distinct vowel sounds. Therefore, the word ”otouto” is pronounced more like ”o to-o to”.)

Considering these challenges, we decided to use the second method for scoring, using pattern detection, with the hope that the pattern detection would allow the user to have such upward bumps without having their score penalized.

5.4.2 Main Challenges

To implement the pattern matching, we begin by doing a pass of computation over the user's speech and extract features from it. From there, we try to find matches between the features and the dictionary-form pitch-accent pattern. This raises two questions: What do we consider a feature, and how do we match them to the dictionary-form?

To characterize the user's speech, we defined the following features:

- Peaks: Local maxima of the pitch.
- Troughs: Local minima of the pitch.
- Increases: Pitch-increase inflection points.
- Decreases: Pitch-decrease inflection points.
- Plateaus: Roughly flat region of high-pitch.
- Valleys: Roughly flat region of low-pitch.

After extracting the list of features, we traverse the dictionary-form pitch-accent pattern, and at every step we check if the current feature of the speech is compatible with the shape of the pitch-accent pattern at the current location. If the pattern matches, we award one point to the user. If the pattern does not match, we skip the current feature, as a form of naive error-recovery, perhaps similarly to how a programming language parser might skip a misplaced semicolon.

We match the pitch-accent pattern by comparing the pitch accent pattern of each mora and its immediate neighbors with the features. To do this, we experimentally defined a ”match matrix”, which defines which pitch accent patterns can be matched to which features (see table 2.)

As an additional step in the score calculation, we also require the user's speech to roughly match the dictionary-form pitch-accent pattern temporally in order to be awarded a point. This is an important addition, since a lack of temporal association would mean that user speech with a single high peak would generally match any word with a single peak, no matter where the peak really is.

When we finish the traversal of the dictionary-form pitch-accent pattern, or when we run out of user input, we

pitch	peak	trou.	inc.	dec.	plat.	vall.
LLL		●	●	●		●
LLH		●	●			●
LHL	●		●			
LHH	●		●			
HLL		●		●		
HLH		●		●		
HHL	●			●	●	
HHH		●	●	●	●	

Table 2. Matrix used to pattern-match dictionary-form pitch-accents to speech features. Dots indicate compatible speech features.

use the number of correct matches and the total length of the word to compute a percentage of how well the user pronounced the word, and we display this result in the user interface.

5.4.3 Future Work

The scoring system seems to be able to handle the tricky bi-modal cases as in figure 5.4.1. It also appears to work correctly in many other cases, working as predicted for both correct pronunciations, and for deliberately incorrect pronunciations. However, there are still regularly cases where surprising results happen, like being awarded 0% correctness for a word where the red line seems to match perfectly. We suspect that this happens due to noise in the data that causes the feature extraction to mis-identify or mis-match features.

The precise implementation of this system feels somewhat like an art form, since it seems to require a lot of tweaking, and perhaps requires a mix of different methods. For example, maybe we should automatically give a good score to user speech where the red line matches the white line well, regardless of what the pattern-matching system says. This would prevent users from being confused at how a seemingly perfect match of the red line caused them to receive a low score. Furthermore, the continuous nature of simply matching the two lines allows more nuance in the score, whereas the current system only gives effectively discrete ratios of how many morae were correctly matched.

5.5 Informal User Studies

To gather some external feedback, the software was distributed for testing to some members of a small online Japanese study chatroom. These users tested their pronunciation for various words, and expressed their feedback at the results.

Among the immediate feedback from these users was that it feels unnatural to try to match the red line perfectly to the white line, due to the tendency for there to be a small peak at each mora, as was shown in figure 5.4.1. This feedback was precisely that prompted our investigation of the pattern-matching approach of scoring, in an attempt to solve this problem.

Users also noted that one tends to need to exaggerate the differences in pitch in order to get a good result. Indeed, our program more-or-less expects you to speak like a robot, and misses many subtleties of human speech. This may be possible to fix with more fine-tuning of the calibration of various systems in our program, or perhaps with the use of higher quality audio equipment.

Although the users offered their criticism, it seemed like everybody had a bit of fun with the novelty of the system. Users tried out many words out of their own volition, and enjoyed proudly sharing screenshots of their best results with the other users in the group. The long-term effects of the system remain unmeasured, but we hope that the repetition and the visual feedback has helped ingrain the correct pitch-accent patterns in the users' memories. Measuring the effectiveness with respect to long-term memory remains an avenue for future work.

6. CONCLUSION AND FUTURE WORK

In summary, we attacked the research and development of the following features in this project:

- Real-time pitch-accent detection.
- Pitch-accent correction and replay.
- User interface design for pitch-accent training.
- Scoring of pitch-accent accuracy.

We hosted a short informal user study, where users seemed to have fun with the application, and inspired some of the improvements we made in the quality of the pitch-detection and analysis.

The final result of our project consists of a rough implementation of many of the systems that would be required in a commercially-viable application. In particular, our systems would require much additional improvement to achieve a standard of responsiveness and reliability that serious users might expect. Such improvements would be the main concern of future work.

The source code for our project is stored in a private GitHub repository, due to the dubious legality of distributing a CSV file dump of the NHK phonetics dictionary. The source code, along with a Windows build, have been attached to the project submission.

7. REFERENCES

[1] ojad - オンライン日本語アクセント辞書. `http://www.gavo.t.u-tokyo.ac.jp/ojad/`. [Online; accessed 16-October-2017].

[2] pitch accent in spoken-word recognition in japanese. `http://pubman.mpdl.mpg.de/pubman/item/escidoc:68866/component/escidoc:68867/Cutler_1999_Pitch+accent.pdf`. [Online; accessed 16-October-2017].

[3] 日本語教育用アクセント辞典. `http://accent.u-biq.org/`. [Online; accessed 16-October-2017].

[4] The development of lexical pitch accent systems: An autosegmental analysis. https://muse.jhu.edu/article/174223. [Online; accessed 16-October-2017].

[5] *Accent of Extended Word Structures in Tokyo Standard Japanese, The.* エデュカ, 1983.

[6] Alexander Arguelles. Guide to autodidactic foreign language study. http://www.foreignlanguageexpertise.com/foreign_language_study.html, 2010. [Online; accessed 16-October-2017].

[7] ”Dogen”. Japanese phonetics (youtube series).

[8] Hiroya Fujisaki and Keikichi Hirose. Analysis of voice fundamental frequency contours for declarative sentences of japanese. *Journal of the Acoustical Society of Japan (E)*, 5(4):233–242, 1984.

[9] Keikichi Hirose. Accent type recognition of japanese using perceived mora pitch values and its use for pronunciation training system. 10 2017.

[10] Keikichi Hirose, Frédéric Gendrin, and Nobuaki Minematsu. A pronunciation training system for japanese lexical accents with corrective feedback in learner's voice., 01 2003.

[11] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[12] ”javdejong”. Japanese pronunciation / pitch accent. https://github.com/javdejong/nhk-pronunciation. [Online; accessed 16-October-2017].

[13] Goh Kawai and Carlos Toshinori Ishi. A system for learning the pronunciation of japanese pitch accent., 01 1999.

[14] Laurence Labrune. The phonology of japanese, 2012.

[15] ”Kevin Russell”. vowels phonetics analysis.

[16] Greg Short, Keikichi Hirose, and Nobuaki Minematsu. Rule-based method for pitch level classification for a japanese pitch accent call system. 10 2017.

[17] Y. Sugiyama. *The Production and Perception of Japanese Pitch Accent*. Cambridge Scholars, 2012.