

Extended Abstract: An Update to the Urarina Language Verification Project

(A Work in Progress)

Michael Dorin

University of St. Thomas
2115 Summit Ave, OSS 301
St. Paul Minnesota 55105 USA
mike.dorin@stthomas.edu

Judith Dorin

judy@chaski.com

Abstract

Documenting endangered languages is a long-standing, persistent challenge. Many documentation procedures are repetitive, time-consuming, and error-prone. Software tools can substantially reduce this burden. However, with the wide variety of spoken languages in the world, the work involved in finding efficient tools that provide satisfactory results for a specific project and its recordings also takes time and effort. This research presents a pathway to tailor existing tools to projects and automate the language documentation process. This study used directed recordings based on Mel-Frequency Cepstral Coefficients (MFCCs) using commonly available Python libraries. The output was used to reliably identify phonemes in recorded words. During testing, the software achieved better than 86 percent accuracy in phoneme identification for the examined Urarina recorded words. The positive outcomes of this study can contribute to the linguistics field by showing researchers a practical means for creating software tools individualized to a particular language under study. Although this is a work in progress, the preliminary results are favorable and warrant further exploration.

1 Introduction

Capturing and preserving endangered languages presents a formidable challenge and often demands many hours of painstaking work, listening to and documenting recordings. This task can be incredibly daunting for research endeavors operating with limited resources. As more and more languages worldwide are on the verge of extinction, it is imperative to create practical and efficient software to aid the preservation of these endangered languages.

The motivation for this research came from a study performed in the mid-1980s by students from the University of San Marcos in Lima, Peru. Their study focused on the documentation of the Urarina language. This language is spoken by the Urarina

people, an indigenous group residing in the Loreto Region of the Peruvian Amazon. Two of the researchers, Judith Cajas and Beatriz Gualdieri, assembled a collection of audio recordings and handwritten notes (Cajas and Gualdieri, 1987). These audio recordings were digitized and are foundational data for this study.

Automation of phoneme extraction through software research has been an ongoing investigation topic akin to the search for “El Dorado,” and several papers have been written on this topic. For example, Geetha and Vadivel describe phoneme segmentation of the Tamil language. They present the use of spectral transition on 30 unique Tamil words constituting 172 phonemes and have achieved good results (Geetha and Vadivel, 2017). Additionally, Baldwin and Tanaka describe an automated method of grapheme-phoneme alignment for the Japanese language, achieving 94.7 percent accuracy (Baldwin and Tanaka, 1999). Mahmit, Nicola, and Stoicu-Tivadar use cross-correlation to automate the segmentation of audio samples. Their paper also demonstrates excellent results, showing that 83.3 percent of their studied cases achieved the maximum correlation value (Husni et al., 2013). Wang et al. also do an excellent job of describing self-supervised semantic-driven phoneme discovery. Their paper does a good job explaining the path they took, and provides proof demonstrating the effectiveness of their algorithms. Although these papers demonstrate success and provide insight into their math and implemented algorithms, exploiting their efforts is only practical for those with adequate math backgrounds.

Professor Herman Kamper has produced additional important work. Kamper aims to “develop methods that allow machines to acquire speech and language processing capabilities with as little supervision as possible” (Kamper, 2023). Kamper has authored various libraries (Kamper, 2021a), but his work in Dynamic Time Warping (DTW) is

the most interesting for this study. DTW is a time series alignment method helpful for many tasks, including speech recognition. Using DTW, it is possible to measure the similarity between sound waves that may vary in time or speed (Müller, 2007). Although Kamper provides abundant and detailed insight into the use of his work (Kamper, 2021b), many individuals specializing in historical linguistics have limited mathematical and programming backgrounds. As such, using these tools can be difficult or impossible for many historical linguists.

This work creates practical tools by applying the available linguistics libraries to phoneme extraction. First, raw data is obtained effectively through crowd-sourcing. The Chaski Phoneme Project enables volunteers worldwide to contribute by recording their voices pronouncing phonemes from the International Phonetic Alphabet (IPA) (Dorin, 2023). The second part of this project is the creation of a small, easy-to-understand Python program designed to segment the words from the Urarina study done by the San Marcos students. Employing DTW, the software searches for commonalities among the recorded words. Once commonalities are found, they are then compared to the phonemes gathered by the Chaski Phoneme Project. A report is then produced listing the word and the likely phonemes.

As a result of this study, practical Python code for documenting phonemes will be made publicly available. Additionally, linguists will have access to a new collection of IPA phoneme recordings. These resources will be highly beneficial for historical linguists committed to language preservation.

2 Methods

2.1 Python and Libraries

Although this research simplifies the use of existing libraries, basic programming and Python experience is still necessary. For this project, the PyCharm IDE from JetBrains was employed (JetBrains, 2023b) on a laptop running Microsoft Windows 11. Pycharm was installed in a standard manner, but one additional noteworthy step was required for building with the existing linguistics libraries. Cython support for Pycharm was required and information on Cython support may be found on the JetBrains website (JetBrains, 2023a).

Most of the libraries used for this project were installed using Python pip. These libraries are shown in Table 1. Notice that Cython is mentioned on the pip install list, but additional action was still

Necessary Libraries
cython
numpy
scipy
python_speech_features
matplotlib

Table 1: Install these libraries with pip.

required as described in the Cython support page from JetBrains (JetBrains, 2023a).

2.2 Data Preparation

As mentioned in the introduction, fundamental data for this study came from Cajas and Gualdieri’s work (Cajas and Gualdieri, 1987), in which they collected a treasure trove of recordings on cassette tapes of individual words from the Urarina language. These recordings were digitized, noise-reduced, and segmented such that each individual word was in its own .wav file. These tasks were performed using the Sox audio utility (Chris Bagwell, et al., 2021) as well as the FFmpeg utility (Fabrice Bellard et al., 2023). A sample batch file for performing this processing is shown in Listing 1.

```

1 ffmpeg -i $1 -ac 1 mono.$1
2 echo mono.$1 part 1
3 sox mono.$1 filtered.$1 sinc -n
   ↳ 32767 100-5000
4 echo $1 part 2
5 sox filtered.$1 amplified.$1 vol
   ↳ 7dB
6 echo $1 part 3
7 sox amplified.$1 padded.$1 pad
   ↳ .202
8 echo $1 part 4
9 rm filtered.$1
10 rm amplified.$1
11 rm mono.$1

```

Listing 1: Audio Process Batch File

Also, as previously mentioned, recordings of phonemes came from the Chaski Phoneme Project (Dorin, 2023). At the time of writing, the Chaski Phoneme Project had more than 1,100 recordings from volunteers living in five different countries. Although many recordings could be of better quality, no culling has been performed at this time. It was, however, necessary to harmonize the phoneme recordings’ sampling rate with the Urarina record-

ings' sampling rate. Some of the Chaski Phoneme recordings also had to be repaired. Commands for this process are shown in Listing 2

```
1 sox sample.wav -r 44100 sample.  
   ↪ out.wav  
2 sox --ignore-length corrupted.wav  
   ↪ fixed.wav
```

Listing 2: Commands to Fix Sampling Rate and Broken Files

2.3 Processing

Processing the Urarina words was done in several steps. Each step is described below.

2.3.1 MFCCs Creation

In this step, each word was read, and appropriate MFCCs data was gathered. This was done exactly as described by Kamper in his sample code, with the exception that the number of Fast Fourier Transforms (FFTs) was set to 1,200, and Window Length was set to 12.5 milliseconds (Kamper, 2021a).

2.3.2 Construction of Urarina Word Blocks

In the second step, the words were divided up into equal-sized parts. The selection of the part quantity was based on considerations that came from the writing of Ivan Obolensky. Obolensky described normal speech as being spoken at 10 to 15 phonemes per second and, at the extreme, the upper limit of intelligibility begins at about phonemes 40 per second (Obolensky, 2011). The second consideration came from the mathematical concept of the Nyquist rate. The Nyquist rate states that to capture a signal accurately, the sampling rate must be at least twice the signal's highest frequency for a signal. An introduction to signaling and sampling authored by Candès and Wakin provides excellent information on this concept (Candès and Wakin, 2008).

Considering the window size for MFCCs calculation used by the Kamper Library is 12.5 milliseconds, with a 10-millisecond step, and with acknowledgment of Obolensky and Nyquist, 25 milliseconds was chosen as a reasonable size with the thought that a fast speaker has a tempo of 40 phonemes per second. The composition of a typical MFCCs block is illustrated in Figure 1.

2.3.3 Sorting and Searching

Next, using the DTW library, all the word blocks were compared to blocks from other Urarina words.

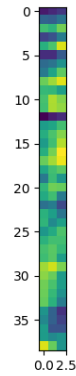


Figure 1: Example MFCC Composition of a Single Block

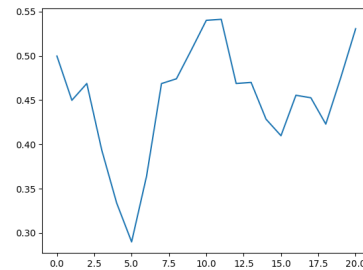


Figure 2: Sample of Successful Block Discovery

Figure 2 demonstrates one of the results of the DTW algorithm. The spike in the image represents where "target audio" is contained within the "search audio." This whole process is computationally expensive (Papadimitriou, 2003), and initially, only a small number of the recorded Urarina words were used. Discovered common audio blocks were stored with information on the word where they originated. By sorting blocks by commonality, it is possible to report a breakdown of phonemes used in the selected words. Commonality tests were also reversed. That is to say, if a location block seven in a particular word was found to be a match at location block eight of a different word, employing the rules of total order semantics (Baeten, 1993), it follows that the block at location eight of the second word must also agree with the block at location seven in the first word. Tests were run to confirm this. Figure 3 shows three of the Urarina words used for this testing. Of the eleven Urarina words chosen for this initial testing, four of the words were selected to be identical words spoken at different times by the same speaker.

2.3.4 Phoneme Discovery

Ultimately, the database of common blocks discovered in each word was compared against the phoneme recordings from the Chaski Phoneme Project (Dorin, 2023). The Chaski phonemes were also broken up into 25 millisecond blocks. As this is also computationally expensive, this current work focused solely on vowels.

3 Results and Discussion

3.1 Commonality Among Words

3.1.1 Identical Words

The blocks separated from identical words matched perfectly. This test was essential to verify the application of total order semantics. (If A is the same as B, then it must be true that B is the same as A.)

3.1.2 Common Blocks in Different Words

Regarding the blocks separated from different words, in general, the software correctly aligned the blocks used in the study test set. The blocks that were not perfectly aligned were found to be neighbors of the correct block. The Urarina words used for verification can be found in Figure 3.

3.2 Phoneme Discovery

Although the scanning was limited to vowels for this work in progress, the results have been very satisfying. The custom Python software correctly identified vowel phonemes from the recordings of Urarina words at a success of 86 percent. Most of the failures were due to the software's confusion over vowel specialization, such as nasalization. This class of issues can be resolved by expanding and improving the Chaski Phoneme Project's database.

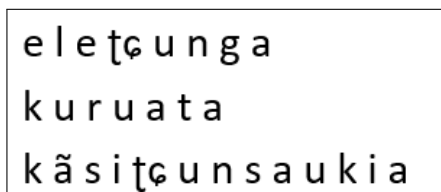


Figure 3: Phonemes of three Urarina words.

4 Conclusion

This work demonstrates it is practical to create tools for phoneme documentation using commonly available linguistics libraries. Raw data was obtained

through the Chaski Phoneme Project, which effectively used crowd-sourcing to create a database of phoneme recordings. A small, easy-to-understand Python program was produced to segment words. The program next used DTW to search for commonalities among the Urarina words. Once commonalities were found, they were compared to the Chaski phoneme database using DTW. The program then produced a report for each word and its associated phonemes. As a result of this research, practical Python code for documenting phonemes is now available, and linguists now have access to a new collection of IPA phoneme recordings. Though this project still needs refinement, once complete, these resources will be highly beneficial for historical linguists committed to language preservation.

Acknowledgements

I wish to acknowledge Judith Cajas and Beatriz Gualdieri for their pioneering work in language preservation. I wish to thank Anders Koskinen for his work reviewing this text.

References

- J. C. M. Baeten. 1993. The total order assumption. In *NAPAW 92*, pages 231–240, London. Springer London.
- Timothy Baldwin and Hozumi Tanaka. 1999. Automated japanese grapheme-phoneme alignment. In *Proc. of the International Conference on Cognitive Science*, pages 349–54. Citeseer.
- Judith Cajas and Beatriz Gualdieri. 1987. Kača êje (lengua urarina): Aspectos de la fonología. Master's thesis, Tesis de licenciatura, Universidad Nacional Mayor de San Marcos.
- Emmanuel J Candès and Michael B Wakin. 2008. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2):21–30.
- Chris Bagwell, et al. 2021. Sox - sound exchange. <https://sourceforge.net/projects/sox/>. Accessed: September 8, 2023.
- Michael Dorin. 2023. The Chaski Phoneme Project. <https://www.chaski-linguistics.org/>. Accessed: September 8, 2023.
- Fabrice Bellard et al. 2023. [Ffmpeg official website](#). Accessed: September 8, 2023.
- K Geetha and R Vadivel. 2017. Phoneme segmentation of tamil speech signals using spectral transition measure. *Oriental Journal of Computer Science and Technology*, 10(1):114–119.

- Husniza Husni, Yuhanis Yusof, and Siti Sakira Kamaruddin. 2013. Evaluation of automated phonetic labeling and segmentation for dyslexic children's speech. *Proceedings of the World Congress on Engineering 2013 Vol II*.
- JetBrains. 2023a. Pycharm cython support - jetbrains documentation. <https://www.jetbrains.com/help/pycharm/cython.html>. Accessed: September 8, 2023.
- JetBrains. 2023b. Python IDE for Professional Developers. <https://www.jetbrains.com/pycharm/>. Accessed: September 8, 2023.
- Herman Kamper. 2021a. Speech dtw: A python toolkit for fast and flexible dynamic time warping. https://github.com/kamperh/speech_dtw. Accessed: September 8, 2023, Additional Path: `speech/_dtw/_qbe.ipynb`.
- Herman Kamper. 2021b. Youtube channel: Herman kamper's machine learning tutorials. <https://www.youtube.com/@HermanKamperML>. Accessed: September 8, 2023.
- Herman Kamper. 2023. Home Page of Herman Kamper. <https://www.kamperh.com/>. Accessed: September 8, 2023.
- Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84.
- Ivan Obolensky. 2011. The rhythms of phonemes. <https://dynamicdoingness.com/the-rhythms-of-phonemes/>. Accessed: September 8, 2023.
- Christos H Papadimitriou. 2003. Computational complexity. In *Encyclopedia of computer science*, pages 260–265. Association for Computing Machinery (ACM).