LANGUAGE DOCUMENTATION & CONSERVATION

Vol. 12 (2018), pp. 80–123
http://nflrc.hawaii.edu/ldc
http://hdl.handle.net/10125/24763
Revised Version Received: 17 Jan 2018

Forced Alignment for Understudied Language Varieties: Testing Prosodylab-Aligner with Tongan Data

Lisa M. Johnson
University of Utah
Weber State University

Marianna Di Paolo
University of Utah
National Museum of Natural History (Smithsonian)

Adrian Bell
University of Utah

Automated alignment of transcriptions to audio files expedites the process of preparing data for acoustic analysis. Unfortunately, the benefits of auto-alignment have generally been available only to researchers studying majority languages, for which large corpora exist and for which acoustic models have been created by large-scale research projects. Prosodylab-Aligner (PL-A), from McGill University, facilitates automated alignment and segmentation for understudied languages. It allows researchers to train acoustic models using the same audio files for which alignments will be created. Those models can then be used to create time-aligned Praat TextGrids with word and phone boundaries marked.

For the benefit of others who wish to use PL-A for research projects, this paper reports on our use of PL-A on Tongan field recordings, reviewing the software, outlining required steps, and providing tips. Since field recordings often contain more background noise than the laboratory recordings for which PL-A was designed, the paper also discusses the relative benefits of removing background noise for both training and alignment purposes. Finally, it compares acoustic measures based on various alignments and compares boundary placements with those of human aligners, demonstrating that automated alignment is both feasible and less time-consuming than manual alignment.

1. Introduction Linguists engaged in sociolinguistics and language documentation face similar problems when it comes to efficiently processing large corpora of recorded speech. Though field recordings can be collected efficiently, it may take months or years to process the audio for certain types of analysis. In addition to the effort involved in transcription, phonetic analysis often requires the time-consuming alignment of transcription to audio. The expense related to this process may limit both the questions researchers can explore and the amount of data they can analyze. However, recent advances in speech recognition technology have led to the development

E-ISSN 1934-5275

of tools to automate time alignment of transcription to audio, particularly for English data (See, for example, Evanini et al. 2009; Goldman 2011; Kisler et al. 2012; Rosenfelder 2013; Reddy & Stanford 2015).[1]

Can researchers investigating understudied languages benefit from these technological advances? Cross-language forced alignment – aligning recordings and transcriptions of an understudied language using a model trained on a majority language – has been attempted with some success (Kempton et al. 2011; Kurtic et al. 2012; DiCanio et al. 2013; Strunk et al. 2014; Kempton 2017). However, difficulties in matching phonemic inventories, the need to convert orthographic transcriptions to another system, and the expected errors due to language mismatch make this approach less than ideal.

Fortunately, new tools make it possible to train acoustic models specifically for any language using as little as one hour of transcribed recordings. This paper reviews the use of one such tool, Prosodylab-Aligner (PL-A) (Gorman et al. 2011), developed at McGill University's Prosody Lab. Our tests show that PL-A produced good alignments from a relatively small corpus of Tongan word list recordings. Though it lacks a user-friendly interface and some of the features offered by more sophisticated aligners, its success demonstrates the feasibility and utility of trained forced alignment with understudied languages, and suggests that researchers need not resort to majority-language models. Despite the time required to prepare files and make corrections to TextGrid outputs, we have found that forced alignment can increase efficiency, consistency, and replicability.

The paper is divided into five main sections. In §2, we provide background on the Tongan Ethnolinguistic Project and on forced alignment. In §3, we review PL-A's major components and outline each step in the training and alignment process. In order to help others who are considering the use of the tool in their own research, we include troubleshooting notes and tips. In §4 we compare the results of multiple alignment tests with different types of files and different parameter settings. A key question explored in this section is how well PL-A performs on long field recordings with background noise. In §5 we compare PL-A's alignment to a manually corrected sample, and in §6 we summarize our conclusions and recommendations for potential users.

2. Background

2.1 Tongan Ethnolinguistic Project The Tongan Ethnolinguistic Project is a large-scale study of post-migration Tongans/Tongan Americans in the United States that will include longitudinal and cross-sectional data on the formation of new post-migration ethnolinguistic identities. In order to identify potentially important linguistic

[1]Special thanks to Kyle Gorman and Craig Johnson for help in technical troubleshooting and compiling, to Keelan Evanini for HTK explanations, and to Carter Holt for his work on file cleanup. Project supported by: Bell, Adrian V. & Di Paolo, Marianna. 2014. Language, ethnic markers, and the adaptation of Tongan immigrants to Utah. University Research Council Grant, University of Utah. Also supported by: Holt, Carter. 2015. Boosting phonetics research through technology. Undergraduate Research Opportunity Program. Summer 2015 Research Assistantship. (Marianna Di Paolo, mentor).

variables, we are collecting recordings from Tongans in the United States and in the Kingdom of Tonga. In 2014, team members visited several villages across the Tongan islands of Tongatapu, Ha‘apai, Vava‘u, and ‘Eua, recording interviews, word lists, and short reading passages from a total of 40 speakers. The long-term project plan includes collecting both English and additional Tongan speech samples through interviews and crowd-sourcing. Recognizing the time required to transcribe and align existing and future recordings for sociophonetic analysis, we began searching for tools that might expedite the process. After attending a presentation at the 2015 Linguistic Society of America Annual Meeting (Bale et al. 2015) and speaking with the creator of PL-A, we determined to test the forced alignment tool on our word list data. Although we will ultimately be aligning and analyzing connected speech as well, these preliminary tests with word list readings were conducted to determine whether forced alignment would be a feasible and efficient option with the kind of recordings contained in our corpus.[2]

2.2 Forced alignment In recent years, several tools have been developed to time-align an orthographic transcription to recorded speech at the word and phone level, sometimes combining this with automated formant extraction. Those tools include: the Penn Phonetics Lab forced aligner (Yuan & Liberman 2008), now wrapped into the Forced Alignment and Vowel Extraction (FAVE) (Rosenfelder et al. 2011; Rosenfelder 2013); EasyAlign (Goldman 2011); Prosodylab-Aligner (Gorman et al. 2011); MAUS/WebMAUS (Kisler et al. 2012); and the Dartmouth Linguistic Automation suite (DARLA)[3] (Reddy & Stanford 2015). Though the specific methods vary somewhat, these so-called “forced alignment” tools rely on the Hidden Markov Model Toolkit (HTK) (Cambridge University 1989-2015) speech recognition technology to create a statistical model associating phonetic symbols to speech signals. Some newer tools – Gentle (Ochshorn & Hawkins 2016) and Montreal Forced Aligner (McAuliffe et al. 2016; McAuliffe et al. 2017a; McAuliffe et al. 2017b) – employ the Kaldi ASR Toolkit (Povey et al. 2011) rather than HTK to create their statistical models.

Forced alignment can be classified as *unconstrained* or *constrained* based on the kind of transcription used. In unconstrained alignment, the aligner is given no information about where to look for given words in a recording and must try to align the words and phones based only on linear order starting at the beginning of the audio file (Strunk et al. 2014). This type of alignment can be contrasted with constrained alignment, in which the transcription is time-aligned to sections of the recording, such as utterances or breath groups. Pre-aligning the data in this way provides the aligner with additional information about where to look for particular words and phones, generally leading to better results. A particular alignment tool may use unconstrained alignment, constrained alignment, or both. For example, the FAVE web interface uses

[2]Forced alignment of connected speech is addressed in the first author’s review of Montreal Forced Aligner (in preparation).

[3]Technically, DARLA is not a new forced alignment tool. It uses Prosodylab-Aligner behind the scenes to create aligned TextGrids. DARLA’s innovation is to combine speech recognition technology, forced alignment, and automated vowel formant extraction in a single, user-friendly, web-based environment.

constrained alignment, while DARLA and MAUS offer both constrained and unconstrained options.

Forced alignment tools promise great savings in time, and therefore cost. Manually aligning a transcription at the phone level can take up to 800 times the duration of the audio – or as much as 13 hours for a one-minute recording (Goldman 2011; Schiel et al. 2012). Automating that process has made formerly impossible studies of large corpora a reality.[4] That the FAVE system is becoming a standard tool in English variationist sociolinguistics speaks to the utility and efficiency of such advances.

Another benefit of forced alignment is improved consistency and replicability. Manual alignments are not perfect, and trained professionals do not always agree on where boundaries should be placed. Strunk et al. (2014) report differences in word boundary placement between human annotators for several languages. Across eight samples from five languages, the mean interaligner difference was 85.5 ms, and the median was 35.5 ms. In a test using English recordings, Goldman (2011) found human vs. human agreement on phone boundary placement to be 79% at a 20 ms threshold and 62% at 10 ms. The results were similar when French recordings were used, showing 81% agreement at 20 ms and 57% at 10 ms. The same study found agreement between a machine alignment and each human alignment to be comparable to the inter-human agreement. Citing previous reviews, DiCanio et al. (2013) make a more general statement about agreement rates between forced alignment and hand alignments, stating that "forced alignment on models trained on their target language ...typically average above 80% within 20 ms" (2240).

Though automated alignment is not expected to be error free, some have suggested that the consistency provided by forced alignment may make even the errors predictable, which may be an advantage over human error. (See, for example, Sikveland et al. 2010; Kurtic et al. 2012.) Reddy & Stanford (2015) suggest that this allows for replicability and the reporting of error rates along with results in the same way that they are reported in statistical modeling. In addition, we contend that where manual adjustments must be made to boundary placement, researchers can focus on fixing specific problems based on well-articulated criteria. This may improve consistency compared to when a human must segment and align an entire recording, considering a multitude of factors at once.

Unfortunately, most of the benefits of forced alignment have been restricted to research on English and other resource-rich languages.[5] Major aligners have been trained using copious data, sometimes hundreds of hours of transcribed speech. Many researchers have assumed that a full alignment system could not be created for languages with considerably less available data (Kempton et al. 2011; Kurtic et al. 2012; DiCanio et al. 2013; Strunk et al. 2014; Kempton 2017). In order to work around the small data problem, those researchers have resorted to cross-language forced align-

[4]Achieving the correct balance between accuracy and efficiency is a complicated task and depends on the type of analysis to be undertaken. For more detailed discussions about accuracy and efficiency, see Schiel et al. (2012), Goldman (2011), and Reddy & Stanford (2015).

[5]EasyAlign is available for English, French, Spanish, and Taiwan Min. A complete list of languages supported by MAUS can be found at https://clarin.phonetik.uni-muenchen.de/BASWebServices/#!/services/WebMAUSGeneral.

ment (CLFA), using models trained on unrelated mainstream languages to align data from understudied and endangered languages. Although this process is not expected to produce the same quality of results as a tool like FAVE, it can be used "to give a good initial alignment on small amounts of challenging data in the under-resourced language" (Kurtic et al. 2012: 1326). Researchers using CLFA report their various results as "encouraging" (Strunk et al. 2014; Kempton 2017) and "fairly good" (DiCanio et al. 2013). Though differences in data and study design make direct comparisons difficult, some results from CLFA studies are included in Table 1.

Table 1. Example study results for cross-language forced alignment (CLFA)

Study	Training Language	Aligned Language	Type of data	% Agreement <20 ms
DiCanio et al. (2013)	English (hmalign)	Yoloxóchitl Mixtec	isolated words	61%
DiCanio et al. (2013)	English (p2fa-Penn)	Yoloxóchitl Mixtec	isolated words	52%
Kempton (2017)	Czech	Nikyob (Nigeria)	isolated words	66%
Kurtic et al. (2012)	Russian	Bosnian Serbo-Croatian	free conversation	49%
Kurtic et al. (2012)	Hungarian	Bosnian Serbo-Croatian	free conversation	41%
Kurtic et al. (2012)	American English	Bosnian Serbo-Croatian	free conversation	43%
Kurtic et al. (2012)	Czech	Bosnian Serbo-Croatian	free conversation	47%

Strunk et al. (2014) report results differently, using differences in boundary placement measured in milliseconds. Table 2 shows results for several language samples using the MAUS "sampa" mode, which combines acoustic models from several languages supported by MAUS.[6] Note that this study only considered word boundary placement and did not analyze word-internal phone boundaries.

Table 2. Strunk et al. (2014) results for six test samples using MAUS sampa model for constrained forced alignment

Language and difficulty of sample	# of Words	Mean difference	Median difference	Max difference
Baure (easy)*	502	160 ms	101 ms	1,030 ms
Baure (hard)	689	204 ms	60 ms	3,214 ms
Bora (easy)	289	148 ms	30 ms	3,338 ms
Even (easy)	405	196 ms	34 ms	2,272 ms
Even (hard)	236	248 ms	63 ms	770 ms
Sri Lanka Malay (easy)	204	207 ms	41 ms	6,127 ms

*The authors define "easy" texts as good quality recordings of single speakers with few background noises; "hard" texts have multiple speakers, overlapping speech, and more background noise.

As noted by study authors, CLFA shows promise, though manual adjustments would generally be required. However, there are downsides to the CLFA approach. For example, the accuracy of CLFA depends partly on how closely the phone set,

[6]The table shows only the results for constrained alignment, which were generally much better than the results for unconstrained alignment.

syllable structure, and phonological rules of the target language match those of the language on which the acoustic model was trained (DiCanio et al. 2013). Finding the best fit between model and target language can be time consuming and may require testing of candidate models (Kempton et al. 2011; Kurtic et al. 2012; Kempton 2017). Once a model is selected, transcriptions may need to be converted to match the conventions of the model. This step requires extra effort or tools, and the resulting time-aligned transcriptions may look nothing like the orthography of the study language.

Prosodylab-Aligner (PL-A), reviewed in this paper, takes another approach, allowing users to train their own acoustic models for less-studied languages using a small amount of data. Given a set of sound files, simple orthographic transcriptions, and a pronunciation dictionary file, PL-A produces time-aligned TextGrid files for use in Praat (Boersma & Weenink 2015) or for import into other software, such as ELAN (Max Planck Institute for Psycholinguistics). To date, PL-A has been used with a variety of majority and minority languages, including English (U.S.A., Canadian, British, Aviation, South African), French, Arabic (Gulf), Irish, Cantonese, German, Polish, Mandarin, Tagalog, Spanish, Ch'ol, Mi'gmaq, and Kaqchikel (Gorman, p.c.). Though models trained with small amounts of data may not be as good as those trained on large corpora, the language-specific model still offers advantages over CLFA. Notably, a language-specific model can be refined as new data and information become available. During the initial stages of research, information on variation may be sparse, but further research may reveal alternate pronunciations that can be added to the dictionary, leading to improved acoustic models and alignments. (Note the examples of alternate pronunciations included in the English pronouncing dictionary in Figure 3, below, which is representative of the CMU dictionary (Carnegie Mellon University 1993–2016) used by both PL-A and FAVE.)

Prosodylab-Aligner is one of the first tools to make language-specific model training available to linguists who are not software programmers. Its specific requirements and components are described in the next section. For each step in the preparation, training, and alignment processes we provide general instructions as well as a review of how that process worked in our own tests.

3. Using Prosodylab-Aligner

3.1 Getting started Prosodylab-Aligner, developed at McGill by Kyle Gorman and Michael Wagner, is a set of Python scripts for performing forced alignment using HTK.[7] It is available for free download from GitHub (https://github.com/prosodylab-/Prosodylab-Aligner). Officially, it has been tested on Mac OS X and Linux, and instructions are provided for Mac installation (http://cslu.ohsu.edu/ gormanky/aligner-/aligner-tutorial-startup.html). Since our tests were conducted on a PC, we know

[7]The McGill Prosody Lab has recently produced a second tool for forced alignment on understudied languages. This software, Montreal Forced Aligner, is easier to use and offers some advances over PL-A. It is not discussed here but will be reviewed in a separate paper.

that a Windows environment is also an option. Regardless of platform, the installation and setup process is somewhat complicated, though doable. In general, users must install Python and related packages and must install and compile HTK[8] before installing the set of scripts that comprise the aligner itself.

Though we began by installing all of the components on a Mac, we eventually ran all of our tests on a Windows machine (Microsoft Surface Pro 3,[9] Intel Core i5-4300U processor, 8GB LPDDR3 RAM, 256GB SSD, and 200GB micro SD). Our choice to use a Windows platform was based primarily on the availability of resources. The PC at our disposal was newer and faster than our Mac, and we had access to a Windows programmer to help with troubleshooting.[10] We have written up our instructions to build on a Windows x64 PC, and they will be made available upon request.

Unlike FAVE and DARLA (which work only on English), PL-A doesn't have a user-friendly web interface or comprehensive documentation. At first, linguists without programming experience may find the installation and command line interface intimidating. For this reason, we will share tips that may help others learn from our experience and navigate the process more easily.

3.2 Data Preparation PL-A requires the following for the training and alignment processes:

- audio files
- transcription files
- a dictionary file
- a configuration file

3.2.1 Audio files PL-A requires audio files in .wav format. The default sample rate is 16 kHz (the standard minimum acceptable sample rate), and recordings at other sample rates are automatically resampled. To avoid this resampling process, which can be very slow,[11] users can either resample their files before running the training command (using the resample.sh script included in the package or using software like Audacity, available at http://www.audacityteam.org/download/) or can override the resampling process by changing the sample rate setting in the configuration file. (See §3.2.4.) The recommended minimum amount of training data is one hour of recorded

[8]The version of HTK downloaded and used for the Tongan tests described here is 3.4. Since that time, a newer version, 3.5 has been released.

[9]The last two models were trained on the Surface Pro 4, Core i5 6300u 2.4ghz, Intel 520 Graphics, 8GB RAM, 256 GB Samsung MZFLV256 NVMe SSD. The last two alignment tests were also performed on this machine.

[10]This doesn't mean that the tool would not have worked on our Mac. Though we encountered some errors early in the process, the PL-A developer was very helpful in resolving them. Since most people use PL-A on a Mac, it's likely to run most smoothly on that platform.

[11]Of the automatic resampling performed by PL-A, the README file states: "Resampling this way can take a long time, especially with large sets of data. It is therefore recommended that sample rate specifications are made using resample.sh. This requires installing SoX." (https://github.com/prosodylab/Prosodylab-Aligner/blob/master/README.md)

audio. PL-A was designed with laboratory recordings in mind. The instructions recommend using audio files and transcripts corresponding to single sentences or breath groups, though longer or shorter files may also be used. Using very short sound files is a way of constraining the alignment or limiting where the aligner can look for words and phones. However, because one-sentence audio files recorded in a lab are not the only kind of data field linguists deal with, we tested the tool on long field recordings of word lists to determine how it would perform under those conditions.

The audio data used for the present study consist of word list recordings extracted from longer interviews featuring other speech styles as well. They were recorded in Tonga during May of 2014 using lavalier microphones (Audio-Technica AT831b Miniature Cardioid Condenser Microphone) and a Zoom H4n digital recorder set at 16 bit and 44.1 kHz. (We did not resample to 16kHz.) Because we wanted to preserve the word list context for each speaker, we did not edit the audio files into small, single-utterance files. The recordings were made in a field setting rather than a laboratory, and despite efforts to minimize problems, most of the recordings contain the kind of background noise that is typical of field recordings. In this case, extraneous noises include sounds from animals, cars, wind, church bells, laughter, etc. In order to test the effects of such noise on the training and alignment processes, we performed tests on two sets of files hereafter referred to as *dirty* and *clean*.

- Dirty files: These files include all background noises in the recording. The only editing performed on these files used Audacity to eliminate extraneous conversation and isolate the word list reading from other parts of the interview.[12]
- Clean files: These files have been edited to remove tokens with background noise as well as extraneous noise between words.[13] Because these recordings will be used for acoustic analysis, we made no alterations to the audio signal. Thus, the *clean-up* refers only to the purging of ambient noises and of affected tokens while leaving the remaining tokens untouched. Cleaning 22 audio files took an undergraduate research assistant 120–150 hours.[14] Table 3 illustrates the extent to which the recordings were edited, listing both the original recording length and the edited recording length for the first five speakers' word list readings.[15]

[12]A few tokens were also removed due to wind interference that was severe enough to prevent speech from being heard and recognized.

[13]In addition to background noise, this process also eliminated long pauses. However, a minimum of 20 ms of "silence" was left on either side of each included word token, and all boundaries were placed at zero crossings.

[14]This estimate includes the time required to train the research assistant to use Praat and make the edits.

[15]A reviewer points out that we could have cleaned the files by inserting stretches of silence over background noise or problematic tokens rather than by deleting them. This would have preserved the original lengths of the files, as well as information about speech rate that may be useful for some kinds of analysis. That method would also have allowed us to make direct comparisons between TextGrids created using clean and dirty versions of the same recordings. We agree that is probably a superior approach, though we did not think of it when we began our project.

The resulting dataset includes 22 clean files (one per speaker), for a total recording time of 1:41:30, and 16 dirty files (one per speaker, different speakers from those in the first group), for a total recording time of 2:39:44. For comparison, we also have dirty versions of five of the recordings in the clean set (1:23:01). Both clean and dirty files contain only one voice, and only the single track recorded by that speaker's lavalier microphone.[16] Both the clean set and the dirty set include recordings from both men and women.

Table 3. Original and edited recording length for first five speakers

Speaker ID#	Original Recording Length	Edited Recording Length
001	16:22	5:59
002	13:48	5:31
003	15:42	6:22
004	16:22	4:59
005	20:45	4:50

3.2.2 Transcription files PL-A does not require (or accept) any time-aligned transcription input, placing it firmly into the unconstrained alignment category.[17] Transcription files must be in plain text format, using UTF-8 encoding, and saved with a .lab extension. The prescribed format requires single spaces between words and no carriage returns or punctuation. Regular spelling conventions can be followed using Unicode for special characters. Figure 1 provides an English example from the PL-A documentation.

Figure 1. Transcription example (English text, from PL-A README file)

```
BARACK OBAMA WAS TALKING ABOUT HOW THERE'S A MISUNDERSTANDING THAT ONE MINORITY GROUP CAN'T
GET ALONG WITH ANOTHER SUCH AS AFRICAN AMERICANS AND LATINOS AND HE'S SAID THAT HE HIMSELF HAS
SEEN IT HAPPEN THAT THEY CAN AND HE'S BEEN INVOLVED WITH GROUPS OF OTHER MINORITIES
```

The simplest way to create PL-A transcriptions is to type them directly into a basic text editor program (like TextEdit or Notepad). The PL-A developers have also provided Python scripts that can be used to prepare transcription files from tab-delimited text files or from transcriptions made in ELAN or Praat. Information on these scripts can be found at https://github.com/prosodylab/prosodylab.alignertools/blob/master/README.md.

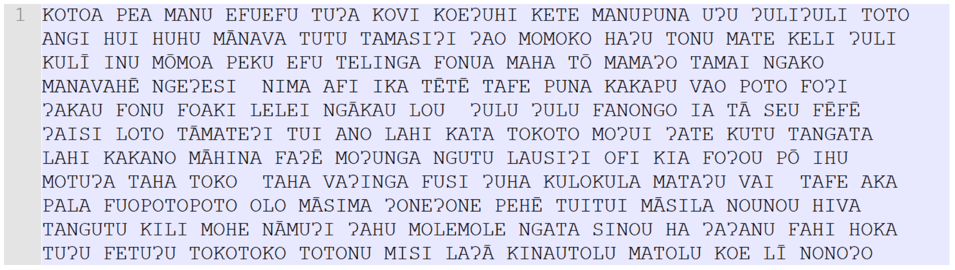
We began our transcription process in ELAN using a controlled vocabulary based on the word list and setting loose boundaries around individual words. For clean file transcriptions, we exported the relevant tier as a TextGrid then used Praat to edit the

[16]On the same trip, two other recordings were made accidentally using the built-in microphones on the recorder rather than the lavalier microphones. These recordings were excluded from the training and alignment tests because of the high level of background noise and relatively poor recording of the speaker's voice.

[17]As noted in §3.2.1 and §6, one way of constraining alignment in PL-A is to use very short audio and transcription files, as recommended by the software developer.

transcriptions, leaving labels only in the intervals containing clear tokens without background noise. Extracting non-empty intervals, concatenating recoverably, and writing to table produced transcriptions (paired with edited audio files) that could then be edited in Word and Notepad++. The transcriptions of dirty files were exported from ELAN as tab-delimited text and then formatted in Word and Notepad++. Traditional Tongan orthography was used, with one exception: glottal stops were represented with an IPA (Unicode) character <ʔ> rather than with the traditional symbol <‘>, since apostrophes and quotation marks can cause problems when running scripts (Figure 2).

Figure 2. Transcription example (Tongan word list, from Tongan Ethnolinguistic Project)



```
KOTOA PEA MANU EFUEFU TUʔA KOVI KOEʔUHI KETE MANUPUNA UʔU ʔULIʔULI TOTO
ANGI HUI HUHU MĀNAVA TUTU TAMASIʔI ʔAO MOMOKO HAʔU TONU MATE KELI ʔULI
KULĪ INU MŌMOA PEKU EFU TELINGA FONUA MAHA TŌ MAMAʔO TAMAI NGAKO
MANAVAHĒ NGEʔESI  NIMA AFI IKA TĒTĒ TAFE PUNA KAKAPU VAO POTO FOʔI
ʔAKAU FONU FOAKI LELEI NGĀKAU LOU  ʔULU ʔULU FANONGO IA TĀ SEU FĒFĒ
ʔAISI LOTO TĀMATEʔI TUI ANO LAHI KATA TOKOTO MOʔUI ʔATE KUTU TANGATA
LAHI KAKANO MĀHINA FAʔĒ MOʔUNGA NGUTU LAUSIʔI OFI KIA FOʔOU PŌ IHU
MOTUʔA TAHA TOKO  TAHA VAʔINGA FUSI ʔUHA KULOKULA MATAʔU VAI  TAFE AKA
PALA FUOPOTOPOTO OLO MĀSIMA ʔONEʔONE PEHĒ TUITUI MĀSILA NOUNOU HIVA
TANGUTU KILI MOHE NĀMUʔI ʔAHU MOLEMOLE NGATA SINOU HA ʔAʔANU FAHI HOKA
TUʔU FETUʔU TOKOTOKO TOTONU MISI LAʔĀ KINAUTOLU MATOLU KOE LĪ NONOʔO
```

3.2.3 Dictionary file The PL-A download includes a North American English dictionary file based on the CMU dictionary. The creators also maintain an online repository for dictionaries at https://github.com/prosodylab/prosodylab.dictionaries. To train acoustic models for other languages, users must provide their own pronunciation dictionary, in plain text format, UTF-8 encoding, saved with a .dict extension. The dictionary must follow the prescribed format as in Figure 3: word entry followed by phones separated by single spaces. Each entry appears on a separate line. Note that alternate pronunciations for the same word appear on separate lines. When creating a dictionary file, you can choose your own transcription system for pronunciations. However, if you use special characters, they need to be Unicode compatible. You will provide PL-A with the key to reading your pronunciations in the configuration file, described in §3.2.4.

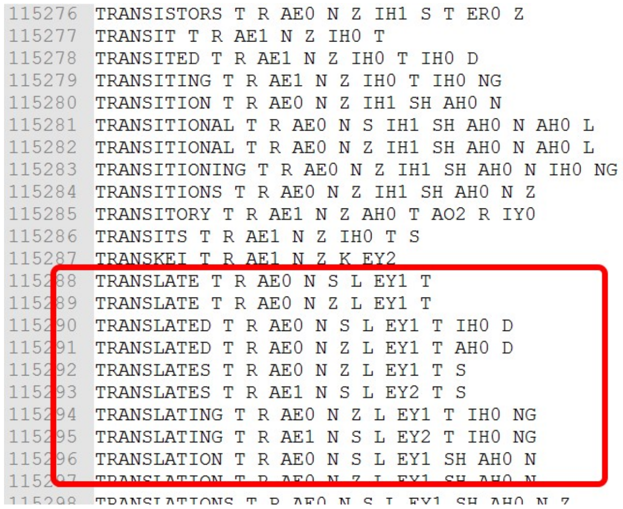
Crucially, the dictionary entries must be sorted according to the Python sort order, which may not match the sort performed by other software or the traditional alphabetical order in printed dictionaries for your language. Proper sorting can be achieved using the command "sort.py," which invokes the Python *sorted()* function supplied by PL-A.[18] Information on Python scripts that can create a dictionary file from transcriptions can be found at https://github.com/prosodylab/prosodylab.alignertools/blob/master/README.md.

Because Tongan orthography is characterized as being phonemic (Shumway 2009), and because little research is available on variation, the pronunciations in the test

[18]For information on this function, refer to: https://wiki.python.org/moin/HowTo/Sorting.

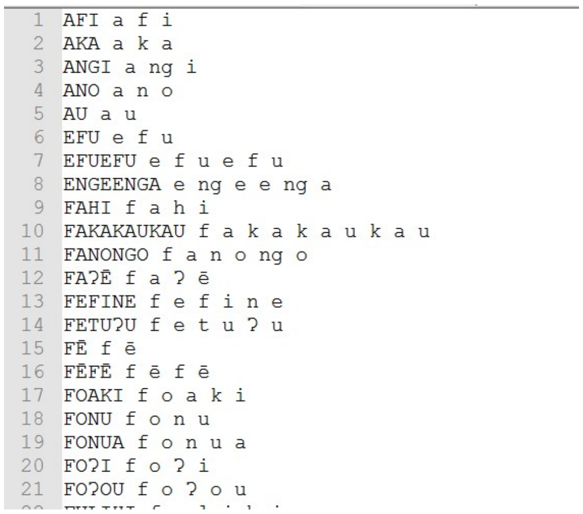
dictionary were based solely on the traditional spelling (Churchward 1959). For the pilot test on Tongan data, we created a dictionary file containing only the words used in the word list[19] task (TonganWL.dict). An excerpt is shown in Figure 4.

Figure 3. Dictionary file example (English, included in download)



```
TRANSISTORS T R AE0 N Z IH1 S T ER0 Z
TRANSIT T R AE1 N Z IH0 T
TRANSITED T R AE1 N Z IH0 T IH0 D
TRANSITING T R AE1 N Z IH0 T IH0 NG
TRANSITION T R AE0 N Z IH1 SH AH0 N
TRANSITIONAL T R AE0 N S IH1 SH AH0 N AH0 L
TRANSITIONAL T R AE0 N Z IH1 SH AH0 N AH0 L
TRANSITIONING T R AE0 N Z IH1 SH AH0 N IH0 NG
TRANSITIONS T R AE0 N Z IH1 SH AH0 N Z
TRANSITORY T R AE1 N Z AH0 T AO2 R IY0
TRANSITS T R AE1 N Z IH0 T S
TRANSKEI T R AE1 N Z K EY2
TRANSLATE T R AE0 N S L EY1 T
TRANSLATE T R AE0 N Z L EY1 T
TRANSLATED T R AE0 N S L EY1 T IH0 D
TRANSLATED T R AE0 N Z L EY1 T AH0 D
TRANSLATES T R AE0 N Z L EY1 T S
TRANSLATES T R AE1 N S L EY2 T S
TRANSLATING T R AE0 N Z L EY1 T IH0 NG
TRANSLATING T R AE1 N S L EY2 T IH0 NG
TRANSLATION T R AE0 N S L EY1 SH AH0 N
TRANSLATION T R AE0 N Z L EY1 SH AH0 N
TRANSLATIONS T R AE0 N S L EY1 SH AH0 N Z
```

Figure 4. Dictionary file example (Tongan, created from word list task)

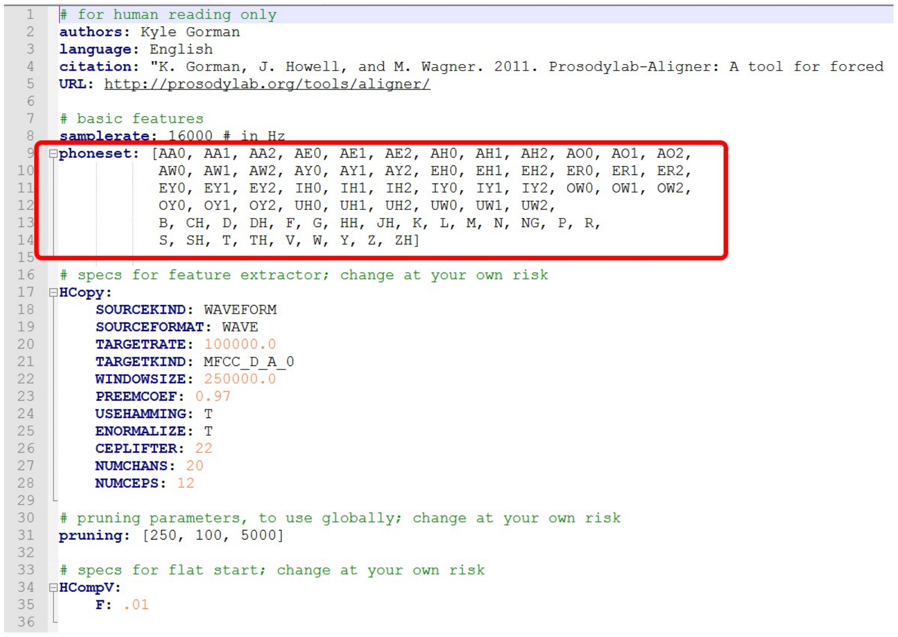


```
AFI a f i
AKA a k a
ANGI a ng i
ANO a n o
AU a u
EFU e f u
EFUEFU e f u e f u
ENGEENGA e ng e e ng a
FAHI f a h i
FAKAKAUKAU f a k a k a u k a u
FANONGO f a n o ng o
FAʔĒ f a ʔ ē
FEFINE f e f i n e
FETUʔU f e t u ʔ u
FĒ f ē
FĒFĒ f ē f ē
FOAKI f o a k i
FONU f o n u
FONUA f o n u a
FOʔI f o ʔ i
FOʔOU f o ʔ o u
```

[19]The word list used in the recordings is a modification of Tongan Swadesh List (The Rosetta Project and the Long Now Foundation 2010). The original 180-word list was reduced to 130 words for some speakers to reduce the overall length of the interview. Most speakers read the word list two or three times.

3.2.4 Configuration file [20] The configuration file provides the instructions and settings to be used in the model training process. The file contains a *phoneset* to be used with the dataset, as well as a few settings. The phoneset is the list of characters used in the pronouncing dictionary file described in §3.2.3. The aligner package includes a configuration file for aligning English transcriptions to recordings. The phoneset in the English configuration file uses ARPAbet, the transcription conventions used in the CMU pronunciation dictionary. Training acoustic models and aligning transcriptions in other languages requires a separate configuration file, in plain text format, UTF-8 encoding, saved with a .yaml extension.[21] The following screenshot shows the phoneset and default settings in the English configuration file.

Figure 5. Configuration file example (English, included in PL-A download)



```
# for human reading only
authors: Kyle Gorman
language: English
citation: "K. Gorman, J. Howell, and M. Wagner. 2011. Prosodylab-Aligner: A tool for forced
URL: http://prosodylab.org/tools/aligner/

# basic features
samplerate: 16000 # in Hz
phoneset: [AA0, AA1, AA2, AE0, AE1, AE2, AH0, AH1, AH2, AO0, AO1, AO2,
           AW0, AW1, AW2, AY0, AY1, AY2, EH0, EH1, EH2, ER0, ER1, ER2,
           EY0, EY1, EY2, IH0, IH1, IH2, IY0, IY1, IY2, OW0, OW1, OW2,
           OY0, OY1, OY2, UH0, UH1, UH2, UW0, UW1, UW2,
           B, CH, D, DH, F, G, HH, JH, K, L, M, N, NG, P, R,
           S, SH, T, TH, V, W, Y, Z, ZH]

# specs for feature extractor; change at your own risk
HCopy:
    SOURCEKIND: WAVEFORM
    SOURCEFORMAT: WAVE
    TARGETRATE: 100000.0
    TARGETKIND: MFCC_D_A_0
    WINDOWSIZE: 250000.0
    PREEMCOEF: 0.97
    USEHAMMING: T
    ENORMALIZE: T
    CEPLIFTER: 22
    NUMCHANS: 20
    NUMCEPS: 12

# pruning parameters, to use globally; change at your own risk
pruning: [250, 100, 5000]

# specs for flat start; change at your own risk
HCompV:
    F: .01

```

In general, the user doesn't need to adjust the settings in the configuration file, hence the warning to "change at your own risk."

We created the configuration file for our Tongan tests (Figure 6) by making the following modifications to the English file:

1. A Tongan phoneset was substituted for the English, including a digraph (ng)[22] for a velar nasal and Unicode characters for long vowels and the glottal stop.

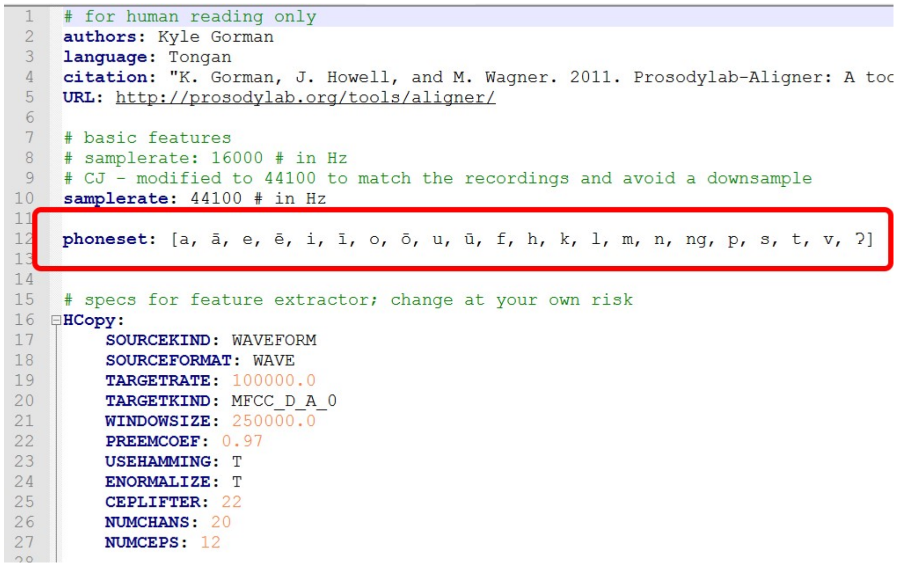
[20]The configuration file is described in current instructions (https://github.com/prosodylab/Prosodylab-Aligner/blob/master/README.md) but is not mentioned in the online tutorial videos for earlier versions (http://prosodylab.org/tools/aligner/). People learning about the tool from video tutorials should keep this in mind.

[21]If you choose to use ARPAbet for a language other than English, it would have to be modified. IPA characters in Unicode may be a better option.

[22]We chose to use the digraph in order to follow Tongan orthographic convention. Alternatively, the Unicode <ŋ> could have been used.

2. To avoid resampling, we changed the sample rate setting to 44100 Hz.[23]
3. For some of the tests we made a slight modification to the TARGETRATE[24] setting as described in §3.3.2.

Figure 6. Configuration file example (Tongan, created by modifying the English file)



```
# for human reading only
authors: Kyle Gorman
language: Tongan
citation: "K. Gorman, J. Howell, and M. Wagner. 2011. Prosodylab-Aligner: A toc
URL: http://prosodylab.org/tools/aligner/

# basic features
# samplerate: 16000 # in Hz
# CJ - modified to 44100 to match the recordings and avoid a downsample
samplerate: 44100 # in Hz

phoneset: [a, ā, e, ē, i, ī, o, ō, u, ū, f, h, k, l, m, n, ng, p, s, t, v, ʔ]


# specs for feature extractor; change at your own risk
HCopy:
    SOURCEKIND: WAVEFORM
    SOURCEFORMAT: WAVE
    TARGETRATE: 100000.0
    TARGETKIND: MFCC_D_A_0
    WINDOWSIZE: 250000.0
    PREEMCOEF: 0.97
    USEHAMMING: T
    ENORMALIZE: T
    CEPLIFTER: 22
    NUMCHANS: 20
    NUMCEPS: 12
```

3.2.5 Data preparation: Troubleshooting The formats for all of these files are simple, but we found that small issues can cause big problems, and solutions may not be immediately obvious.

- The requirement that dictionary entries be listed according to the Python sort order, for example, is not well documented, though noncompliance will cause the aligner to crash.
- Saving files with UTF-8 encoding seems straightforward, but a word processing program (such as Word) that allows users to save files in plain text with UTF-8 encoding may insert a byte order mark (BOM) at the beginning of the document. The BOM is not visible, but it will also cause the aligner to crash. Therefore, it's important to save all text files used with PL-A without the BOM. If you are using a PC, the free program Notepad++ (https://notepad-plus-plus.org/) provides an option to "Encode in UTF-8" (as opposed to "Encode in UTF-8-BOM"), which will produce usable text files.[25]

[23] See the information in §3.2.1 and footnote 11 about resampling.

[24] Because of constraints in place when the first version of HTK was created, TARGETRATE is measured in 100 nanosecond units. That means that when the TARGETRATE is set at the default of 100000, measurements are extracted every 10 msec. (Young et al., 1995–2006:62, footnote 1).

[25] One reviewer recommends UniRed (http://www.esperanto.mv.ru/UniRed/ENG/index.html) as another convenient Unicode text editor.

- Further problems can be caused by a carriage return at the end of a file. Like the BOM problem, this causes the aligner to crash, but the error message does not specifically identify the problem or specify a solution. A carriage return can be removed easily in any text editing program in the way that you would normally remove an extra line. We used Notepad++.

- Finally, some programs automatically add a .txt extension to text files, even when the filename already includes another extension, such as the .lab, .yaml, and .dict extensions required by PL-A. If this occurs, you must remove .txt for the aligner to run. Note: If your operating system preferences are set to hide extensions, you might not be aware of the problem. You may have to change your preference settings in order to reveal and remove the superfluous extensions.[26]

These observations are not included to discourage use of Prosodylab-Aligner. On the contrary, we hope that being aware of possible problems will help others avoid or resolve them quickly. Slight improvements to the aligner's error reporting mechanisms would also make it easier for non-programmers to use.

3.3 Training: Creating an acoustic model

3.3.1 The model training process The model training process produces an acoustic model that will be used to align transcriptions to audio. Model training is executed by entering a Python script in a command line interface (Terminal on a Mac or Command Prompt on a PC). The directory should be set to the Prosody folder, which contains all of the scripts and tools downloaded during the installation process. When entering the command for the script, the user specifies the following:

- the path to the configuration file
- the path to the dictionary file
- the path to the folder containing training data
- the name of the .zip file to which the model will be written

The script requires that all training data – the full set of audio and transcription files – be located in the same folder. Because the aligner relies on filenames to associate transcriptions with the audio, each pair of audio and transcription files must have the same name. (The extensions, of course, will be different: .wav for audio and .lab for transcription text.) Though it's not strictly necessary, Gorman (p.c.) recommends

[26]An additional problem arose from copying characters between specific programs. To minimize input errors, transcriptions for a short reading passage were sometimes copied directly from the PowerPoint presentation used to present the stimuli for the task. However, PowerPoint does not properly read Unicode characters, so symbols for long vowels were improperly imported into the transcriptions and needed to be replaced later. This was an easy problem to fix, but identifying the problem was somewhat time-consuming.

using the ISO 639-3 three-character code (SIL International 2015) for the language in the name of the .zip file for the acoustic model.

The model training is accomplished in cycles, with a set number of iterations, or *epochs*, in each cycle. The default number of epochs is five, but another number can be specified. Because the aligner performs three rounds of training, increasing the number of epochs by one will triple the time required to complete the process. Finally, the user can also request "verbose" or even "more verbose" output when entering the command. We recommend the latter. The training process begins when the command is entered. When the system encounters no problems, progress is indicated by status updates in the interface ("Preparing corpus," "Training iteration 1," etc.).

The quality of the acoustic model can be affected by many factors, including the amount of data, the quality of the data, and various parameter settings. To test the effects of multiple variables, we used our word list recordings to produce the eight test models shown in Table 4. We call each run of the training procedure a "training test" and identify the input data, the settings, and the output model associated with each test in the table.

Table 4. Summary of training tests and runtimes

Test ID #	Type and Number of Audio Files	# of Epochs	TARGETRATE	Name of Acoustic Model Created	Runtime
TonT001	clean (22 files)	5	100000	ton-001-mod.zip	1:04:43
TonT002	clean (22 files)	10	100000	ton-002-mod.zip	0:28:45
TonT003	clean (22 files)	15	100000	ton-003-mod.zip	1:00:49
TonT004	dirty (16 files)	5	125000	ton-004-mod.zip	1:11:05
TonT005	clean & dirty (38 files)	5	125000	ton-005-mod.zip	1:44:00
TonT006	clean (22 files)	5	125000	ton-006-mod.zip	0:17:52
TonT010	clean (17 files)	5	100000	ton-010-mod.zip	0:16:00
TonT011	dirty (11 files)	5	125000	ton-011-mod.zip	0:18:00

Each test is given a unique ID number. Test IDs beginning "TonT" are training tests, each of which produced one acoustic model.[27] Comparisons between models are discussed in §4 below.

The length of time required to train a model varies, but in our tests the process took from 16 minutes to one hour 44 minutes, with most tests being completed in under 30 minutes.[28] Factors influencing the runtime may include the amount of data, the number of epochs specified, and the processing power of the computer. Using the computer for other tasks while the training module is running may affect available

[27]The test numbers are discontinuous here because several models were trained using reading passage recordings before the final two training tests using only word list data. Because the models using reading passage recordings have not been properly tested yet, those have been omitted from these results. As mentioned in a previous footnote, the last two training tests were performed on a Surface Pro 4 rather than a Surface Pro 3. See footnote 9 for complete specifications.

[28]The runtime for the first test was unusually long, considering the amount of data and the specified settings. We think that the computer went to sleep in the middle of the process, so this runtime should not be considered representative.

computing power. Note that the PL-A process runs on a single core, so running it on a machine with multiple cores will not significantly alter processing speed.

3.3.2 Troubleshooting the training process

- The training process will fail if the files or directories have not been properly prepared or if the syntax for the command is not correct. In such cases, an error message may identify the source of the problem (e.g., "No such file or directory" or "Formatting error in dictionary").

- One common error results from a mismatch between the transcription files and the pronunciation dictionary. If any of the transcription files contains a word not listed in the dictionary – either because of a transcription error or because of a missing entry in the dictionary file – the training process will abort. To help users resolve this issue, PL-A produces a list of out-of-dictionary words in the form of a text file called OOV.txt. Unfortunately, that file does not indicate which transcription file(s) contain(s) the out-of-dictionary word(s), making it difficult to locate the problem in a large data set.[29] To improve this process, we added a few lines of code to include the name of the file(s) with the missing word(s) in the output.[30] (In case this will be useful to others, we will submit this as a pull request to the PL-A GitHub.) On a UNIX-like (e.g. Mac OS X) system, the *grep*[31] *-l* function can help the user identify which .lab files contain out-of-dictionary words (Gorman, p.c.). On a PC, users can search all files in a specified folder for problem words using Notepad++.

- When training models using dirty files, we encountered one error ("ERROR [+7390] StepAlpha: Alpha prune failed") that was not documented in the PL-A materials or the HTK Book (Young et al. 1995–2006). After a web search, we resolved the issue by increasing the TARGETRATE setting in the configuration file from 100000 (default) to 125000.[32] This change resulted in acoustic feature measurements being extracted every 12.5 ms rather than every 10 ms. Potential effects of this change are considered in §4 below.

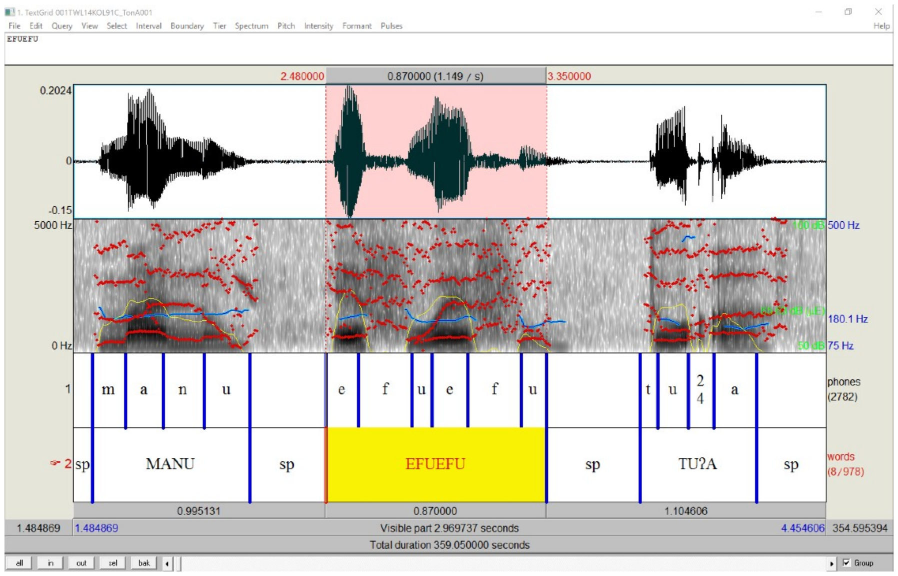
3.4 Alignment: Producing time-aligned TextGrids

3.4.1 The alignment process Once you have created an acoustic model for your research language, you can use that model to produce two-tiered time-aligned Praat TextGrids, as in Figure 7. The "words" tier shows the boundaries between words and indicates sections of silence (labeled "sp" between words and "sil" at the beginning

[29]This problem has been corrected in the Prosody Lab's newer product, Montreal Forced Aligner (see footnote 7).
[30]Code written by Craig Johnson.
[31]One reviewer notes that grep can also be installed on Windows with the GnuWin32 tools (http://gnuwin32.sourceforge.net/).
[32]See footnote 24.

and the end of the file). The "phones" tier subdivides each word into individual segments based on the sequence of phones given in the pronunciation dictionary file.

Figure 7. Sample TextGrid with two tiers, created by Prosodylab-Aligner



Like the training process, the alignment process is initiated by executing the Python script in Terminal or Command Prompt. Again, the directory should be set to the Prosody folder. The command specifies:

- the name and path of the acoustic model (.zip file) to be used
- the path to the dictionary file
- the path to the folder containing the audio and transcription files to be aligned

The TextGrids are created in the same folder as the sound and transcription files and are given the same names, with the .TextGrid extension. The system also creates a .csv file listing a confidence score for each file. Gorman (p.c.) describes these as log probabilities for each alignment (for the file as a whole), but they are not well documented in either the PL-A instructions or the HTK Book. So far, we have not found these to be useful, since they are very difficult to interpret.[33]

The alignment process is quicker than the training process. Among our test alignments, the longest took about 52 minutes (for 38 files, 4:21:14 total recording time), and the shortest four minutes (for five files, 1:23:01 total recording time). We conducted 16 alignment tests using the eight models created during the training tests, as

[33]The aligner also creates a text file (.aligned.mlf) listing timestamps for all of the boundaries placed during that alignment session, written as integers. The boundaries are listed as beginning and endpoints of segments on the "phones" tier, and the label for the segment is included. This is an intermediate file the aligner produces before creating TextGrids. On a Mac system, this file is hidden (Gorman, p.c.). However, if you run the aligner on a PC, you will see this file in the folder with your TextGrids.

summarized in Table 5. Each alignment test is given a unique ID number that begins with "TonA."[34] A complete explanation of each test and comparisons of results are included in §4, below.

Table 5. Summary of alignment tests and runtimes

Test ID #	Type and Number of Aligned Files	Acoustic Model Used in Alignment (and Type of Training Files)	# of Epochs	TARGETRATE	Runtime
TonA001	clean (22)	ton-001-mod.zip (trained on clean)	5	100000	0:13:19
TonA002	clean (22)	ton-002-mod.zip (trained on clean)	10	100000	0:12:45
TonA003	clean (22)	ton-003-mod.zip (trained on clean)	15	100000	0:20:20
TonA004	dirty (16)	ton-001-mod.zip (trained on clean)	5	100000	0:36:58
TonA005	clean & dirty (38)	ton-004-mod.zip (trained on dirty)	5	125000	0:30:45
TonA006	clean & dirty (38)	ton-005-mod.zip (trained on clean & dirty)	5	125000	0:51:50
TonA007	dirty (16)	ton-002-mod.zip (trained on clean)	10	100000	0:25:50
TonA008	dirty (16)	ton-003-mod.zip (trained on clean)	15	100000	0:26:20
TonA009	dirty (5)	ton-001-mod.zip (trained on clean)	5	100000	0:17:15
TonA010	dirty (5)	ton-002-mod.zip (trained on clean)	10	100000	0:18:55
TonA011	dirty (5)	ton-003-mod.zip (trained on clean)	15	100000	0:26:00
TonA012	dirty (5)	ton-004-mod.zip (trained on dirty)	5	125000	0:14:00
TonA013	dirty (5)	ton-005-mod.zip (trained on clean & dirty)	5	125000	0:11:40
TonA014	clean & dirty (43)	ton-006-mod.zip (trained on clean)	5	125000	0:39:10
TonA017	clean (5)	ton-010-mod.zip (trained on clean)	5	100000	0:04:00
TonA018	dirty (5)	ton-011-mod.zip (trained on dirty)	5	125000	0:05:00

3.4.2 Troubleshooting the alignment process

- As in the training process, problems with the script syntax or in the location or format of files will terminate the alignment process and produce an error message.

[34]The test numbers are discontinuous, as explained in footnote 27.

- In our experience, the alignment module gave no feedback indicating status when it was progressing normally. We monitored the progress using the Processes tab in Task Manager.

- When opened in Praat, the TextGrids show one strange inconsistency between the tiers. In the "words" tier, the labels match the format of the transcriptions and dictionary file. The macrons over long vowels and the Unicode glottal stop character <ʔ> display correctly. However, the Unicode characters do not display correctly in the "phones" tier. Where Unicode characters were used for pronunciations in the dictionary file, a code is displayed in the TextGrid. For example, the Unicode <ʔ> appears as <24> on the tier display. When that interval is selected, the Praat text window shows 1224. This is not a serious problem, though: the codes are consistent for each character, the identity of the character is obvious from the word transcription, and the problem can be corrected with a search and replace operation or script. However, it is worth noting, since any scripts for extracting measurements would need to reference the label as it appears in the TextGrid file.

4. Alignment comparisons So far, we have explained how to use PL-A (1) to create acoustic models for a language using field recordings and (2) to align orthographic transcriptions of those recordings to the audio signal. We have also described our specific experience using PL-A to train an acoustic model based on our Tongan word list recordings and to produce time-aligned Praat TextGrids for those recordings. In this section, we discuss the comparisons we made between alignments to assess the effects of different variables on the training and alignment processes. The alignments we created are listed in Table 5, above.

In §4.1, we explain the logic behind each test comparison, the variables involved, and the apparent effects. To illustrate the qualitative differences between alignments, we include screenshots of sample TextGrids for each comparison we discuss. The screenshots are just samples and may or may not be representative of the full corpus. They do, however, match our impressions as we reviewed sections of alignments for several speakers, visually inspecting the TextGrids at different zoom levels and listening to words, phones, and "silences" as segmented by PL-A. For most of our tests, the clean set of files and the dirty set included different speakers. Therefore, the clean file alignment examples in this section are for Speaker 003's recording, and the dirty file alignment examples are for Speaker 026's recording. (See §3.2.1 for an explanation of the dirty and clean files.) The speaker number is in the top left corner of each screenshot. The tier names (on the right side of the screenshot) reflect the alignment test numbers in the comparisons.

Each screenshot is followed by a table that includes timestamps for each boundary in the screenshot, as well as the difference in boundary placement between the models in the figure. The table only includes information for boundaries shown in the screenshot and is included as an example. The mean and median figures may not be representative of the full recording or of other speakers' recordings.

In §4.2 we make quantitative comparisons between the alignments to assess how large an effect each change in variable produced. In that section, we refer to the comparisons and variables discussed in §4.1. However, the quantitative comparisons offered in §4.2 take into account the full word list recording for each speaker, rather than just the snapshots presented in §4.1. Because we are ultimately interested in using forced-aligned data to feed automated phonetic analysis, our quantitative discussion compares extracted measurements of F1 and F2, common measures used in sociophonetic study.

4.1 Comparisons of test alignments Sections 4.1.1 through 4.1.4 describe the key variables in our alignment tests and the implications associated with each. Because we were most concerned about the effects of file cleanup and the amount of time required to prepare data, we discuss the clean/dirty comparisons first. The other comparisons relate to changes in settings and to the effects of using the same dataset for training and alignment.

4.1.1 Clean vs. dirty The most basic distinction in the tests is whether the audio files are clean or dirty. As explained in §3.2.1, files in the former category have been cleaned up extensively, eliminating extraneous background noises between words (e.g., wind, cars, roosters, pigs) and removing any tokens that could be problematic because of background noise. Dirty files retain almost all of the background noise. The basic questions related to the clean/dirty comparisons are these:

- Does PL-A do a better job when the *files used in model training* have been cleaned?
- Does PL-A do a better job when the *files to be aligned* have been cleaned?
- Ultimately, is it worth the time and effort to clean up files before training and/or alignment?

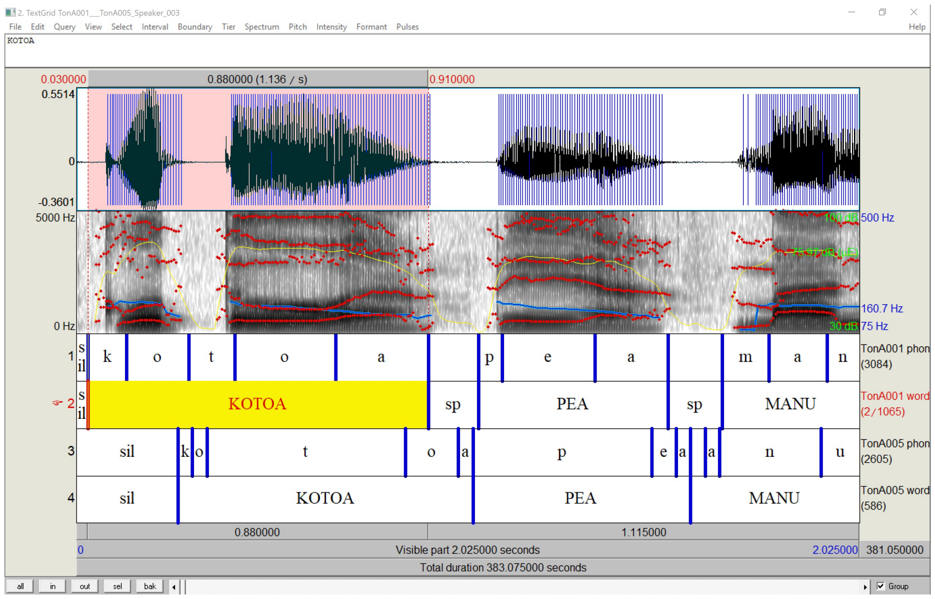
Because training and alignment are completed in two steps, we needed to assess the effect of using clean files at each stage. Table 6 summarizes the tests we performed and the kinds of files we used at each stage. The alignment tests are referenced by test ID (e.g., TonA001), and speakers are identified by three-digit reference numbers.

Table 6. Basic clean/dirty test summary

	Models Trained on Clean Files	**Models Trained on Dirty Files**	**Models Trained on Clean and Dirty Files**
Aligning Clean Files	TonA001 (Aligned Speakers 001-023*)	TonA005 (Aligned Speakers 001-023*)	TonA006 (Aligned Speakers 001-023*)
Aligning Dirty Files	TonA004 (Aligned Speakers 024-041**)	TonA005 (Aligned Speakers 024-041**)	TonA006 (Aligned Speakers 024-041**)

*There are no recordings for Speaker 008.
**Recordings for speakers 032 and 033 were excluded because of different recording conditions.

Figure 8. Clean file alignment based on a clean model (*top*, TonA001) and on a dirty model (*bottom*, TonA005) – beginning of recording



When comparing the alignments created in each test, we found that models trained on clean files produced much better alignments than those trained on dirty files. In the TextGrid comparison in Figure 8, the top alignment (TonA001) was based on a model trained on clean data, while the bottom alignment (TonA005) was based on a model trained on dirty files. In both cases, the transcription was being aligned to the same clean audio file for Speaker 003.

The screenshot shows the first words in the word list recording and demonstrates that the alignment based on the clean model is reasonable, while the alignment based on the dirty model is wrong from the very beginning. Table 7 compares the placement of boundaries shown in Figure 8 and shows a mean difference of 192 ms and a median difference of 173 ms for these boundaries.

The screenshot in Figure 9 shows the same two alignments for the last word of the sound file. Over six and a half minutes into the recording, the boundaries in the top alignment (based on the clean model) still appear reasonable, and those in the bottom alignment (based on the dirty model) are still misaligned. In fact, Table 8 shows that the boundary differences at the end of the recording are even greater than at the beginning.

We see similar results when aligning dirty files. The comparison in Figure 10 shows two TextGrids created for the same dirty recording. The top alignment (TonA-004) is based on a model trained on clean files. The bottom alignment (TonA005) is based on a model trained on dirty files.

In this example, the clean model alignment on top looks quite good, though the aligner seems to be cutting off the very end of the words. However, the dirty model alignment on the bottom failed to even find the words. The large differences in boundary placement summarized in Table 9 suggest that using a clean model is even more important when aligning dirty files than when aligning clean ones. (Though we don't include a separate screenshot, these models continue in much the same way throughout the recording, and an example from the end would support the analysis here.)

Table 7. Differences in boundary placement for screenshot in Figure 8 – clean alignment based on clean model (TonA001) and dirty model (TonA005) – beginning of recording

Boundary	TonA001 Timestamp	TonA005 Timestamp	Difference
sil/k	0.03	0.2625	233 ms
k/o	0.13	0.3000	170 ms
o/t	0.29	0.3375	48 ms
t/o	0.41	0.8500	440 ms
o/a	0.67	0.9875	318 ms
a/sp	0.91	1.0250	115 ms
sp/p	1.04	1.0250	15 ms
p/e	1.10	1.4875	388 ms
e/a	1.34	1.5500	210 ms
a/sp	1.53	1.5875	57 ms
sp/m	1.67	1.5875	83 ms
m/a	1.79	1.6250	165 ms
a/n	1.94	1.6625	278 ms
n/u	2.10	1.9250	175 ms
mean			192 ms
median			173 ms

Table 8. Differences in boundary placement for screenshot in Figure 9 – clean alignment based on clean model (TonA001) and dirty model (TonA005) – end of recording

Boundary	TonA001 Timestamp	TonA005 Timestamp	Difference
h/i	381.75	382.1250	375 ms
i/n	381.96	382.3625	403 ms
n/a	382.07	382.4000	330 ms
a/sp (h)	382.21	382.5000	290 ms
(a) sp/h	382.35	382.5000	150 ms
h/a	382.48	382.9500	470 ms
a/i	382.69	383.0125	322 ms
i/sil	382.87	383.0500	180 ms
mean			315 ms
median			326 ms

Figure 9. Clean file alignment based on a clean model (*top*, TonA001) and on a dirty model (*bottom*, TonA005) – end of recording

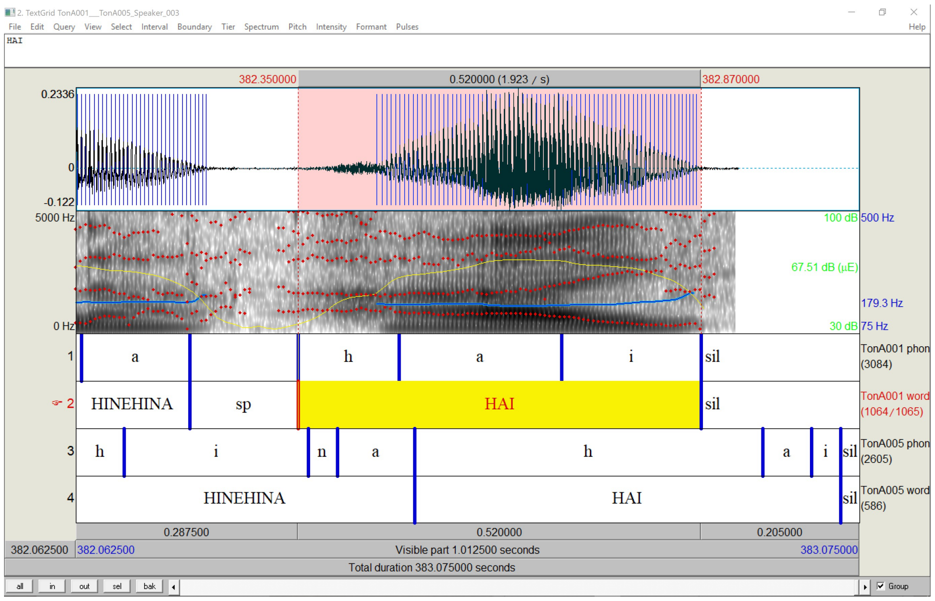
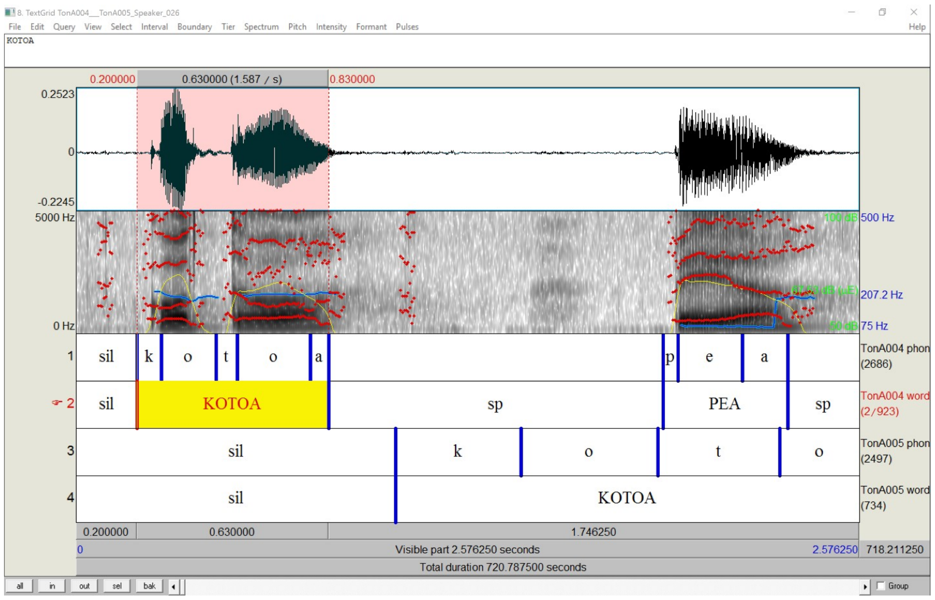


Figure 10. Dirty file alignment based on a clean model (*top*, TonA004) and on a dirty model (*bottom*, TonA005)

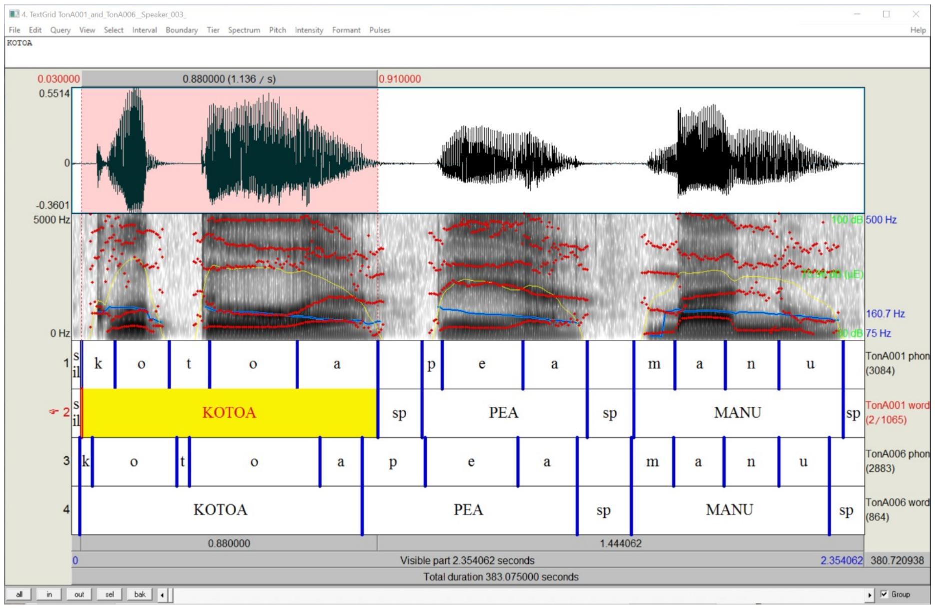


Even including dirty files along with clean files in the training data had a detrimental effect on overall alignment, though this was less problematic when aligning clean files than when aligning dirty files. Consider the TextGrid comparison in Figure 11. The top alignment (TonA001) was based on a model trained on only clean files. The bottom alignment (TonA006) was based on a model trained on both clean and dirty files. In both cases, the transcriptions were aligned to a clean recording, the same one as in Figures 8 and 9.

Table 9. Differences in boundary placement for screenshot in Figure 10 – dirty file alignment based on a clean model (TonA004) and a dirty model (TonA005)

Boundary	TonA004 Timestamp	TonA005 Timestamp	Difference
sil/k	0.20	1.0500	850 ms
k/o	0.28	1.4625	1183 ms
o/t	0.46	1.9125	1453 ms
t/o	0.53	2.3125	1783 ms
o/a	0.77	2.6500	1880 ms
a/sp	0.83	3.3500	2520 ms
sp/p	1.93	3.5125	1583 ms
p/e	1.98	3.7625	1783 ms
e/a	2.19	5.1500	2960 ms
a/sp	2.34	5.2750	2935 ms
mean			1893 ms
median			1783 ms

Figure 11. Clean file alignment based on a clean model (*top*, TonA001) and on a mixed clean and dirty model (*bottom*, TonA006) – beginning of recording


4. TextGrid TonA001_and_TonA006_Speaker_003
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
KOTOA
0.030000
0.880000 (1.136 / s)
0.910000
0.5514
0
-0.3601
5000 Hz
0 Hz
500 Hz
160.7 Hz
75 Hz
1 sil k o t o a p e a m a n u
TonA001 phon (3084)
2 sil KOTOA sp PEA sp MANU sp
TonA001 word (2/1065)
3 k o t o a p e a m a n u
TonA006 phon (2883)
4 KOTOA PEA sp MANU sp
TonA006 word (864)
0.880000
1.444062
0
Visible part 2.354062 seconds
2.354062
380.720938
Total duration 383.075000 seconds
all in out sel bak
Group

The exact differences between boundary placements shown in Figure 11 are summarized in Table 10. The biggest difference shown in this example arises between the words *kotoa* and *pea*. The first alignment appropriately inserts a silence segment between the two words, while the second alignment includes the silence in the *p* segment.

This alignment is not as bad as that shown in Figure 8, which used a model trained on all dirty files. In this case, the aligner did a better job finding *manu* after misaligning *pea*, and the mean and median differences for the boundaries in Table 10 are much smaller than those in Table 8. However, the alignment at the end of the recording is not as good, and we found that including dirty files in the training data generally lowered the quality of clean file alignments.

Table 10. Differences in boundary placement for screenshot in Figure 11 – clean file alignment based on a clean model (TonA001) and on a mixed clean and dirty model (TonA006) – beginning of recording

Boundary	TonA001 Timestamp	TonA006 Timestamp	Difference
sil/k	0.03	0.0250	5 ms
k/o	0.13	0.0625	68 ms
o/t	0.29	0.3125	23 ms
t/o	0.41	0.3500	60 ms
o/a	0.67	0.7375	68 ms
a/sp (p)	0.91	0.8625	48 ms
(a) sp/p	1.04	0.8625	178 ms
p/e	1.10	1.0500	50 ms
e/a	1.34	1.3250	15 ms
a/sp	1.53	1.5000	30 ms
sp/m	1.67	1.6625	7 ms
m/a	1.79	1.7875	2 ms
a/n	1.94	1.9375	2 ms
n/u	2.10	2.1000	0 ms
u/sp	2.29	2.2500	40 ms
mean			40 ms
median			30 ms

Including dirty files in the training data was an even bigger problem when the files to be aligned were dirty.[35] In Figure 12, the top alignment (TonA004) is based on a clean model. This is the same alignment as seen in Figure 10. The bottom alignment (TonA006) is based on a mixed model trained on both clean and dirty files. The boundary differences in the example are summarized in Table 11.

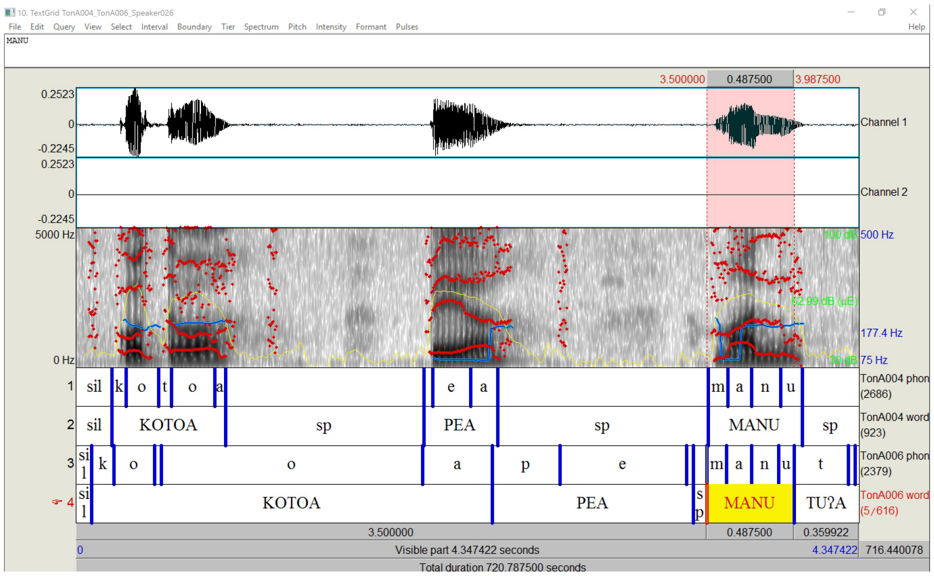
In this example, the model that included dirty files in the training data has considerable difficulty finding the first two words. Although it recovers somewhat for *manu*, similar problems occur throughout the recording. All the dirty alignments based on a mixed clean and dirty model had similar problems. Though they weren't as bad as the alignments based only on dirty training data (compare Table 11 to Table 9),

[35]As a reviewer points out, this is probably due to the training process incorporating noise into the acoustic models which is then found again and associated with certain segments in the alignment phase.

they were problematic enough to cause serious concern. Correcting those alignments would require much more time than correcting the alignments based on only clean data.

The screenshots shown in this section provide only snapshots from alignments for two speakers. However, they illustrate the kinds of differences we observed between clean model and dirty model alignments for other speakers as well. Models trained on clean files generally produced better alignments than those trained on dirty files or on a combination of clean and dirty files, regardless of whether the files to be aligned were clean or dirty. Interestingly, having a clean model was even more important when the files to be aligned were dirty. For our dataset, it seems more important, and time efficient, to clean the files to be used in training (at least an hour of recorded data) than to clean all of the files to be aligned.

Figure 12. Dirty file alignment based on a clean model (*top*, TonA004) and on a mixed clean and dirty model (*bottom*, TonA006) – beginning of recording


10. TextGrid TonA004_TonA006_Speaker026
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
MANU
3.500000 0.487500 3.987500
0.2523 0 -0.2245 Channel 1
0.2523 0 -0.2245 Channel 2
5000 Hz 0 Hz 500 Hz 177.4 Hz 75 Hz
1 sil k o t o a e a m a n u TonA004 phon (2686)
2 sil KOTOA sp PEA sp MANU sp TonA004 word (923)
3 sil k o o a p e m a n u t TonA006 phon (2379)
4 sil KOTOA PEA sp MANU TUʔA TonA006 word (5/616)
3.500000 0.487500 0.359922
0 Visible part 4.347422 seconds 4.347422 716.440078
Total duration 720.787500 seconds

4.1.2 TARGETRATE: 100000 vs. 125000 As explained in §3.3.2, the models trained using dirty files (models trained on only dirty files and models trained on both clean and dirty files) required a different TARGETRATE setting from the default to avoid a software crash. Could what appear to be clean/dirty effects actually result from the difference in TARGETRATE setting? The next comparison was made to answer that question.

In this comparison, the same clean files were aligned based on clean models that differed only in TARGETRATE setting. In TonA001, the TARGETRATE was set at the default 100000; in TonA014, it was set at 125000. Figure 13 illustrates this comparison at the beginning of the word list audio file for Speaker 003.

Table 11. Differences in boundary placement for screenshot in Figure 12 – Dirty file alignment based on a clean model (TonA004) and on a mixed clean and dirty model (TonA006) – beginning of recording

Boundary	TonA004 Timestamp	TonA006 Timestamp	Difference
sil/k	0.20	0.0875	113 ms
k/o	0.28	0.2125	68 ms
o/t	0.46	0.4375	23 ms
t/o	0.53	0.4750	55 ms
o/a	0.77	1.9250	1155 ms
a/sp	0.83	2.3125	1483 ms
sp/p	1.93	2.3125	383 ms
p/e	1.98	2.6875	708 ms
e/a	2.19	3.3875	1198 ms
a/sp	2.34	3.4250	1085 ms
sp/m	3.51	3.5000	10 ms
m/a	3.62	3.6125	8 ms
a/n	3.75	3.7500	0 ms
n/u	3.91	3.9000	10 ms
u/sp	4.03	3.9875	43 ms
sp/t	5.29	3.9875	1303 ms
mean			478 ms
median			90 ms

Table 12. TARGETRATE comparison

Models Trained on Clean Files, at 100000 TARGETRATE	Models Trained on Clean Files, at 125000 TARGETRATE
TonA001 (Aligned Speakers 001-023*)	TonA014 (Aligned Speakers 001-023*)

*There are no recordings for Speaker 008.

The alignments made based on the two models with different TARGETRATE settings are nearly identical at the beginning of Speaker 003's word list recording. Table 13 shows very small differences in boundary placement in the screenshot, with a mean difference of 8 ms and a median difference of 3 ms. The greatest distance shown was in the o/t boundary, at 65 ms. At the beginning of this recording, the alignment based on a clean 125000 TARGETRATE model (Figure 13) is much better than the alignment based on a dirty 125000 TARGETRATE model (Figure 8).

From the snapshot at the beginning of the recording (Figure 13), it is difficult to determine whether the clean 100000 TARGETRATE model or the clean 125000 TARGETRATE model produced the better alignment. The differences are clearer when we look at the alignments for the end of the recording, as shown in Figure 14.

After six and a half minutes, the TextGrid based on the model with the TARGETRATE of 125000 is misaligned. The differences in boundary placement in this screenshot are summarized in Table 14.

Figure 13. Clean file alignment based on a 100000 TARGETRATE model (*top*, TonA001) and a 125000 TARGETRATE model (*bottom*, TonA014) – beginning of recording

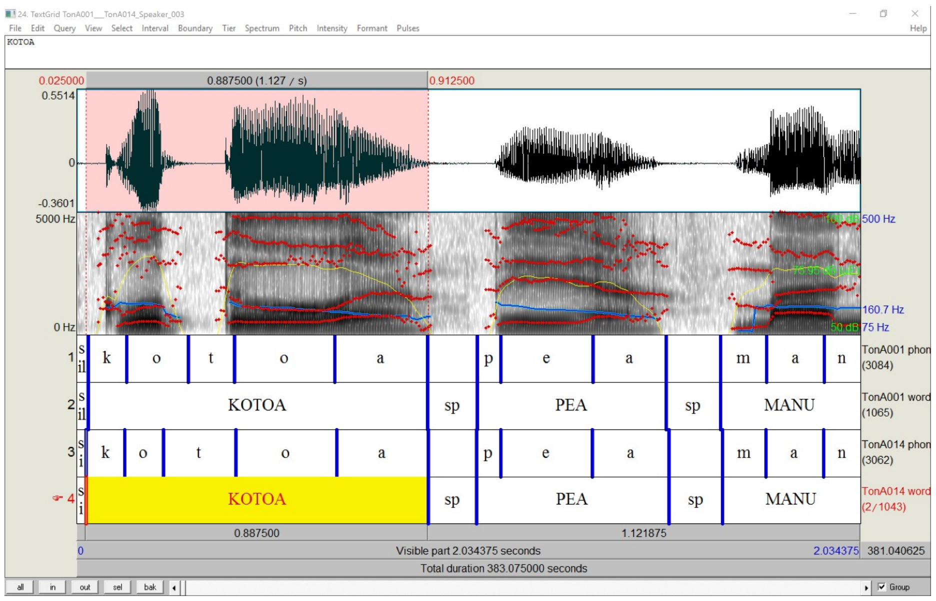

24. TextGrid TonA001__TonA014_Speaker_003
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
KOTOA
0.025000
0.887500 (1.127 / s)
0.912500
0.5514
0
-0.3601
5000 Hz
0 Hz
500 Hz
160.7 Hz
75 Hz
50 dB
1 sil k o t o a p e a m a n
TonA001 phon (3084)
2 sil KOTOA sp PEA sp MANU
TonA001 word (1065)
3 si k o t o a p e a m a n
TonA014 phon (3062)
4 si KOTOA sp PEA sp MANU
TonA014 word (2/1043)
0.887500
1.121875
0
Visible part 2.034375 seconds
2.034375
381.040625
Total duration 383.075000 seconds
all in out sel bak
Group

Table 13. Differences in boundary placement for screenshot in Figure 13 – clean file alignment based on model with 100000 TARGETRATE setting (TonA001) and 125000 TARGETRATE setting (TonA014) – beginning of recording

Boundary	TonA001 Timestamp	TonA014 Timestamp	Difference
sil/k	0.03	0.0250	5 ms
k/o	0.13	0.1250	5 ms
o/t	0.29	0.2250	65 ms
t/o	0.41	0.4125	3 ms
o/a	0.67	0.6750	5 ms
a/sp	0.91	0.9125	2 ms
sp/p	1.04	1.0375	2 ms
p/e	1.10	1.1000	0 ms
e/a	1.34	1.3375	3 ms
a/sp	1.53	1.5375	8 ms
sp/m	1.67	1.6750	5 ms
m/a	1.79	1.7875	2 ms
a/n	1.94	1.9375	2 ms
mean			8 ms
median			3 ms

Figure 14. Clean file alignment based on a 100000 TARGETRATE model (*top*, TonA001) and a 125000 TARGETRATE model (*bottom*, TonA014) – end of recording

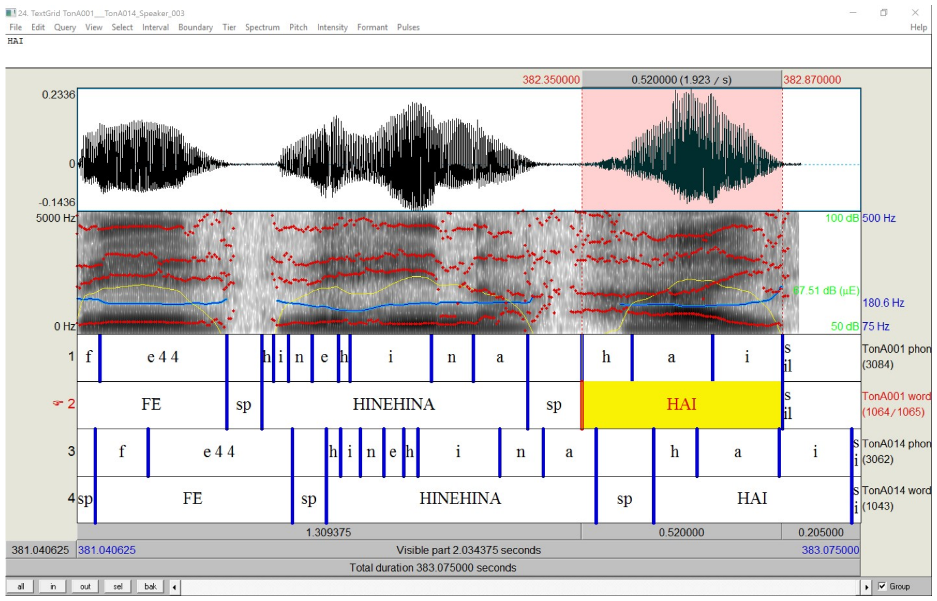

24. TextGrid TonA001__TonA014_Speaker_003
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses
Help
HAI
382.350000
0.520000 (1.923 / s)
382.870000
0.2336
0
-0.1436
5000 Hz
0 Hz
100 dB
500 Hz
97.51 dB (μE)
180.6 Hz
50 dB
75 Hz
f e44 h i n e h i n a h a i sil
FE sp HINEHINA sp HAI sil
f e44 h i n e h i n a h a i si
sp FE sp HINEHINA sp HAI si
TonA001 phon (3084)
TonA001 word (1064/1065)
TonA014 phon (3062)
TonA014 word (1043)
1.309375
0.520000
0.205000
381.040625
Visible part 2.034375 seconds
383.075000
Total duration 383.075000 seconds
all in out sel bak
Group

Table 14. Differences in boundary placement for screenshot in Figure 14 – clean file alignment based on model with 100000 TARGETRATE setting (TonA001) and 125000 TARGETRATE setting (TonA014) – end of recording

Boundary	TonA001 Timestamp	TonA014 Timestamp	Difference
f/e44 (ē)	381.10	381.2250	125 ms
e44 (ē) /sp	381.43	381.6000	170 ms
sp/h	381.52	381.6875	168 ms
h/i	381.55	381.7250	175 ms
i/n	381.59	381.7750	185 ms
n/e	381.65	381.8375	188 ms
e/h	381.72	381.8875	167 ms
h/i	381.75	381.9250	175 ms
i/n	381.96	382.1375	178 ms
n/a	382.07	382.2500	180 ms
a/sp	382.21	382.3875	178 ms
sp/h	382.35	382.5375	188 ms
h/a	382.48	382.6500	170 ms
a/i	382.69	382.8625	173 ms
i/sil	382.87	383.0500	180 ms
mean			173 ms
median			175 ms

It is important to note, however, that the TonA014 misalignment at the end of the recording in Figure 13 is not as problematic as the alignments based on models trained on dirty files (Figure 9 and Table 8). Though TARGETRATE does affect the alignment to some degree, it seems unlikely to account for all of the clean/dirty effects described in §4.1.1.

4.1.3 Same vs. different For most of the tests, we were working with one set of recordings that had been cleaned, and a different set of recordings that had been left dirty. That means that when a model trained on clean files aligned clean files, *it aligned the exact files on which it had been trained.* The same is true for dirty files aligned by a model trained on dirty files. Could what appear to be clean/dirty effects actually result from the aligner encountering the exact same files in both the training and alignment contexts? The next comparison was made to answer that question.[36]

Consider the comparison in Figure 15. The top alignment is TonA001, used in previous examples. The clean model by which the alignment was made was trained on the same files that were to be aligned. The second alignment, TonA017, was aligned using a model that was similar to that used in TonA001, the only difference being that the five files aligned in that test had been removed from the training dataset.[37] (See TonT001 and TonT010 in Table 4, above.)

Table 15. Same vs. different comparisons

	Training Files Same as Aligned Files	**Training Files Different from Aligned Files**
Clean-Clean (Training files and aligned files both clean)	TonA001 · Trained on speakers 001-023* clean · Aligned speakers 001-023* clean · (Compare speakers 001-005 clean)	TonA017 · Trained on speakers 006-023* clean · Aligned speakers 001-005 clean

*There are no recordings for Speaker 008.

The screenshot in Figure 15 shows small alignment differences at the beginning of the recording, with the two models placing nearly half of the shown boundaries at identical locations. The differences in the placement of boundaries shown in this screenshot are summarized in Table 16. The average differences are minor, with a mean of 8 ms and a median of 10 ms.

The screenshot in Figure 16, showing the same two alignments at the end of the recording, also shows many identical boundaries in the two alignments. The differences are summarized in Table 17.

[36]Although PL-A developers advertise its ability to use the same corpus for training and alignment, an attendee at NWAV 44 pointed out that aligners are generally tested using different files for the training and alignment steps. We appreciated the feedback and designed the tests in §4.1.3 in response.

[37]A reviewer notes that removing the training files results in a smaller dataset used for training as well as a training dataset that does not include the files to be aligned, both of which could affect the quality of the model and the resulting alignments. Because removing the five files produced a relatively small effect, we did not perform further tests to determine which factor was more significant.

Figure 15. Clean file alignment based on a clean model that included alignment files in training (*top*, TonA001) and a clean model that did not (*bottom*, TonA017) – beginning of recording

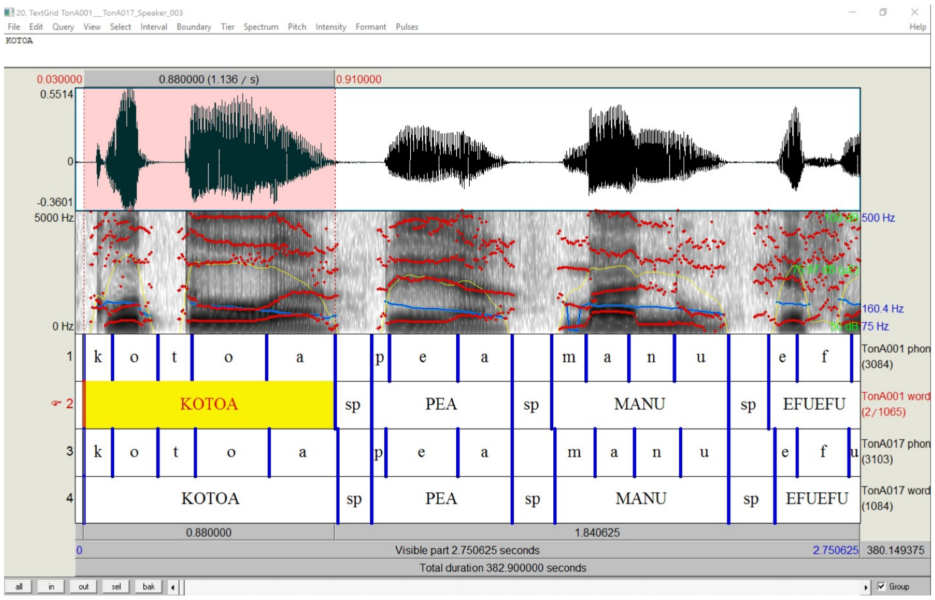

20. TextGrid TonA001__TonA017_Speaker_003
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
KOTOA
0.030000 0.880000 (1.136 / s) 0.910000
0.5514 0 -0.3601
5000 Hz 0 Hz 500 Hz 160.4 Hz 75 Hz
1 k o t o a p e a m a n u e f TonA001 phon (3084)
2 KOTOA sp PEA sp MANU sp EFUEFU TonA001 word (2/1065)
3 k o t o a p e a m a n u e f u TonA017 phon (3103)
4 KOTOA sp PEA sp MANU sp EFUEFU TonA017 word (1084)
0.880000 1.840625
0 Visible part 2.750625 seconds 2.750625 380.149375
Total duration 382.900000 seconds
all in out sel bak Group

Figure 16. Clean file alignment based on model that included alignment files in training (*top*, TonA001) and a model that did not (*bottom*, TonA017) – end of recording

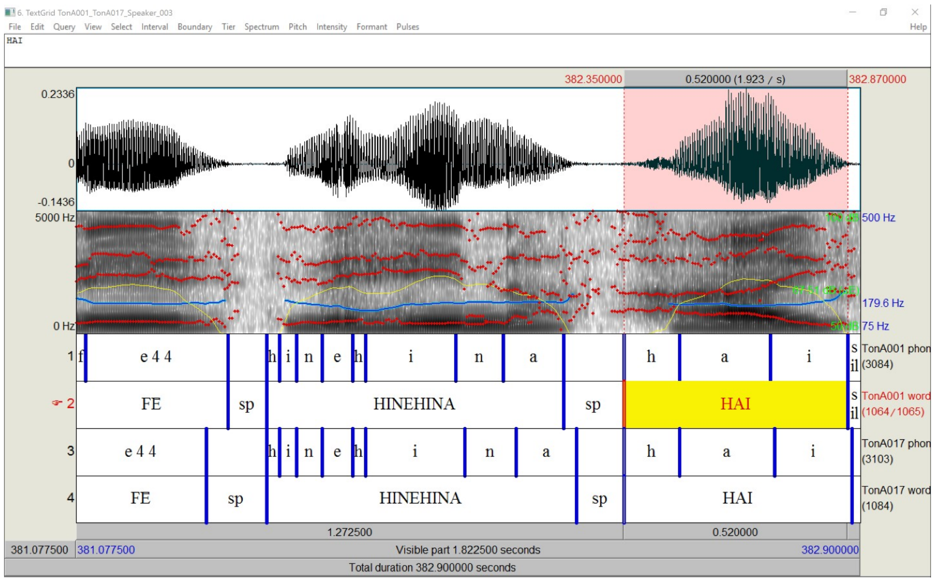

6. TextGrid TonA001_TonA017_Speaker_003
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
HAI
382.350000 0.520000 (1.923 / s) 382.870000
0.2336 0 -0.1436
5000 Hz 0 Hz 500 Hz 179.6 Hz 75 Hz
1 f e 4 4 h i n e h i n a h a i sil TonA001 phon (3084)
2 FE sp HINEHINA sp HAI sil TonA001 word (1064/1065)
3 e 4 4 h i n e h i n a h a i TonA017 phon (3103)
4 FE sp HINEHINA sp HAI TonA017 word (1084)
1.272500 0.520000
381.077500 381.077500 Visible part 1.822500 seconds 382.900000
Total duration 382.900000 seconds

Table 16. Differences in boundary placement for screenshot in Figure 15 – clean file alignment based on model that included alignment files in training data (TonA001) and one that did not (TonA017) – beginning of recording

Boundary	TonA001 Timestamp	TonA017 Timestamp	Difference
sil/k	0.03	0.03	0 ms
k/o	0.13	0.13	0 ms
o/t	0.29	0.29	0 ms
t/o	0.41	0.42	10 ms
o/a	0.67	0.68	10 ms
a/sp	0.91	0.92	10 ms
sp/p	1.04	1.04	0 ms
p/e	1.10	1.09	10 ms
e/a	1.34	1.34	0 ms
a/sp	1.53	1.53	0 ms
sp/m	1.67	1.68	10 ms
m/a	1.79	1.82	30 ms
a/n	1.94	1.96	20 ms
n/u	2.10	2.12	20 ms
u/sp	2.29	2.29	0 ms
sp/e	2.43	2.45	20 ms
e/f	2.53	2.53	0 ms
f/u	2.72	2.71	10 ms
mean			8 ms
median			10 ms

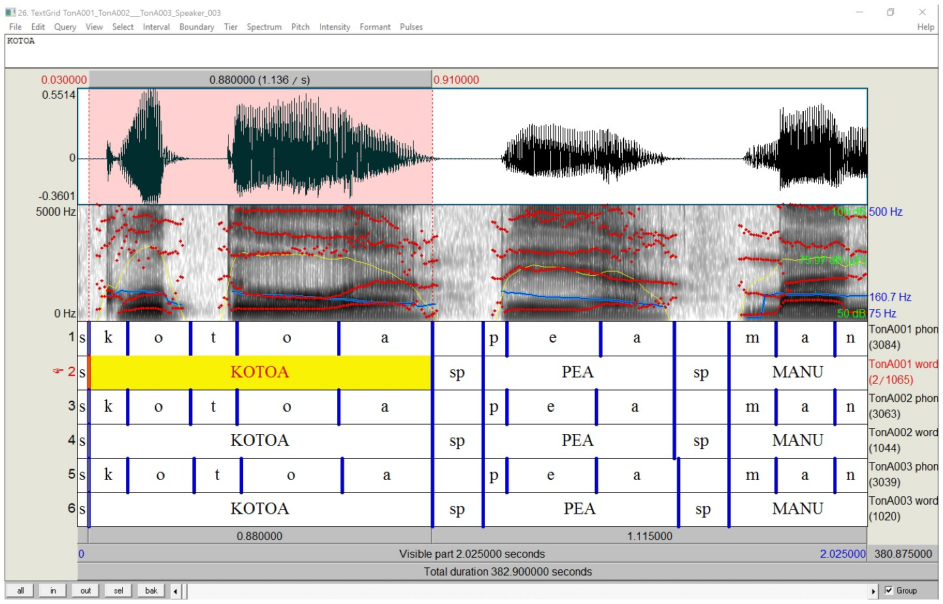
Table 17. Differences in boundary placement for screenshot in Figure 16 – clean file alignment based on model that included alignment files in training data (TonA001) and one that did not (TonA017) – end of recording

Boundary	TonA001 Timestamp	TonA017 Timestamp	Difference
e44/sp	381.43	381.38	50 ms
sp/h	381.52	381.52	0 ms
h/i	381.55	381.55	0 ms
i/n	381.59	381.59	0 ms
n/e	381.65	381.65	0 ms
e/h	381.72	381.72	0 ms
h/i	381.75	381.75	0 ms
i/n	381.96	381.98	20 ms
n/a	382.07	382.10	30 ms
a/sp	382.21	382.24	30 ms
sp/h	382.35	382.35	0 ms
h/a	382.48	382.48	0 ms
a/i	382.69	382.70	10 ms
i/sil	382.87	382.88	10 ms
mean			11 ms
median			0 ms

Whether the acoustic model was trained on the same files as those used in the alignment had little effect on the alignment in this example, as evidenced by the 0 ms median difference in Table 17. From this snapshot we cannot determine which alignment is better, and the two alignments remain similar from the beginning to the end of the recoding. The overall difference between the two models is quantitatively assessed in §4.2. We conducted a similar test using dirty files. However, given the poor general performance of dirty models (§4.1.1), those results are not discussed here. They are summarized with the other quantitative comparisons in §4.2.

4.1.4 Number of epochs As explained in §3.3.1, model training is accomplished in cycles, with a set number of iterations, or *epochs*, in each cycle. The default number of epochs in each cycle is five, but PL-A gives the user the opportunity to adjust that number. Since training is accomplished in three cycles, increasing the number of epochs by one increases the total number of iterations by three. We wanted to know whether increasing the number of epochs improved the quality of the alignments. The next comparisons were made to answer that question. We trained acoustic models using the same clean training files at three different epoch settings: 5, 10, and 15. We then used those models to align transcriptions to the clean recordings, as summarized in Table 18.

Figure 17. Clean file alignment based on models trained in 5 epochs (*top*, TonA001). 10 epochs (*middle*, TonA002) and 15 epochs (*bottom*, Ton003)


26. TextGrid TonA001_TonA002__TonA003_Speaker_003
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses Help
KOTOA
0.030000
0.880000 (1.136 / s)
0.910000
0.5514
0
-0.3601
5000 Hz
0 Hz
500 Hz
160.7 Hz
75 Hz
1 s k o t o a p e a m a n TonA001 phon (3084)
2 s KOTOA sp PEA sp MANU TonA001 word (2/1065)
3 s k o t o a p e a m a n TonA002 phon (3063)
4 s KOTOA sp PEA sp MANU TonA002 word (1044)
5 s k o t o a p e a m a n TonA003 phon (3039)
6 s KOTOA sp PEA sp MANU TonA003 word (1020)
0.880000
1.115000
0
Visible part 2.025000 seconds
2.025000
380.875000
Total duration 382.900000 seconds
all in out sel bak
Group

The screenshot in Figure 17 shows that changing the number of epochs in the training cycle had little effect on the alignments in this example. The summary of boundary differences in Table 19 shows only very small differences in boundary placement, with a mean difference of 4 ms and a median difference of 0 ms. As with the previous example, it is difficult to guess from this snapshot which alignment is better, but the overall size of the effect will be demonstrated in the next section.

Table 18. Epoch setting comparison

Clean-Clean 5 epochs	Clean-Clean 10 epochs	Clean-Clean 15 epochs
TonA001 (Aligned Speakers 001-023*)	TonA002 (Aligned Speakers 001-023*)	TonA003 (Aligned Speakers 001-023*)

*There are no recordings for Speaker 008.

Table 19. Differences in boundary placement for screenshot in Figure 17 – clean file alignment based on models trained in 5 epochs (TonA001), 10 epochs (TonA002) and 15 epochs (TonA003)

Boundary	TonA001 Timestamp	TonA002 Timestamp	TonA003 Timestamp	Difference
sil/k	0.03	0.03	0.03	0 ms
k/o	0.13	0.13	0.13	0 ms
o/t	0.29	0.29	0.30	10 ms
t/o	0.41	0.41	0.42	10 ms
o/a	0.67	0.67	0.68	10 ms
a/sp	0.91	0.91	0.91	0 ms
sp/p	1.04	1.04	1.04	0 ms
p/e	1.10	1.10	1.10	0 ms
e/a	1.34	1.33	1.33	10 ms
a/sp	1.53	1.53	1.54	10 ms
sp/m	1.67	1.67	1.67	0 ms
m/a	1.79	1.79	1.79	0 ms
a/n	1.94	1.94	1.94	0 ms
mean				4 ms
median				0 ms

4.2 Quantitative comparisons: Euclidean distance measures The analysis of the screenshots provided in the previous sections provides an evaluation of the PL-A boundaries based on a sample of tokens from several speakers. Because we plan to use the corpus for automated sociophonetic analysis, we have also compared the effects on extracted vowel formant measurements, a commonly analyzed variable, from TextGrids created in different alignment tests across full recordings from all speakers.[38] For each comparison described in §4.1, we used PraatR (Albin 2014) to extract F1 and F2 measurements for all tokens of phonemically short vowels from 10% to 90% into the vowel's trajectory, sampled at 1% intervals.[39] After normalizing using a Lobanov transformation (Lobanov 1971), we used R to calculate the Euclidean

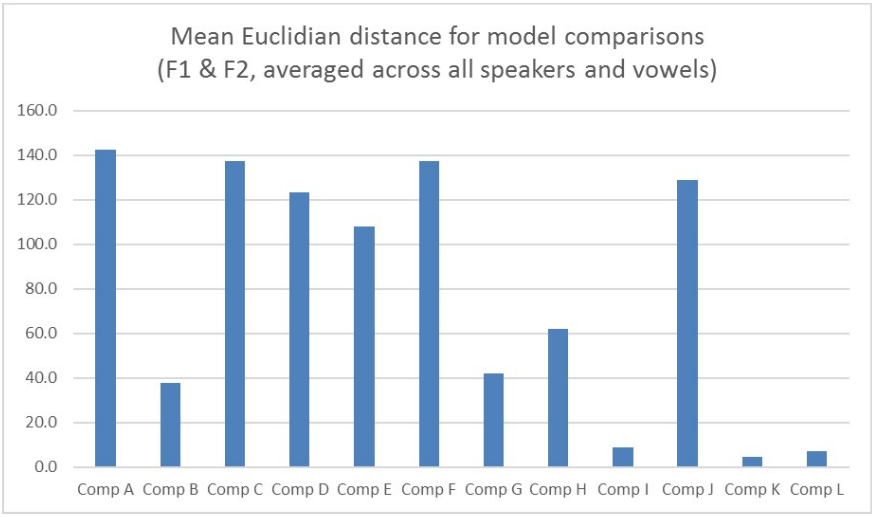
[38]Here we follow the lead set by Reddy & Stanford (2015), who used Euclidean distance between vowel formants to compare the results of two automated alignment and extraction tools. They presented their results in the form of vowel plots and mean distance in Hz.

[39]The default settings extracted measurements for vowels with a duration longer than .001 seconds. For the purposes of the present study, we assume that the Tongan phonemic inventory has five short vowels (i, e, a,

distance between formant measurements and then averaged across all speakers. The average distance of all vowels was compared, as well as the average distance for each of the five vowels. Table 20 presents the full list of the comparisons we made in our tests and the distance measures associated with them. A higher number indicates a greater difference. (Standard deviations are included in parentheses.)

The relative difference in distance measures for Comparisons A-L are illustrated in Figure 18. For simplicity, the graph shows only the distance measures averaged across all vowels.

Figure 18. Graph of distance measures in Table 20 (all vowels)



The distance measures listed in Table 20 and represented in Figure 18 do not tell us which alignments are better. They only indicate how different the measurements extracted from the two alignments in each comparison are. However, they confirm the observations made in §4.1 and lead us to the following conclusions based on our dataset:

- Clean vs. dirty (§4.1.1)
  - Our discussion and examples show that models trained on clean data create better alignments of both clean and dirty files. Comparisons A and D confirm that this distinction has a large effect across all the recordings.
  - We observed that including dirty files in the training data negatively affected both clean and dirty file alignment. Comparisons B and E confirm that this effect is much greater for dirty file alignment than for clean file alignment.

o, u) and five corresponding but contrasting long vowels. This distinction is represented in the orthography, with long vowels being written with a macron. We acknowledge that some researchers have argued that the long vowels are better described as a sequence of two vowels (Taumoefolau 2002; Anderson & Otsuka 2006), but we believe that the different phonological analysis would not lead to different hypotheses for our study.

  - Comparisons C and F are less meaningful because they indicate great differences between two bad alignments, one based on a dirty model and one based on a mixed clean and dirty one.

- TARGETRATE (§4.1.2): We observed that changing the TARGETRATE setting from 100000 to 125000 had a negative effect over time, though its effects were not obvious at the beginning of the example alignment. Comparison G confirms that TARGETRATE effect is smaller than the clean/dirty effect across all files in the comparison.

- Same vs. different (§4.1.3): We noticed very little difference in alignments based on whether the files to be aligned were included in the training dataset when both the training and alignment files were clean. Comparison I confirms that this effect was small across all clean-clean alignments. Though we can't definitively say which alignment was better, we see no reason to exclude files to be aligned when performing model training. (Comparisons H and J show much larger effects, but both of these compared two bad, dirty-model alignments.)

- Number of epochs (§4.1.4): In our observation, changing the epoch setting created very small changes in alignments of clean files using clean models. Comparisons K and L confirm that the effect of this change is very small across all files in the comparison. We did not test the effect of epoch setting on aligning dirty files.

5. Comparison to human aligners In order to compare the difference between PL-A's results and what a human aligner might do, we took a 60-word sample from the corpus and made hand corrections to it. The sample consisted of the first six words for the first five speakers, taken from both the clean and dirty versions of their recordings. The total number of boundaries in the sample was 534. The PL-A alignments used in the test were aligned based on a model trained on clean files with default settings for TARGETRATE (100000) and epochs (5).

After discussing boundary placement criteria and testing those criteria together on a separate "practice" set of tokens, two researchers independently made corrections to the 60-word sample and tracked how much time they spent making those corrections. We then compared the actual differences in where each researcher and PL-A placed the boundaries, measured in milliseconds. Figure 19 presents the results of these comparisons. In the figure, "Word Start" and "Word End" refer to boundaries placed at word edges. "Phone Start" and "Phone End" refer to boundaries for all phones, including those at word edges. Therefore, the "Average Phone" measurement refers to the mean "Phone Start" and "Phone End."

We see the greatest variance in word boundaries, as shown by the first two bars in each section. This is not surprising, since word-final (and even some word-initial) vowels may be partially devoiced, making the boundary between vowel and silence somewhat difficult to determine. In some cases, prevoicing on initial consonants

Table 20. Quantitative comparisons between alignments

Comparison	Candidate 1	Candidate 2	Description of Comparison	Distance Measures with Standard Deviations in Parentheses (Difference in F1 and F2 measurements, averaged across speakers. Higher number means greater difference.)					
				All vowels	a	e	i	o	u
Comp A	TonA001 Trained on CLEAN-Aligning CLEAN	TonA005 Trained on DIRTY-Aligning CLEAN	Same clean files aligned on a clean model and aligned on a dirty model	142.5 (50.3)	140.6 (51.1)	158.2 (29.6)	122.7 (47.1)	107.8 (45.2)	183.5 (41.2)
Comp B	TonA001 T.CLEAN-A.CLEAN	TonA006 T.BOTH-A.CLEAN	Same clean files aligned on a clean model and a clean & dirty model	37.6 (22.2)	32 (18.0)	35 (21.8)	32.1 (17.9)	34.5 (18.4)	54.6 (26.9)
Comp C	TonA005 T.DIRTY-A.CLEAN	TonA006 T.BOTH-A.CLEAN	Same clean files aligned on a dirty model and a clean & dirty model	137.2 (51.3)	140.2 (50.9)	142.2 (30.0)	120.3 (48.4)	101.8 (49.6)	181.3 (41.0)
Comp D	TonA004 T.CLEAN-A.DIRTY	TonA005 T.DIRTY-A.DIRTY	Same dirty files aligned on a clean model and a dirty model	123.3 (39.0)	127.5 (15.4)	151.8 (15.0)	98.6 (18.2)	75.9 (21.5)	162.8 (34.1)
Comp E	TonA004 T.CLEAN-A.DIRTY	TonA006 T.BOTH-A.DIRTY	Same dirty files aligned on a clean model and a clean & dirty model	107.8 (45.4)	145.2 (27.2)	153.8 (37.1)	78.9 (27.4)	68.2 (15.9)	92.9 (34.6)
Comp F	TonA005 T.DIRTY-A.DIRTY	TonA006 T.BOTH-A.DIRTY	Same dirty files aligned on a dirty model and a clean & dirty model	137.2 (51.3)	140.2 (50.9)	142.2 (30.0)	120.3 (48.4)	101.8 (49.6)	181.3 (41.0)
Comp G	TonA001 T.CLEAN-A.CLEAN-100K	TonA014 T.CLEAN-A.CLEAN-125K	Same clean files aligned on a clean model, differing only by target rate	42.1 (23.5)	25.5 (7.3)	32.5 (14.8)	36.3 (19.3)	54.4 (20.8)	61.6 (28.3)
Comp H	TonA005 T.DIRTY-A.DIRTY-SAME	TonA018 T.DIRTY-A.DIRTY-DIFFERENT	Same dirty files, aligned on one dirty model that included those files in training and another dirty model that did not	62.1 (31.7)	117 (18.9)	42.3 (8.2)	62.1 (12.1)	35.4 (9.0)	53.9 (12.0)
Comp I	TonA001 T.CLEAN-A.CLEAN-SAME	TonA017 T.CLEAN-A.CLEAN-DIFFERENT	Same clean files, aligned on one clean model that included those files in training and another clean model that did not	8.8 (5.1)	14 (5.2)	5.1 (2.0)	6.9 (3.9)	9.4 (5.2)	8.8 (5.0)

Continued from previous page

Comparison	Candidate 1	Candidate 2	Description of Comparison	Distance Measures with Standard Deviations in Parentheses (Difference in F1 and F2 measurements, averaged across speakers. Higher number means greater difference.)					
				All vowels	a	e	i	o	u
Comp J	TonA012 T.DIRTY-A.DIRTY-DIFFERENT	TonA017 T.CLEAN-A.CLEAN-DIFFERENT	Same speakers, aligning both a clean version and a dirty version. The dirty version is aligned using a model trained on different dirty files, and the clean version is aligned using a model trained on different clean files	128.9 (33.2)	136 (15.1)	150.3 (28.5)	107.4 (35.9)	105.6 (27.9)	145.2 (36.1)
Comp K	TonA001 T.CLEAN-A.CLEAN-5E	TonA002 T.CLEAN-A.CLEAN-10E	Same clean files aligned one clean model set at 5 epochs and one set at 10 epochs.	4.5 (3.1)	3.1 (1.1)	2.8 (1.7)	6.9 (3.6)	3.6 (2.0)	5.9 (3.7)
Comp L	TonA001 T.CLEAN-A.CLEAN-5E	TonA003 T.CLEAN-A.CLEAN-15E	Same clean files aligned one clean model set at 5 epochs and one set at 15 epochs.	7.1 (3.6)	3.6 (1.4)	10 (2.6)	8.1 (3.6)	6.3 (2.6)	7.4 (4.2)

made placing those boundaries less straightforward than expected as well.[40] Even with these complications, the average differences for each of the three comparisons, as shown by the last bar in each section, are similar at 19.23 ms, 18.18 ms, and 17.15 ms.

Table 21 lists rates of agreement between PL-A boundaries and those placed by the human aligners. Rates are shown for all boundaries and for word-internal boundaries. The higher agreement rates for word-internal boundaries arise from the large variance in word-boundary placement.

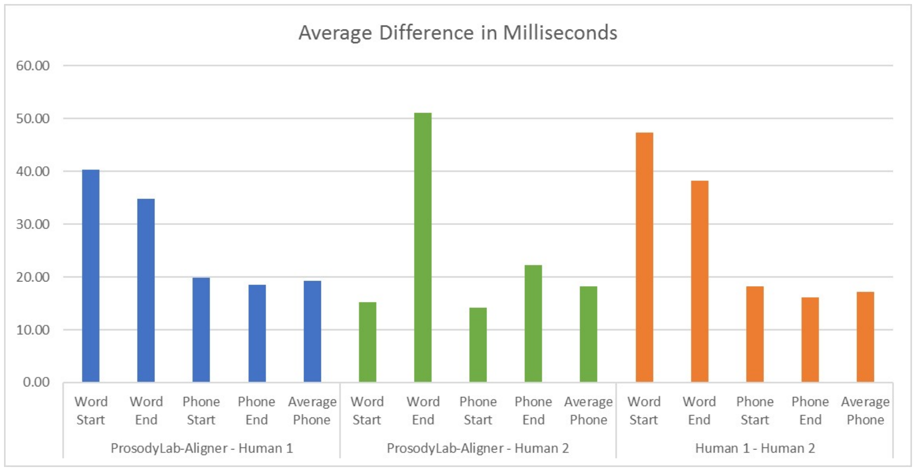
Although this sample is small, the results clearly support our assertion that trained forced alignment is a feasible time-saving option. First, they indicate that the agreement between PL-A and a manual alignment would not be substantially different from the agreement between two manual alignments. Second, they compare favorably to the results of cross-language forced alignment summarized in Tables 1 and 2 in §2.2. Even with a small amount of training data, long audio files, and unconstrained alignment, the language-specific model seems to perform well when trained on clean data.

[40]DiCanio et al. (2013) also found that agreement at word edges was much lower than at non-edges, though the causes may have been different.

Table 21. Rates of agreement between PL-A alignment and manual adjustments (60-word sample)

	All Boundaries			**Word-Internal Boundaries**		
	PLA - Human 1	PLA - Human 2	Human 1 - Human 2	PLA - Human 1	PLA - Human 2	Human 1 - Human 2
% Agreement <20 ms	63%	71%	74%	73%	75%	85%
% Agreement <30 ms	75%	79%	81%	85%	84%	92%
% Agreement <40 ms	84%	87%	88%	93%	93%	97%
% Agreement <50 ms	92%	90%	93%	95%	95%	98%

Figure 19. Differences between PL-A alignment and manual adjustments (60-word sample)



Finally, after performing this test, we believe that making manual corrections to forced alignments would require far less time than manually aligning the files from scratch. One researcher spent an average of 1 minute per phone when making adjustments, while the other researcher averaged about 2 minutes per phone. For both, the speed increased as they became more familiar with the data. In our experience, this was much faster than aligning transcriptions to recordings by hand, especially with little information about phonetic detail and possible variation. In addition, making corrections to a consistent forced alignment allows the researcher to devote more time and attention to problematic areas rather than to the simple boundary placements that the aligner successfully handles. The focused attention on problematic boundaries has also revealed interesting features of the language that may warrant further study.

6. Conclusions and Recommendations In general, we find Prosodylab-Aligner to be a useful tool in processing recordings for phonetic analysis, despite the lack of a user-friendly interface and some errors we encountered along the way. As a first step, it provides a foundation upon which language-specific trained forced alignment technology can build. Based on our tests, we make the following recommendations regarding the use of Prosodylab-Aligner with long field recordings containing background noise:

- Cleaning at least one hour of the recordings and using only clean files for model training
- Using the default TARGETRATE setting (100000)
- Making manual adjustments to boundaries before taking acoustic measurements

At this stage, it's unclear what level of cleaning is required to produce the positive effects we have seen in model training with PL-A. The clean recordings used in this study were heavily edited to remove all background noise. Further study may reveal that removing only the worst problems provides the same or similar benefits at a lower cost.

There may be ways of producing even better results than those described in this paper. For example, increasing the amount of clean training data may improve model performance. Providing the aligner with different kinds of data could also make a difference. Since our tests used word list recordings, the data were limited to a small set of words in citation form. For each of these words, our dictionary file contained only one pronunciation, based solely on orthography. Including connected speech, increasing the number of words, and providing alternate pronunciations could provide the aligner with more information about phonemes in various contexts. Finally, since Prosodylab-Aligner was developed and tested using short audio files containing single utterances, we expect that using short files would produce better results with fewer cascading errors than what we saw when dirty files were used in model training. In fact, breaking up long audio files into individual utterances could be one lower-cost way of cleaning dirty files and constraining alignment.[41]

Beyond these specific observations related to Prosodylab-Aligner, our tests show that forced alignment with acoustic models trained specifically for underdocumented languages is feasible and can be accomplished by linguists without programming experience. Our results suggest that as little as an hour of clean recordings can create an acoustic model that produces reasonably good alignments, even when aligning transcriptions to recordings with background noise. Under such circumstances, cross-language forced alignment may not be necessary, and using a language-specific model offers the opportunity to refine the model as more recordings become available and as researchers learn more about pronunciation and variation.

Based on our tests, we conclude that forced alignment can greatly increase efficiency in preparing sound files for phonetic analysis even for underdocumented languages. Runtimes are remarkably short, and the time required to clean training files and make manual corrections is substantially shorter than aligning transcriptions totally by hand.

References

Albin, Aaron L. 2014. PraatR: An architecture for controlling the phonetics software "Praat" with the R programming language. *The Journal of the Acoustical Society of America* 135(4). 2198–2199.

Anderson, Victoria & Yuko Otsuka. 2006. The phonetics and phonology of "definitive accent" in Tongan. *Oceanic Linguistics* 45(1). 21–42.

Bale, Alan, Jessica Coon, Joel Dunham, Kyle Gorman & Michael Wagner. 2015. LingSync and ProsodyLab-Aligner: Tools for linguistic fieldwork and experimenta-

[41]One reviewer recommends ELAN2split for chopping up large audio files transcribed in ELAN. This tool was developed by Damir Cavar especially for use with Prosodylab-Aligner. Links can be found at https://-gorilla.linguistlist.org/software/#ELAN2split.

tion. Paper presented at the Linguistic Society of America Annual Meeting, Portland, January, 8, 2015.

Boersma, Paul & David Weenink. 2015. Praat: Doing phonetics by computer [Computer program]. (5.4.22). http://www.fon.hum.uva.nl/praat/. Retrieved 10/27/2015.

Cambridge University. 1989–2015. HTK Hidden Markov Model Toolkit.

Carnegie Mellon University. 1993–2016. *CMU pronouncing dictionary*. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Accessed March 18, 2017.

Churchward, Clerk Maxwell. 1959. *Tongan dictionary: Tongan-English and English-Tongan*. Oxford: Oxford University Press.

DiCanio, Christian, Hosung Nam, Douglas H. Whalen, H. Timothy Bunnell, Jonathan D. Amith & Rey Castillo García. 2013. Using automatic alignment to analyze endangered language data: Testing the viability of untrained alignment. *The Journal of the Acoustical Society of America* 134(3). 2235–2246.

Evanini, Keelan, Stephen Isard & Mark Liberman. 2009. Automatic formant extraction for sociolinguistic analysis of large corpora. *Interspeech 2009. Proceedings of the 10th Annual Conference of the International Speech Communication Association*, Brighton, United Kingdom, September 6–10, 2009. 1655–1658.

Goldman, Jean-Philippe. 2011. EasyAlign: An automatic phonetic alignment tool under Praat. *Interspeech 2011. Proceedings of the 12th Annual Conference of the International Speech Communication Association*, Florence, Italy, August 28–31, 2011. 3233–3236.

Gorman, Kyle, Jonathan Howell & Michael Wagner. 2011. Prosodylab-Aligner: A tool for forced alignment of laboratory speech. *Canadian Acoustics* 39(3). 192–193.

Kempton, Timothy. 2017. Cross-language forced alignment to assist community-based linguistics for low resource languages. Paper presented at ComputEL-2, Honolulu, March 6–7, 2017.

Kempton, Timothy, Roger K. Moore & Thomas Hain. 2011. Cross-language phone recognition when the target language phoneme inventory is not known. *Interspeech 2011. Proceedings of the 12th Annual Conference of the International Speech Communication Association*, Florence, Italy, August 28–31, 2011. 3165–3168.

Kisler, Thomas, Florian Schiel & Han Sloetjes. 2012. Signal processing via web services: The use case WebMAUS. Paper presented at the Digital Humanities Conference 2012, Hamburg, Germany, July 16–22, 2012.

Kurtic, Emina, Bill Wells, Guy J. Brown, Timothy Kempton & Ahmet Aker. 2012. A corpus of spontaneous multi-party conversation in Bosnian Serbo-Croatian and British English. *Proceedings of the Eighth International Conference on Language Resources and Evaluation* (LREC'12), Istanbul, Turkey, May 23–25, 2012. 1323–1327.

Lobanov, Boris M. 1971. Classification of Russian vowels spoken by different speakers. *The Journal of the Acoustical Society of America* 49(2B). 606–608.

Max Planck Institute for Psycholinguistics, The Language Archive. ELAN Linguistic Annotator. (4.9.1). program]. https://tla.mpi.nl/tools/tla-tools/elan/.

McAuliffe, Michael, Michaela Socolof, Sarah Mihuc & Michael Wagner. 2016. Montreal Forced Aligner. (v0.7.1). https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner. Retrieved 28 September 2016.

McAuliffe, Michael, Michaela Socolof, Sarah Mihuc, Michael Wagner & Morgan Sonderegger. 2017a. Montreal Forced Aligner: An accurate and trainable aligner using Kaldi. Poster presented at the Linguistic Society of America 91st Annual Meeting, Austin, January 5–8, 2017.

McAuliffe, Michael, Michaela Socolof, Sarah Mihuc, Michael Wagner & Morgan Sonderegger. 2017b. Montreal Forced Aligner: trainable text-speech alignment using Kaldi. Interspeech 2017. Stockholm, Sweden.

Ochshorn, Robert M. & Max Hawkins. 2016. *Gentle: A robust yet lenient forced aligner built on Kaldi.* https://lowerquality.com/gentle/. Accessed April 27, 2017.

Povey, Daniel, Arnab Ghoshal, Giles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer & Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. Paper presented at IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, Waikoloa, December 11–15, 2011.

Reddy, Sravana & James Stanford. 2015. Toward completely automated vowel extraction: Introducing DARLA. *Linguistics Vanguard* 1(1). doi:10.1515/lingvan-2015-0002.

Rosenfelder, Ingrid. 2013. *Forced alignment & vowel extraction (FAVE): An online suite for automatic vowel analysis.* http://fave.ling.upenn.edu/index.html. Accessed November 26. 2015.

Rosenfelder, Ingrid, Joe Fruehwald, Keelan Evanini & Jiahong Yuan,. 2011. FAVE (Forced alignment and vowel extraction) program suite. http://fave.ling.upenn.edu/. Accessed January 16, 2018.

Schiel, Florian, Christoph Draxler, Angela Baumann, Tania Ellbogen & Alexander Steffen. 2012. The production of speech corpora, version 2.5. https://pdfs.semanticscholar.org/4f18/fbdfa480607d967bb731a111c4d30dcb140e.pdf. Accessed January 16, 2018.

Shumway, Eric B. 2009. *Intensive course in Tongan: With numerous supplementary materials, grammatical notes, and glossary, rev. ed.* Lāʻie, Hawaiʻi: The Jonathan Napela Center for Hawaiian and Pacific Islands Studies, Brigham Young University-Hawaii.

Sikveland, A., Anton Öttl, Ingunn Amdal, Mirjam Ernestus, Torbjørn Svendsen & Jens Edlund. 2010. Spontal-N: A corpus of interactional spoken Norwegian. *Proceedings of the Seventh International Conference on Language Resources and Evaluation* (LREC'10), Valletta, Malta, May 17–23, 2010. 2986–2991.

SIL International. 2015. *ISO 639-3 code tables.* http://www-01.sil.org/iso639-3/codes.asp. Accessed January 16, 2018.

Strunk, Jan, Florian Schiel & Frank Seifart. 2014. Untrained forced alignment of transcriptions and audio for language documentation corpora using WebMAUS. *Proceedings of the Ninth International Conference on Language Resources and Evaluation* (LREC'14), Reykjavik, Iceland, May 26–31, 2014. 3940–3947.

Taumoefolau, Melenaite. 2002. Stress in Tongan. *MIT Working Papers in Linguistics* 44. 341–353.

The Rosetta Project and the Long Now Foundation. 2010. *Tongan Swadesh List*. https://archive.org/details/rosettaproject_ton_swadesh-1. Accessed April 2014.

Young, Steve, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Garth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev & Phil Woodland. 1995–2006. (for HTK Version 3.4).

Yuan, Jiahong & Mark Liberman. 2008. Speaker identification on the SCOTUS corpus. *Journal of the Acoustical Society of America* 123(5). 3878.

Lisa M. Johnson
lisa.johnson@anthro.utah.edu
orcid.org/orcid.org/0000-0002-6817-7713

Marianna Di Paolo
dipaolo@anthro.utah.edu

Adrian Bell
adrian.bell@anthro.utah.edu
orcid.org/0000-0002-4670-384X