FORCED ALIGNMENT FOR UNDERSTUDIED LANGUAGE VARIETIES

Lisa M. Johnson, Marianna Di Paolo, Adrian Bell, Carter Holt
University of Utah
ICLDC5 | March 2, 2017

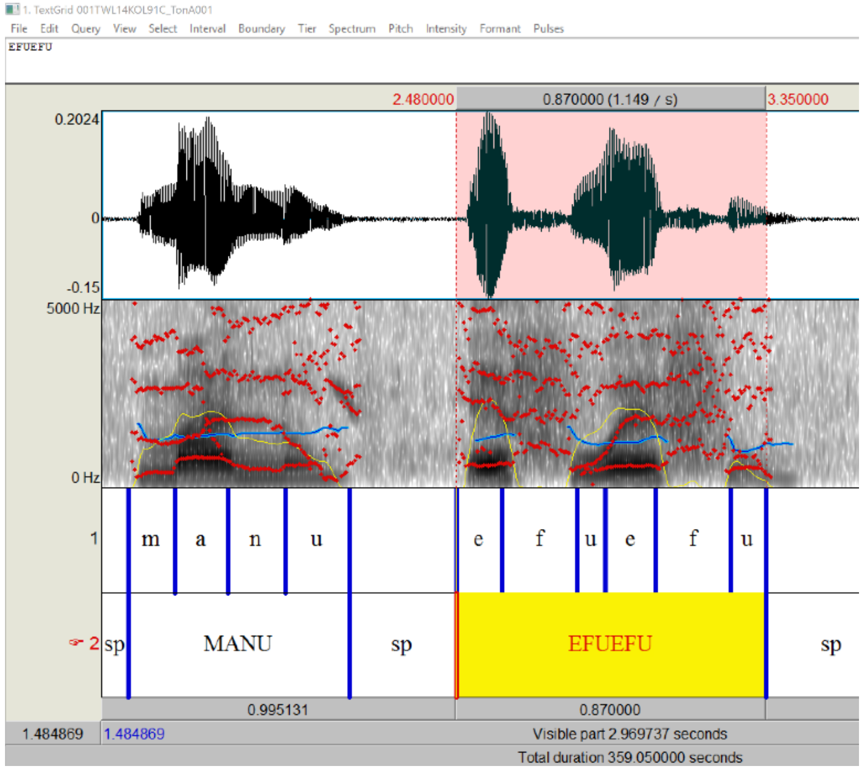
THE PROBLEM

It's necessary in both sociolinguistics, esp. **sociophonetics**, and **language documentation** to efficiently process **large corpora of recorded speech.**

Processing recordings for acoustic analysis is **very time consuming.**

By some estimates, **manual phone-level alignment** may take up to 800x the duration of the audio!

Today we will focus on how these similar processing problems may share similar solutions.

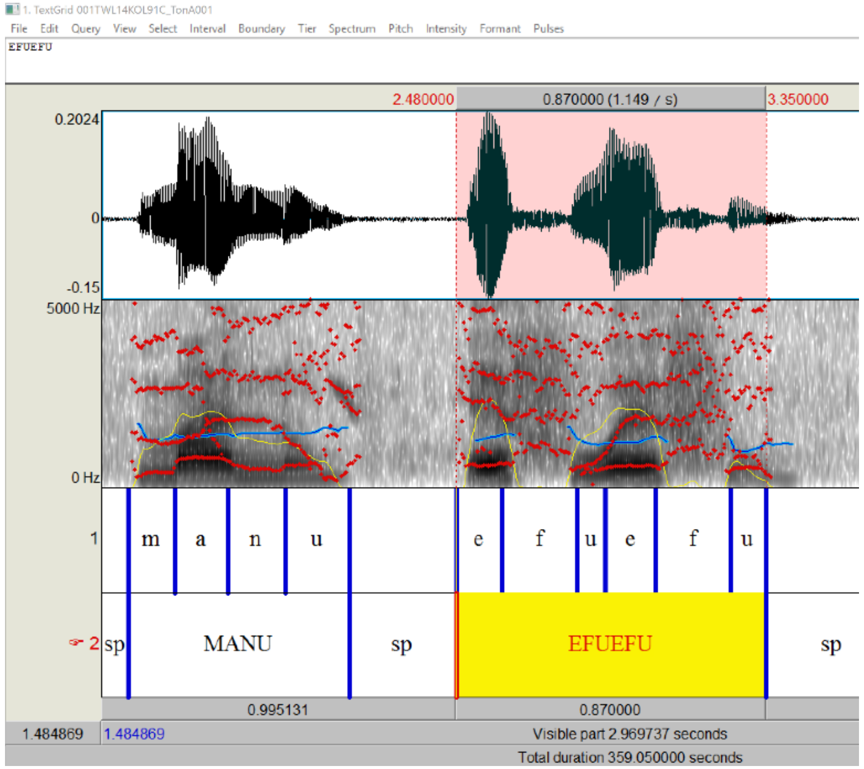


THE PROBLEM

The time and cost associated with processing audio recordings can **limit** the **amount and kinds of data analyzed** and even **the kinds of questions explored.**

Such limitiations are especially problematic when they **inhibit work on underdocumented languages.**

Is there a way to expedite this process?



FORCED ALIGNMENT TOOLS

In recent years, new tools have been developed to time-align orthographic transcriptions to recorded speech at the word and phone level.

These forced alignment tools use speech recognition technology to **create a statistical model associating phonetic symbols to speech signals.**

Sociophonetics has benefited greatly from the use of forced alignment technology

- Developed primarily for majority languages like English, with large extant corpora available

Examples: Forced Alignment and Vowel Extraction (FAVE) (Rosenfelder 2013, Rosenfelder et al. 2011); EasyAlign (Goldman 2011); MAUS/WebMAUS (Kisler, Schiel, and Sloetjes 2012); Prosodylab-Aligner (Gorman, Howell, and Wagner 2011); and the Dartmouth Linguistic Automation suite (DARLA) (Reddy and Stanford 2015)

FORCED ALIGNMENT

Two digital tools developed for forced alignment of underdocumented languages:

- Prosodylab-Aligner (PL-A)
- Montreal Forced Aligner (MFA)

(Both developed at McGill University Prosody Lab)

Key features:

- Don’t require a pretrained model or a large corpus
- Allow model training and alignment using the same dataset

TONGAN ETHNOLINGUISTIC STUDY

Large-scale ethnographic and linguistic study of post-migration Tongans/Tongan Americans in the U.S. (Adrian Bell, PI)

- Formation of new post-migration ethnolinguistic identities
- Longitudinal and cross-sectional data
- Includes data collection by crowd-sourcing
  - Leads to huge linguistic data set
  - Must expedite the linguistic analysis

To identify potentially important linguistic variables in these newly formed U.S. Tongan American communities, exploring

- **linguistic variation in Tonga**
  - potential Tongan sociolinguistic variables
  - varieties of English used in Tonga
- **linguistic variation** the U.S. **English** contact varieties **in Salt Lake Valley**

Best to use same digital tools for both Tongan & ambient English

PROSODYLAB-ALIGNER

For Training and Alignment
(Understudied Languages)

WHAT IS PROSODYLAB-ALIGNER?

- A set of scripts that use HTK (Hidden Markov Toolkit) speech recognition software to create time-aligned TextGrid transcriptions
- Designed with laboratory data in mind, best with short audio files
- Includes a pre-trained North American English model
- Supports model training on user-supplied data
- Does not require and time-aligned training data (uses simple text transcriptions)
- Has been used for a variety of majority and minority languages

  English (U.S.A., Canadian, British, Aviation, South African), French, Arabic (Gulf), Irish, Cantonese, German, Polish, Mandarin, Tagalog, Spanish, Cho'ol, Mi'gmaq, and Kaqchikel. (Gorman, p.c.)

http://prosodylab.org/tools/

GETTING STARTED

Instructions, Issues, and Solutions

WHAT YOU NEED

Requirements/Recommendations

Hardware

- Instructions provided for Mac, Linux, can also be used with Windows

Software Downloads

- Prosodylab-Aligner—GitHub
- Xcode (compiler)—Mac App Store
- HTK (Hidden Markov Toolkit)—HTK website
- Homebrew
- Python
- SoX utilities

What we used

Hardware: Microsoft Surface Pro ¾

- Intel Core i5-4300U / 6300U
- 8GB LPDDR3 RAM
- 256 GB SSD (data files on 200GB micro SD)

Software

- Compiled HTK (x64) on Windows using nmake
- Installed Python environment and required packages

WHAT WE LEARNED

The software can be a bit tricky and buggy, but we got it to work.

The Aligner's developer, Kyle Gorman, was very accessible and helpful.

It's good to have one of these on hand

Craig Johnson, programmer

WHAT YOU NEED

Requirements

Audio files (.wav)

- default at 16 kHz (automatically resamples, but you can override)

Example

WHAT YOU NEED

Requirements

Audio files (.wav)

Transcription files (.lab)

- Plain text, UTF-8
- Prescribed format: All caps, single spaces between words, no carriage returns or punctuation, regular spelling conventions (with Unicode characters)

Example

```
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
```

WHAT YOU NEED

Requirements

Audio files (.wav)

Transcription files (.lab)

Configuration file (.yaml)

- Not mentioned in tutorials for older versions
- Contains settings and a “list of phones”
- English example included in download

Example

```yaml
# for human reading only
authors: Kyle Gorman
language: English
citation: "K. Gorman, J. Howell, and M. Wagner. 2011. Prosodylab-Aligner: A tool for forced alignment of laboratory speech. Canadian Acoustics, 39(3), 192-193."
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

# specs for estimation; change at your own risk
HERest:
    TARGETRATE: 100000.0
    TARGETKIND: MFCC_D_A_0
    WINDOWSIZE: 250000.0
    PREEMCOEF: 0.97
    USEHAMMING: T
    ENORMALIZE: T
    CEPLIFTER: 22
    NUMCHANS: 20
    NUMCEPS: 12

# specs for the decoder; change at your own risk
HVite:
    SFAC: 5
```

```yaml
# for human reading only
authors: Kyle Gorman
language: English
citation: "K. Gorman, J. Howell, and M. Wagner. 2011. Prosodylab-Aligner: A tool for forced alignment of laboratory speech. Canadian Acoustics, 39(3), 192-193."
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

# specs for estimation; change at your own risk
HERest:
    TARGETRATE: 100000.0
    TARGETKIND: MFCC_D_A_0
    WINDOWSIZE: 250000.0
    PREEMCOEF: 0.97
    USEHAMMING: T
    ENORMALIZE: T
    CEPLIFTER: 22
    NUMCHANS: 20
    NUMCEPS: 12

# specs for the decoder; change at your own risk
HVite:
    SFAC: 5
```

WHAT YOU NEED

Requirements

Audio files (.wav)

Transcription files (.lab)

Configuration file (.yaml)

Dictionary file

- Provides pronunciation
- Uses “phones” listed in .yaml file
- Follows prescribed format
- North American English example included in download (others available)

Example

```
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
```

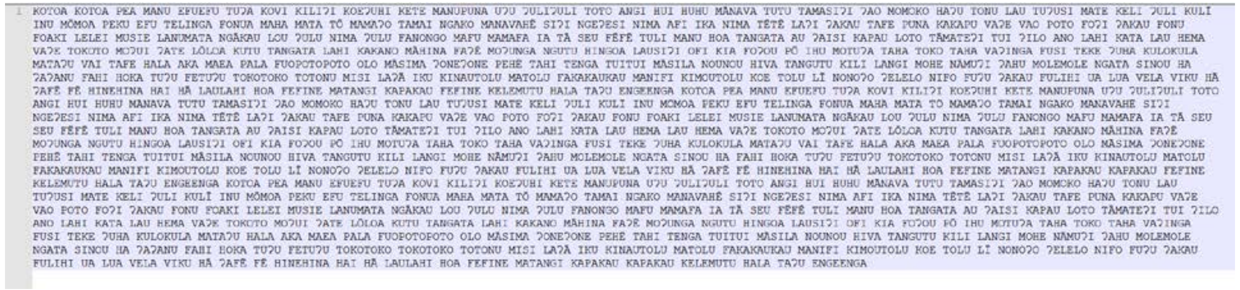
WHAT YOU NEED

What we used

Audio files (.wav)

- Word list readings
- Collected in the field
- Recorded with lavalier mics and Zoom H4n digital recorder
- 16 bit, 44.1 kHz (did not resample)
- Some files “cleaned” in Praat (22 files, 1:41:30), others left “dirty” with only extraneous speech removed in Audacity (16 files, 2:39:44 + 5 “dirty” versions of clean files 1:23:01)

WHAT YOU NEED



What we used

Audio files (.wav)

Transcription files (.lab)

- Originally created in Elan using controlled vocabulary
- Transcriptions of "clean" files: extracted non-empty intervals, and concatenated in Praat, then exported and formatted in Word and Notepad++
- Transcriptions of "dirty" files exported from Elan and prepared in Word and Notepad++.
- Used Tongan orthography (with ʔ instead of ‘)

KOTOA KOTOA PEA MANU EFUEFU TUɁA KOVI KILIɁI KOEɁUHI KETE MANUPUNA UɁU ɁULIɁULI TOTO ANGI HUI HUHU
INU MŌMOA PEKU EFU TELINGA FONUA MAHA MATA TŌ MAMAɁO TAMAI NGAKO MANAVAHĒ SIɁI NGEɁESI NIMA AFI IKA
FOAKI LELEI MUSIE LANUMATA NGĀKAU LOU ɁULU NIMA ɁULU FANONGO MAFU MAMAFA IA TĀ SEU FĒFĒ TULI MANU H
VAɁE TOKOTO MOɁUI ɁATE LŌLOA KUTU TANGATA LAHI KAKANO MĀHINA FAɁĒ MOɁUNGA NGUTU HINGOA LAUSIɁI OFI
MATAɁU VAI TAFE HALA AKA MAEA PALA FUOPOTOPOTO OLO MĀSIMA ɁONEɁONE PEHĒ TAHI TENGA TUITUI MĀSILA NOU
ɁAɁANU FAHI HOKA TUɁU FETUɁU TOKOTOKO TOTONU MISI LAɁĀ IKU KINAUTOLU MATOLU FAKAKAUKAU MANIFI KIMOU
ɁAFĒ FĒ HINEHINA HAI HĀ LAULAHI HOA FEFINE MATANGI KAPAKAU FEFINE KELEMUTU HALA TAɁU ENGEENGA KOTOA
ANGI HUI HUHU MĀNAVA TUTU TAMASIɁI ɁAO MOMOKO HAɁU TONU LAU TUɁUSI MATE KELI ɁULI KULĪ INU MŌMOA PE
NGEɁESI NIMA AFI IKA NIMA TĒTĒ LAɁI ɁAKAU TAFE PUNA KAKAPU VAɁE VAO POTO FOɁI ɁAKAU FONU FOAKI LELE
SEU FĒFĒ TULI MANU HOA TANGATA AU ɁAISI KAPAU LOTO TĀMATEɁI TUI ɁILO ANO LAHI KATA LAU HEMA LAU HEMA
MOɁUNGA NGUTU HINGOA LAUSIɁI OFI KIA FOɁOU PŌ IHU MOTUɁA TAHA TOKO TAHA VAɁINGA FUSI TEKE ɁUHA KULO
PEHĒ TAHI TENGA TUITUI MĀSILA NOUNOU HIVA TANGUTU KILI LANGI MOHE NĀMUɁI ɁAHU MOLEMOLE NGATA SINOU
FAKAKAUKAU MANIFI KIMOUTOLU KOE TOLU LĪ NONOɁO ɁELELO NIFO FUɁU ɁAKAU FULIHI UA LUA VELA VIKU HĀ ɁA
KELEMUTU HALA TAɁU ENGEENGA KOTOA PEA MANU EFUEFU TUɁA KOVI KILIɁI KOEɁUHI KETE MANUPUNA UɁU ɁULIɁU
TUɁUSI MATE KELI ɁULI KULĪ INU MŌMOA PEKU EFU TELINGA FONUA MAHA MATA TŌ MAMAɁO TAMAI NGAKO MANAVAH
VAO POTO FOɁI ɁAKAU FONU FOAKI LELEI MUSIE LANUMATA NGĀKAU LOU ɁULU NIMA ɁULU FANONGO MAFU MAMAFA I
ANO LAHI KATA LAU HEMA VAɁE TOKOTO MOɁUI ɁATE LŌLOA KUTU TANGATA LAHI KAKANO MĀHINA FAɁĒ MOɁUNGA NG
FUSI TEKE ɁUHA KULOKULA MATAɁU HALA AKA MAEA PALA FUOPOTOPOTO OLO MĀSIMA ɁONEɁONE PEHĒ TAHI TENGA T
NGATA SINOU HA ɁAɁANU FAHI HOKA TUɁU FETUɁU TOKOTOKO TOKOTOKO TOTONU MISI LAɁĀ IKU KINAUTOLU MATOLU
FULIHI UA LUA VELA VIKU HĀ ɁAFĒ FĒ HINEHINA HAI HĀ LAULAHI HOA FEFINE MATANGI KAPAKAU KAPAKAU KELEM

WHAT YOU NEED

```yaml
# 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

# 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

# specs for estimation; change at your own risk
HERest:
    TARGETRATE: 100000.0
    TARGETKIND: MFCC_D_A_0
    WINDOWSIZE: 250000.0
    PREEMCOEF: 0.97
    USEHAMMING: T
    ENORMALIZE: T
    CEPLIFTER: 22
    NUMCHANS: 20
    NUMCEPS: 12

# specs for the decoder; change at your own risk
HVite:
    SFAC: 5
```

What we used

Audio files (.wav)

Transcription files (.lab)

Configuration file (.yaml)

- “Phone list” uses 1 digraph (ng) and one Unicode IPA character (ʔ)
- Changed 1 setting (targetrate) to prevent crash.

```
# 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

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

WHAT YOU NEED

```
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
FULIHI f u l i h i
FUOPOTOPOTO f u o p o t o p o t o
FUSI f u s i
FUʔU f u ʔ u
HA h a
HAI h a i
HALA h a l a
HAʔU h a ʔ u
HĀ h ā
HEMA h e m a
HINEHINA h i n e h i n a
HINGOA h i ng o a
HIVA h i v a
HOA h o a
HOKA h o k a
HUHU h u h u
HUI h u i
IA i a
IHU i h u
```

What we used

Audio files (.wav)

Transcription files (.lab)

Configuration file (.yaml)

Dictionary file

- Created for this project from word list.
- Pronunciations based on orthography
- No alternate pronunciations included

```
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
FULIHI f u l i h i
FUOPOTOPOTO f u o p o t o p o t o
```

ISSUES AND SOLUTIONS

- Files must be saved as UTF-8 without “byte order mark” (BOM or “signature”)
- May need to check for extra spaces and carriage returns at the end of the text file
- Dictionary file must be sorted in Python’s sort order (script included)
- Apostrophes can be problematic
- May need to check for hidden .txt extensions

TONGAN TESTS

Training and Alignment

TRAINING

- Produces a acoustic model by which alignments can be created.
- Requires pairs of audio (.wav) files and transcription (.lab) files in the same folder.
- Is accomplished in three cycles, with a set number of iterations ("epochs") in each cycle.
- Is executed by entering a Python script (command line) into Terminal (Mac) or Command Prompt (PC).

Key elements of command line:

- -c lang.yaml (configuration file path)
- -d lang.dict (dictionary file path)
- -e 5 (number of epochs)
- -t lang/ (path of folder containing training data)
- -w lang-mod.zip (zip file to which model will be written)

ISSUES AND SOLUTIONS

- Problems can be difficult to diagnose and resolve.
- Problems with script syntax or file prep/organization cause process to fail.
- It can be hard to determine which files contain out-of-dictionary words.
- Some HTK error codes are not included in the PL-A docs or HTK Book. ("ERROR [+7390] StepAlpha: Alpha prune failed")

- Added a few lines of diagnostics to the code.
- Follow instructions carefully.
- Added code to include this information in the output. (We can make this available.)
- Had to Google the error to see how others had solved the problem. increased Targetrate setting in configuration file from 100000 (default) to 125000; feature measurements extracted every 12.5 ms rather than every10 ms.

TRAINING TESTS

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

ALIGNMENT

- Produces aligned TextGrids based on a previously created acoustic model.
- Requires pairs of audio (.wav) files and transcription (.lab) files in the same folder.
- Is executed by entering a Python script (command line) into Terminal (Mac) or Command Prompt (PC).

Key elements of command line:

- -r lang-mod.zip (‘read’: path to language model)
- -a data/ (‘align’: directory containing files to be aligned
- -d lang.dict (dictionary file path)

ISSUES AND SOLUTIONS

- Program produces no output to show progress through the process.
- Problems with script syntax or file prep/organization cause process to fail.
- Unicode characters display properly in word tier of output TextGrid but as number codes in phone tier.

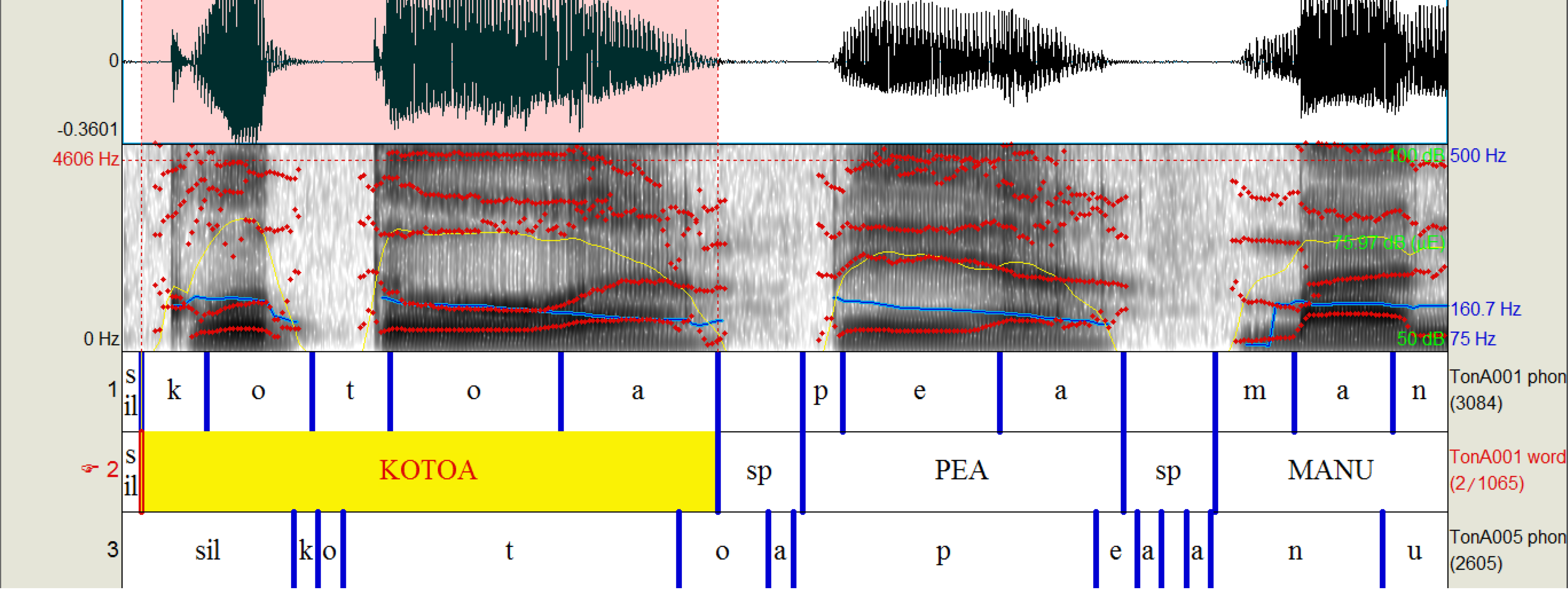
- Used Task Manager (processes tab) to monitor process.
- Follow instructions carefully
- Can search and replace in TextGrid, but the characters are unique consistent so the intended Unicode character is clear.

ALIGNMENT TESTS

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	0:10:57
TonA010	dirty (5)	ton-002-mod.zip (trained on clean)	10	100000	0:13:06
TonA011	dirty (5)	ton-003-mod.zip (trained on clean)	15	100000	0:13:38
TonA012	dirty (5)	ton-004-mod.zip (trained on dirty)	5	125000	0:14:25
TonA013	dirty (5)	ton-005-mod.zip (trained on clean & dirty)	5	125000	0:16:46
TonA014	clean & dirty (43)	ton-006-mod.zip (trained on clean)	5	125000	0:12:02
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

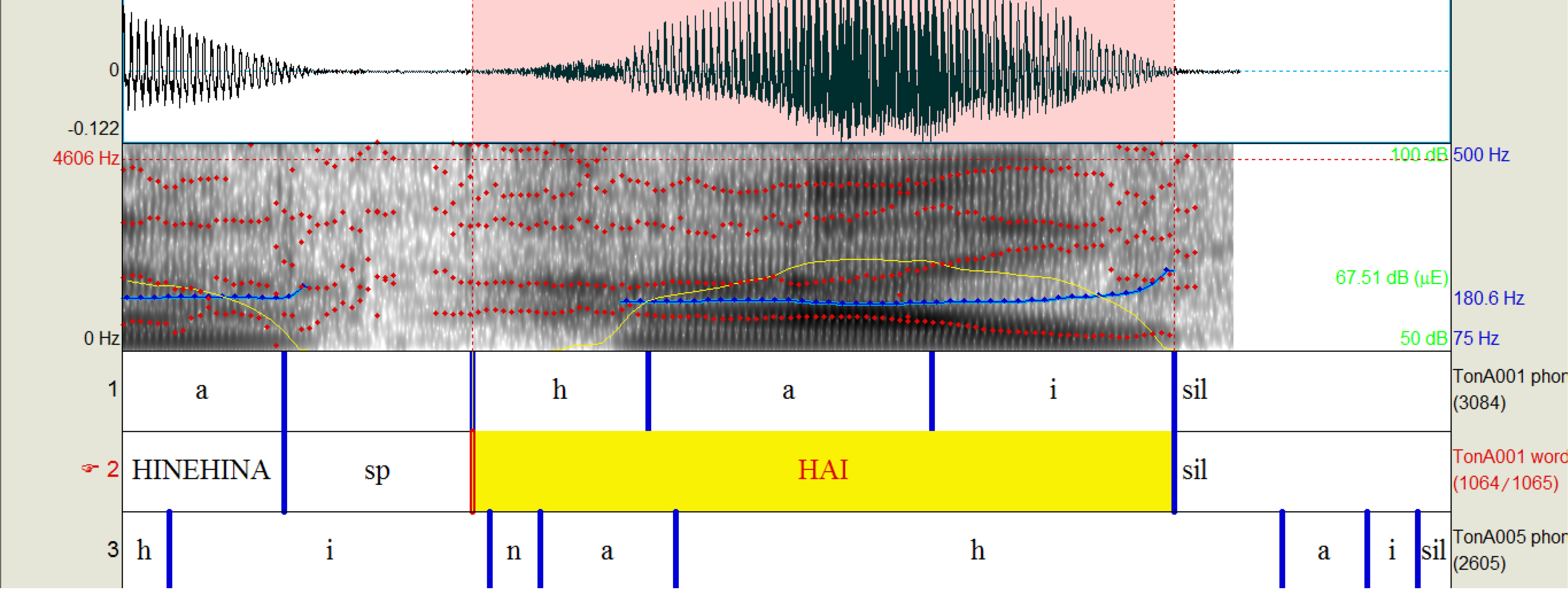
ALIGNMENT COMPARISONS

Reliability and Validity



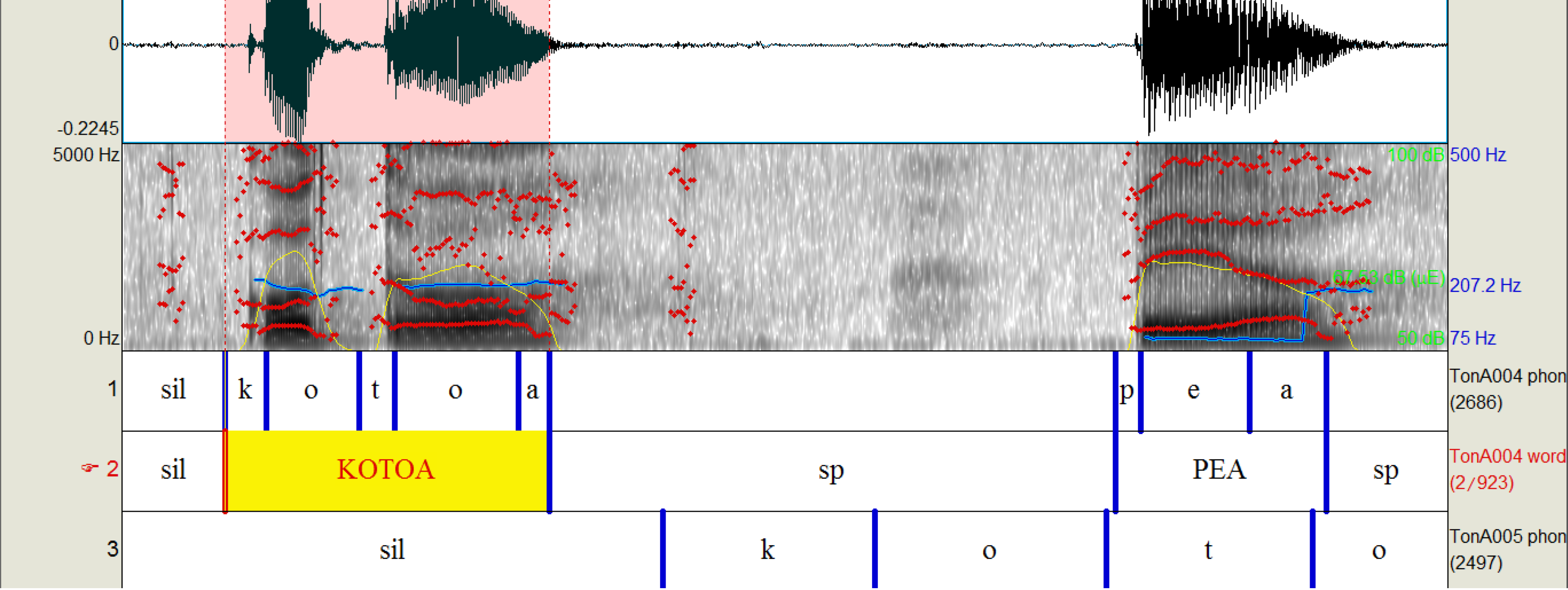
TRAINED ON CLEAN VS. ON DIRTY

TonA001 vs. TonA005
Clean File. Beg. of Recording



TRAINED ON CLEAN VS. ON DIRTY

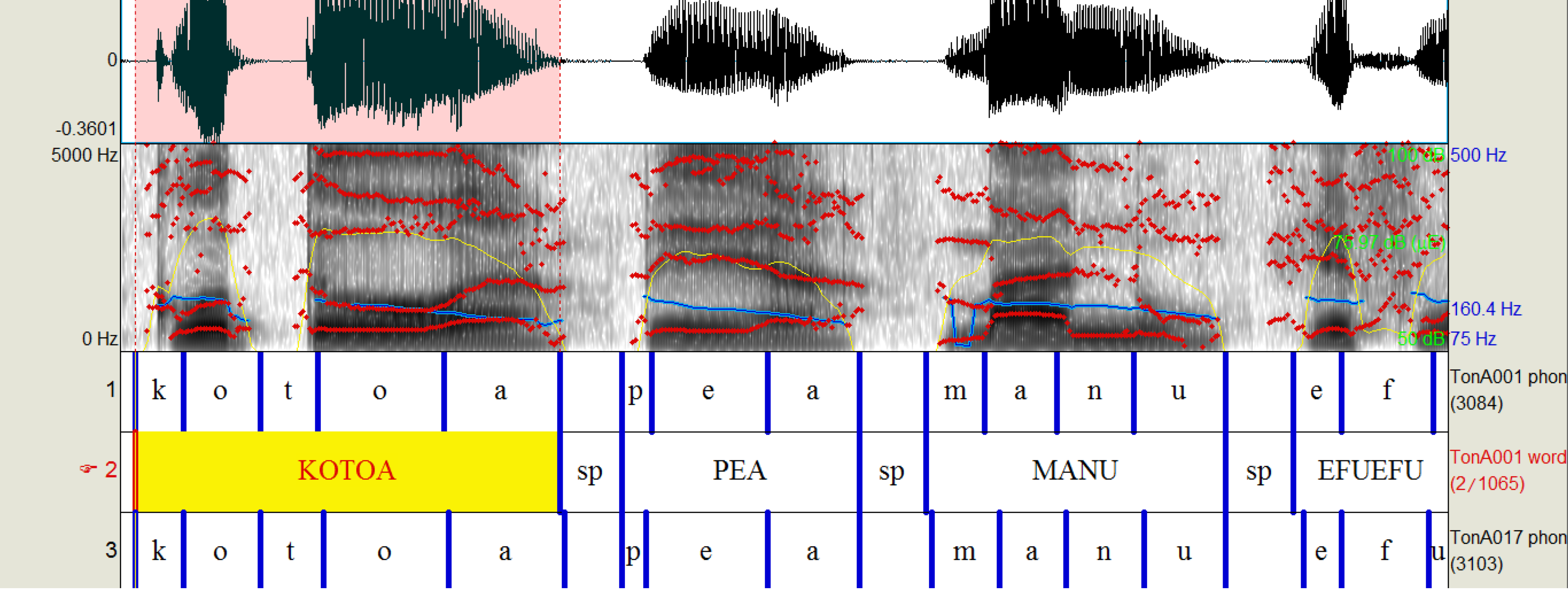
TonA001 vs. TonA005
Clean File. End of Recording



TRAINED ON CLEAN VS. ON DIRTY

TonA001 vs. TonA005

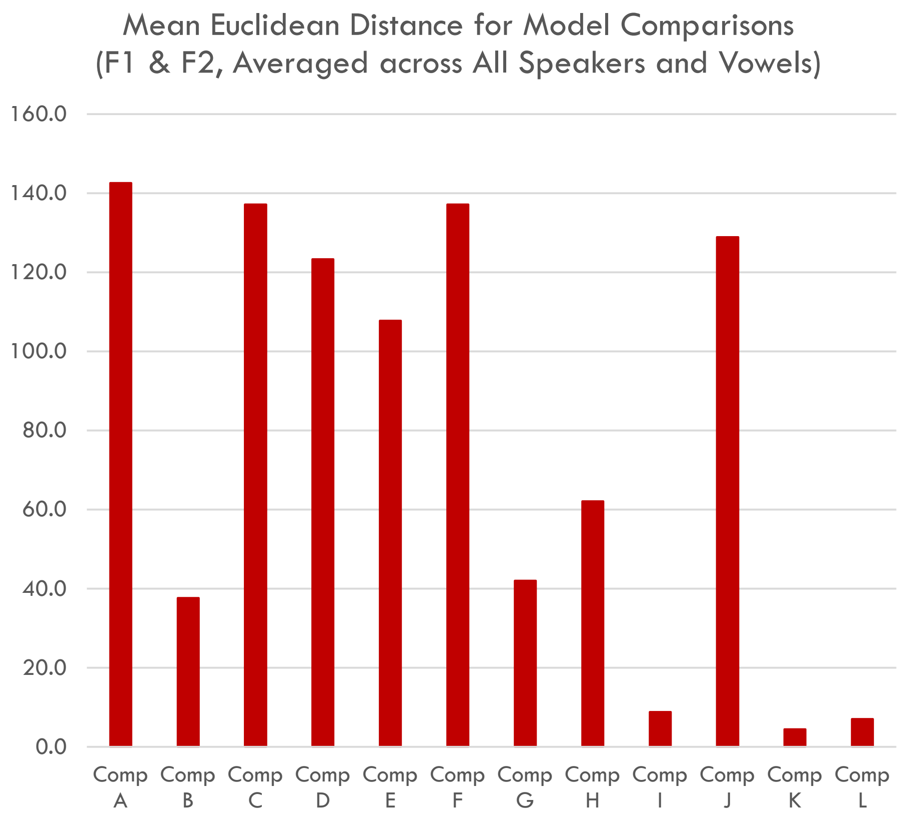
Dirty File. Beg. of Recording



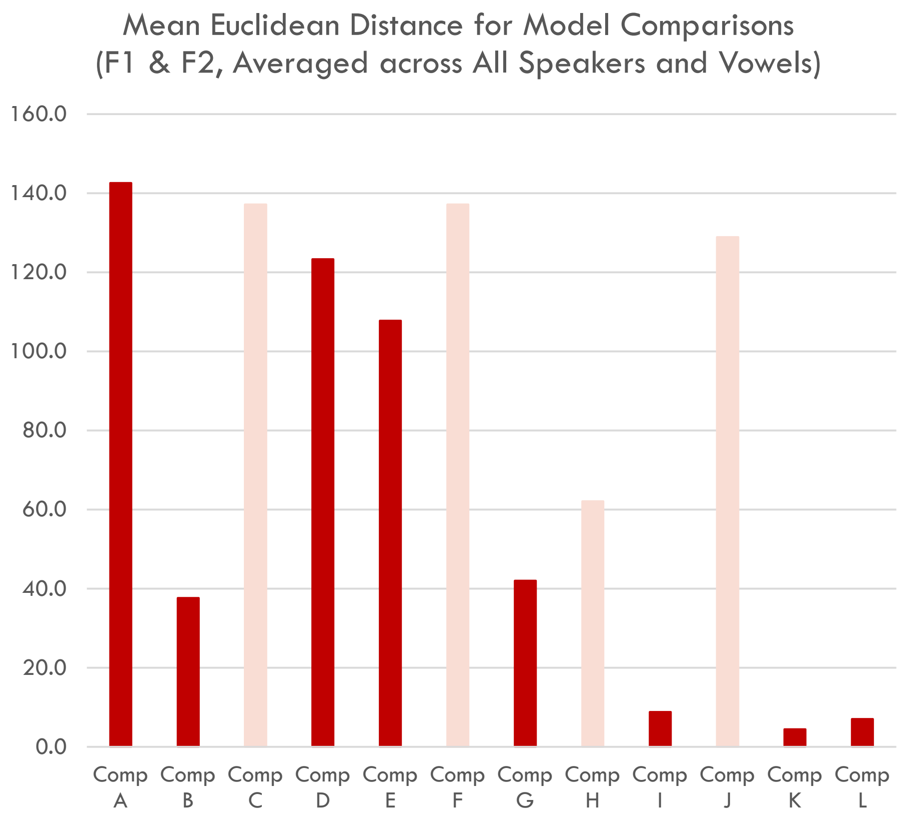
TRAINED ON FILES TO BE ALIGNED?

TonA001: Yes
TonA017: No

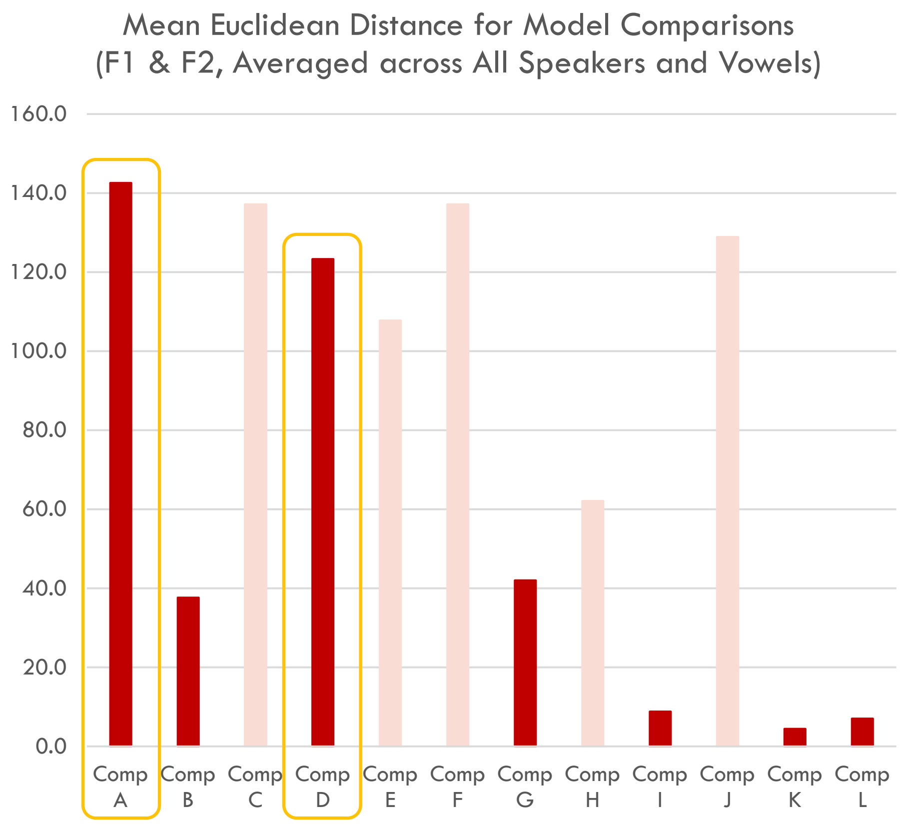
QUANTITATIVE MODEL COMPARISON



QUANTITATIVE MODEL COMPARISON

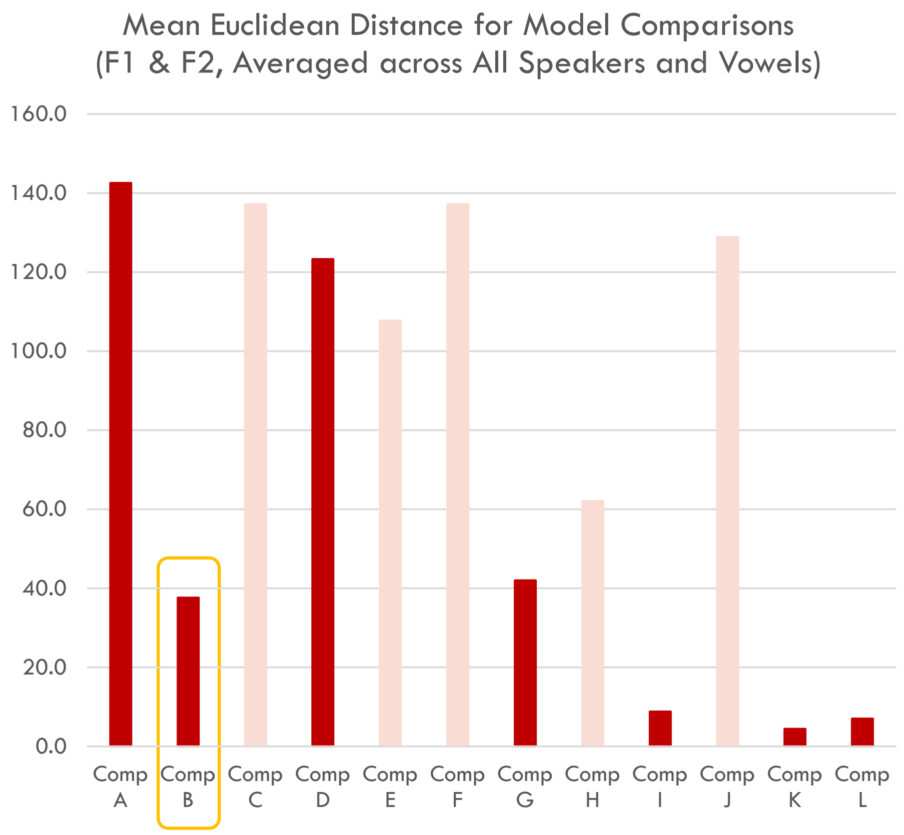


QUANTITATIVE MODEL COMPARISON



Cleaning up the files used for training the acoustic models had a large effect on the alignments. (Comparisons A and D)

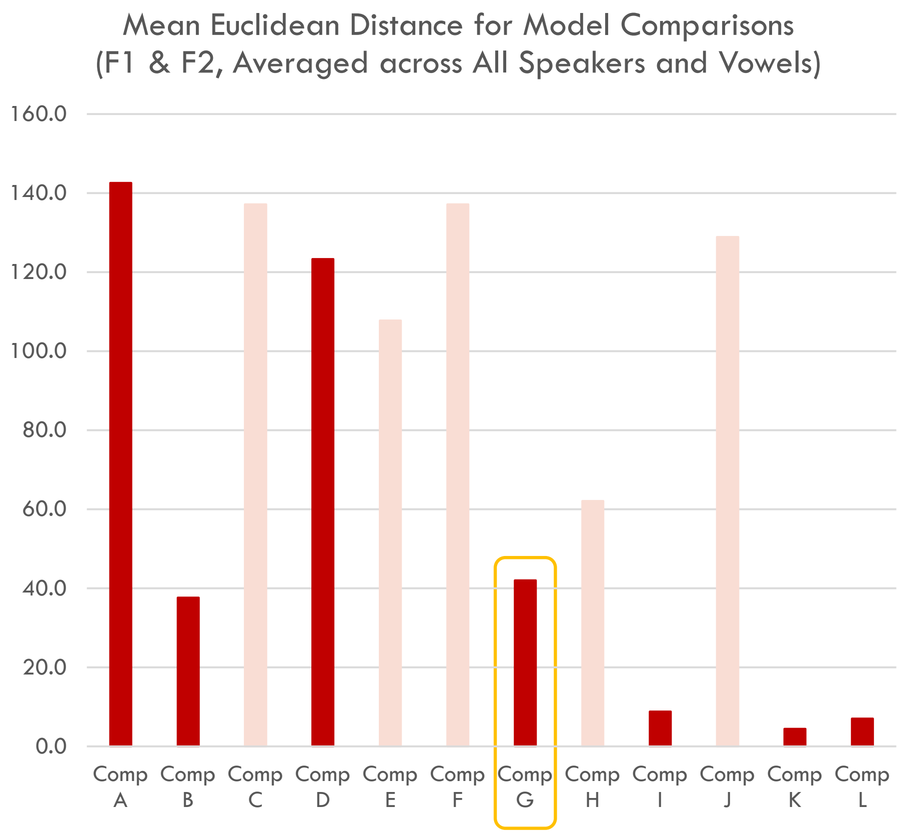
QUANTITATIVE MODEL COMPARISON



Cleaning up the files used for training the acoustic models had a large effect on the alignments. (Comparisons A and D)

Including dirty files along with clean files in the training data had a moderate effect on the alignments. (Comparison B)

QUANTITATIVE MODEL COMPARISON

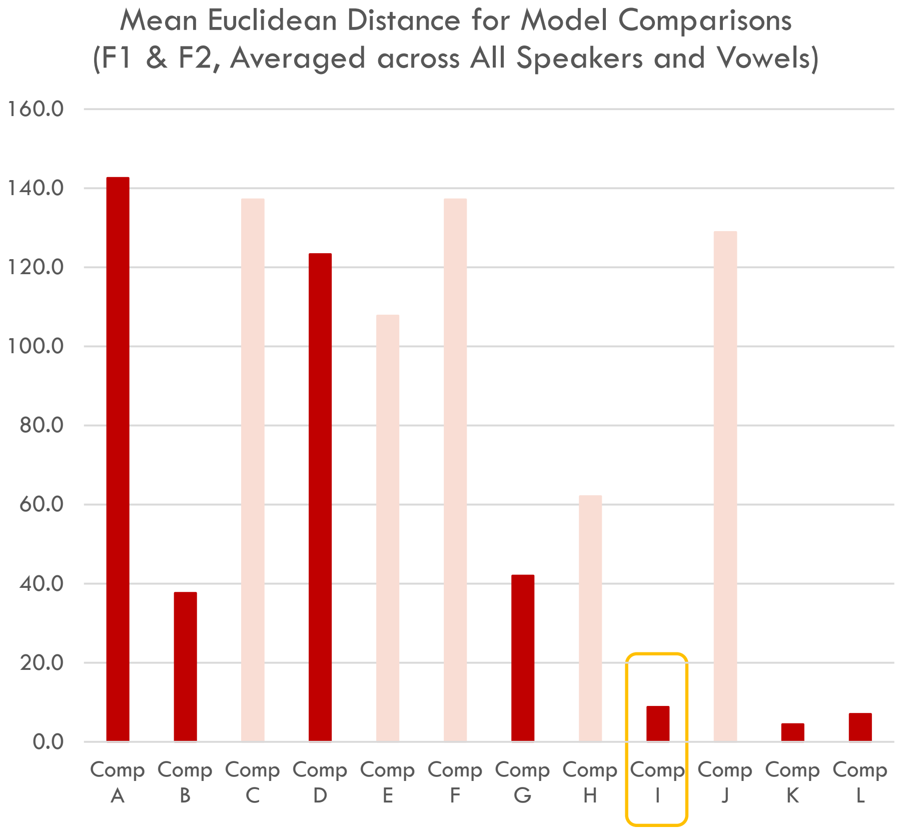


Cleaning up the files used for training the acoustic models had a large effect on the alignments. (Comparisons A and D)

Including dirty files along with clean files in the training data had a moderate effect on the alignments. (Comparison B)

Changing the Targetrate setting from 100000 to 125000 had some effect on the alignments. (Comparison G)

QUANTITATIVE MODEL COMPARISON



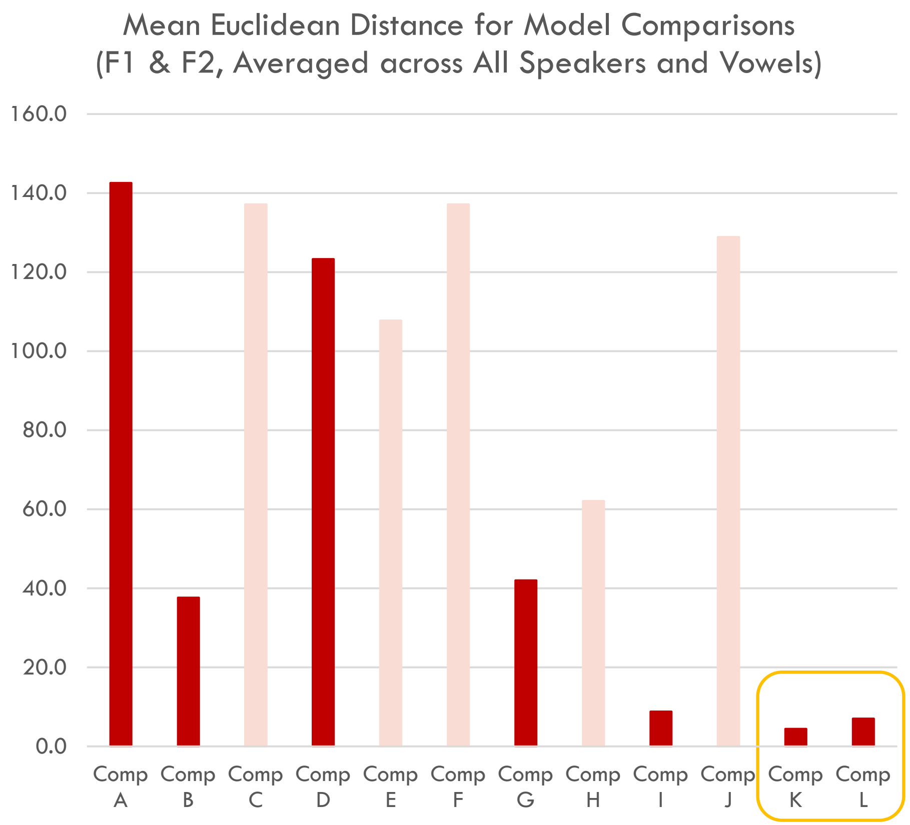
Cleaning up the files used for training the acoustic models had a large effect on the alignments. (Comparisons A and D)

Including dirty files along with clean files in the training data had a moderate effect on the alignments. (Comparison B)

Changing the Targetrate setting from 100000 to 125000 had some effect on the alignments. (Comparison G)

Whether the data used to train the acoustic model included the exact files to be aligned had little effect on the alignments. (Comparison I)

QUANTITATIVE MODEL COMPARISON



Cleaning up the files used for training the acoustic models had a large effect on the alignments. (Comparisons A and D)

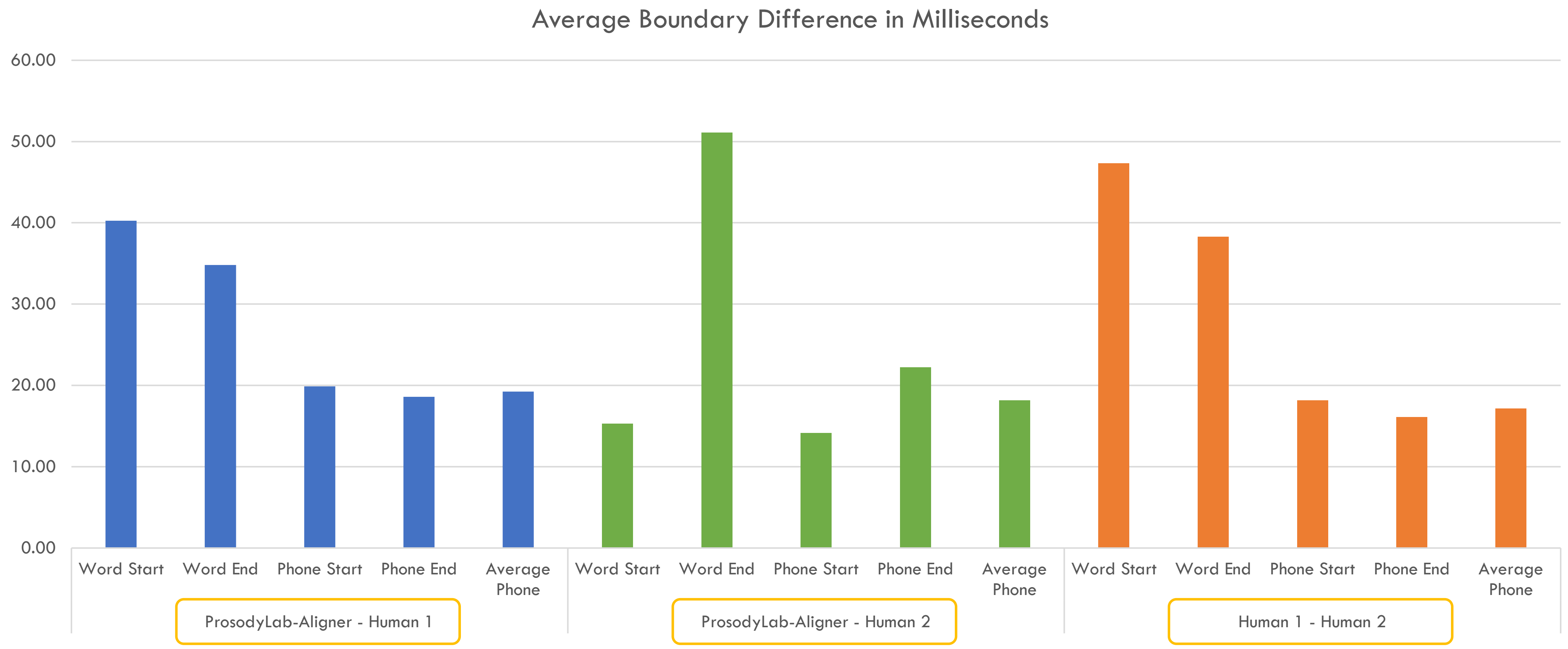
Including dirty files along with clean files in the training data had a moderate effect on the alignments. (Comparison B)

Changing the Targetrate setting from 100000 to 125000 had some effect on the alignments. (Comparison G)

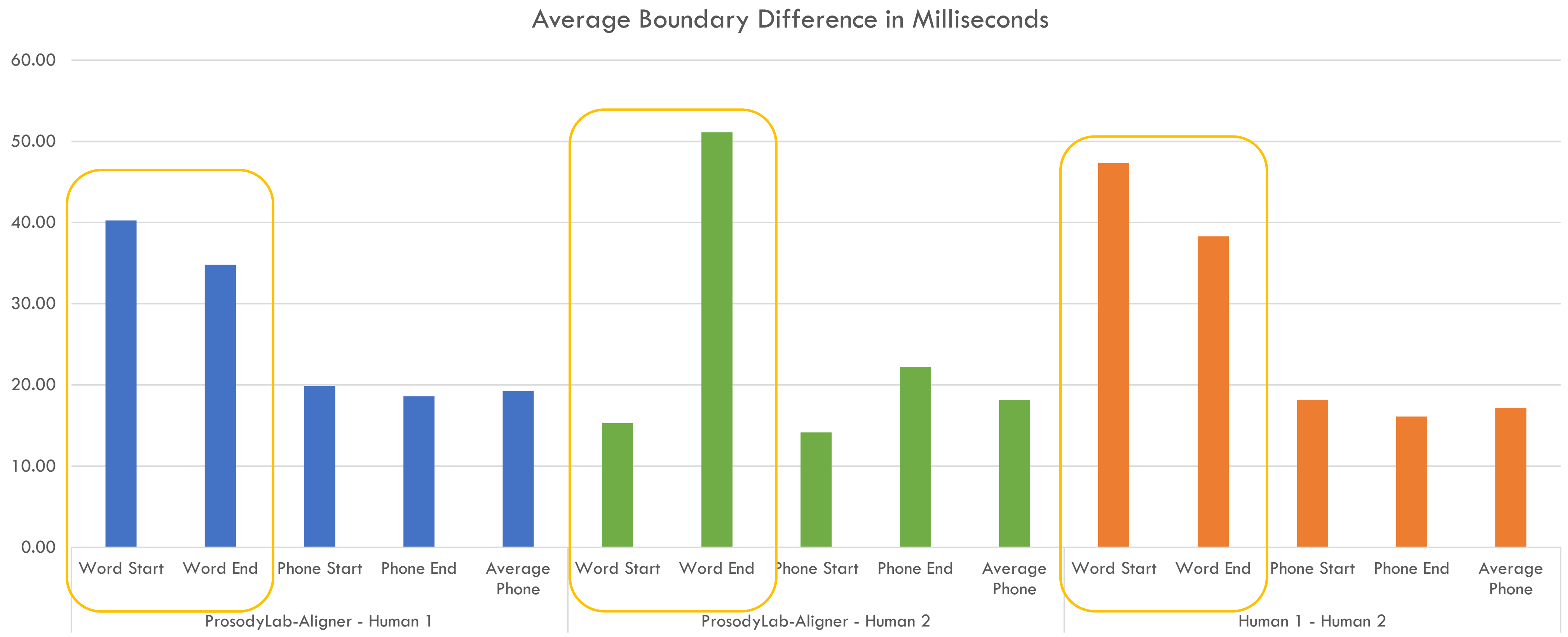
Whether the data used to train the acoustic model included the exact files to be aligned had little effect on the alignments. (Comparison I)

The number of epochs in each cycle of the acoustic model training process had little effect on the final alignments. (Comparisons K and L)

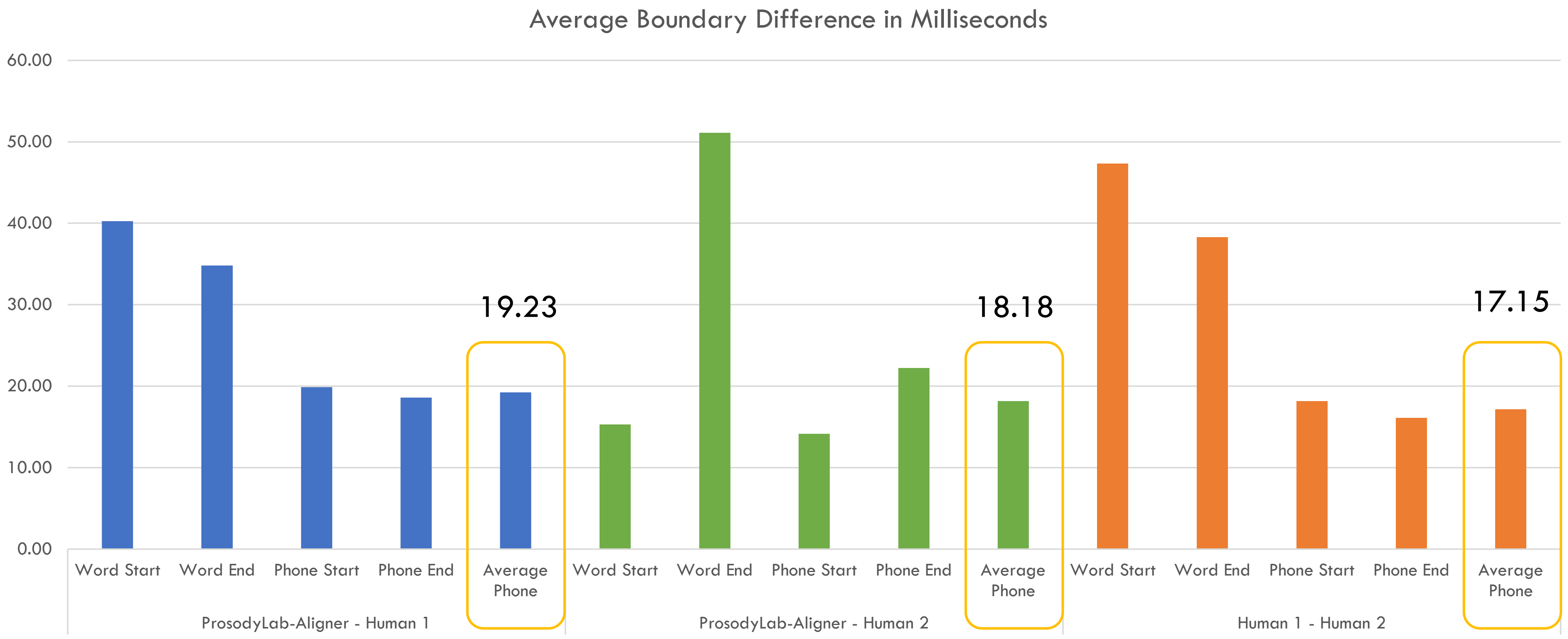
DIFFERENCE BETWEEN PL-A AND HUMAN ALIGNERS



DIFFERENCE BETWEEN PL-A AND HUMAN ALIGNERS



DIFFERENCE BETWEEN PL-A AND HUMAN ALIGNERS



PL-A SUMMARY AND RECOMMENDATIONS

Removing background noise from files used to train acoustic models seems to improve alignments, whether the files to be aligned contain background noise or not.

Cleaning files to be aligned also seems to improve performance, though not as much as cleaning the training files does.

It is better to use a smaller number of clean files than a larger number of mixed clean and dirty files when training acoustic models, even if the files to be aligned are dirty.

It is acceptable to use the same files in both the training and the alignment processes.

The default Targetrate setting of 100000 seems to produce better alignments than the adjusted 125000 setting.

Increasing the number of epochs used in the training process did not produce better alignments, though it did increase the time required to train the acoustic models.

MONTREAL FORCED ALIGNER

A New Alternative

MONTREAL FORCED ALIGNER

Created at the same lab as Prosodylab-Aligner

Like PL-A, can train and align same data or use pretrained acoustic model

Uses Python scripts like PL-A

Uses a different underlying technology:

- Kaldi ASR toolkit instead of HTK

Goes through three stages of training:

- First pass with monophone models
- Second pass using triphone models, which take into account the sound on both sides of the target phone
- Final pass that enhances triphone models by taking into account speaker differences

Has been used on:

- Bulgarian, Mandarin, Croatian, Czech, French, German, Hausa, Korean, Polish, Portuguese, Russian, Swahili, Spanish, Swedish, Thai, Turkish, Ukrainian, Vietnamese, English, Afrikaans, English, Ndebele, Xhosa, Zulu, Setswana, Sesotho sa Leboa, Sesotho, siSwati, Tshivenda, Xitsonga (working on Japanese)

ADVANTAGES OF MONTREAL FORCED ALIGNER

1. Accounts for interspeaker differences by considering speaker ID during acoustic model training.
2. Can align for multiple speakers in the same file
3. Can align without a dictionary if working from a fairly transparent and consistent orthography.
4. Does not crash when encountering out-of-dictionary words
   - unknown word marked as <unk> in the output and list of unknown words generated
5. Automatically strips punctuation from ends of words in transcripts and converts capital letters to lowercase.
6. Accepts two kinds of transcription inputs: PL-A format or Praat TextGrid format

MFA INPUT

Audio Files

Must be in .wav format

Any sampling rate above 16kHz* accepted—consistent sampling rate for each speaker

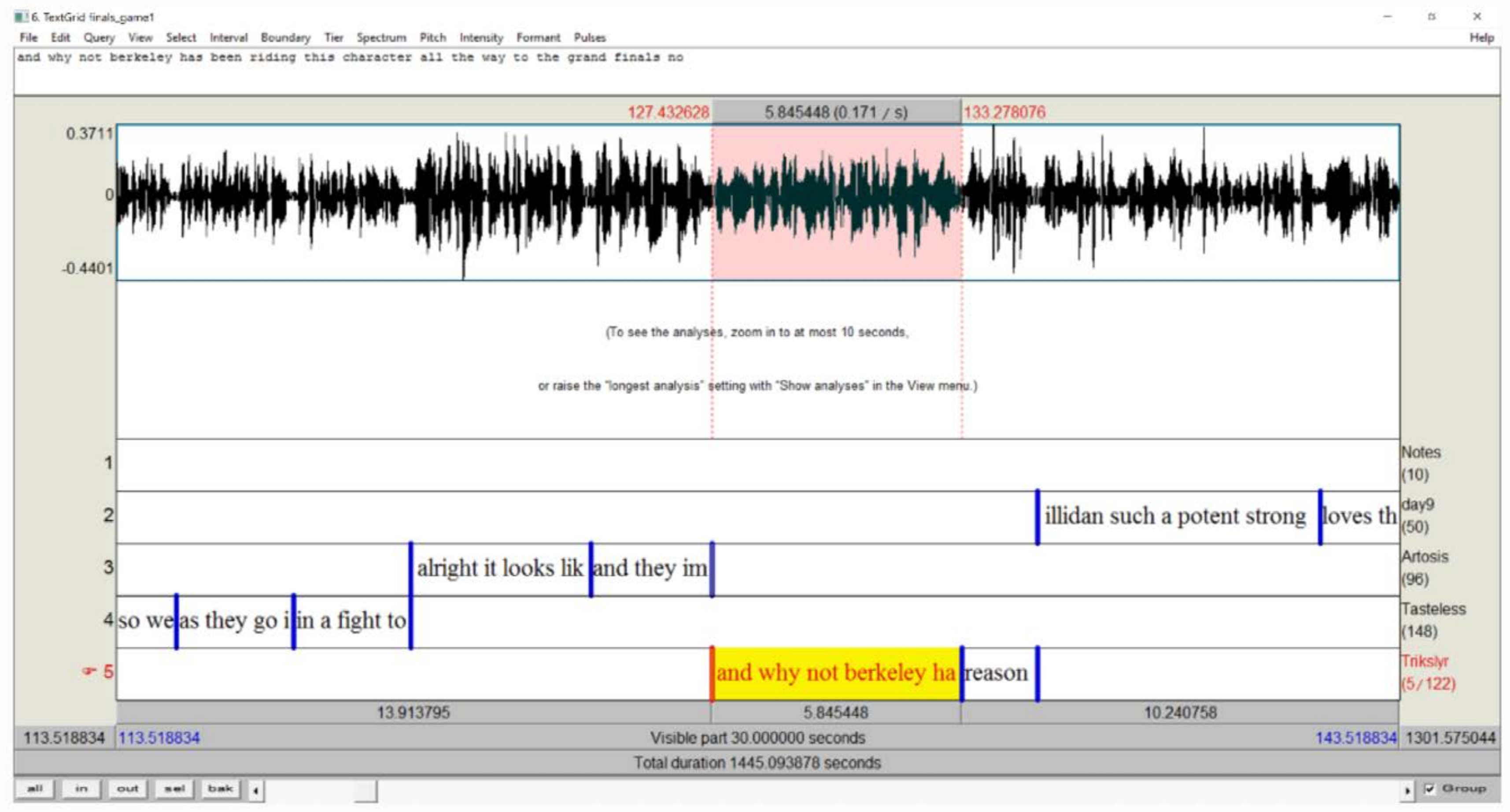
Audio "chunks" should be less than 30 seconds (sound files for PL-A format and intervals for Textgrid format)

Transcription Files

Two allowable formats:

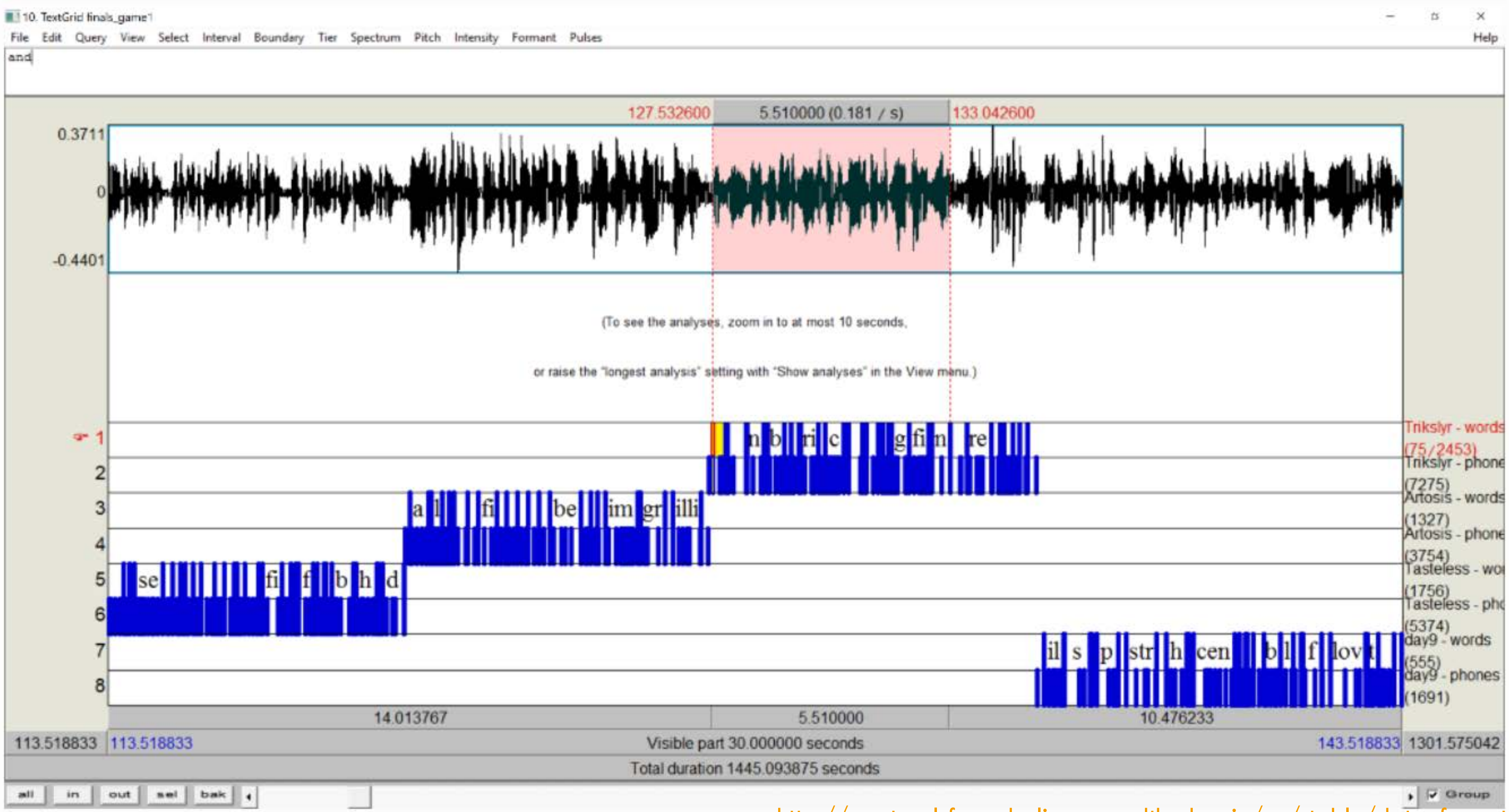
1. PL-A format (plain text, as described in previous slides)
2. TextGrid format (with transcribed "chunks" > 100ms and < 30 seconds

INPUT EXAMPLE (TIER NAME = SPEAKER ID)



http://montreal-forced-aligner.readthedocs.io/en/stable/data_format.html#prosodylab-format

OUTPUT EXAMPLE (WORD AND PHONE TIERS)



http://montreal-forced-aligner.readthedocs.io/en/stable/data_format.html#prosodylab-format

TRAINING AND ALIGNMENT

Accomplished in one step using a Python script

```
bin/mfa_train_and_align corpus_directory [dictionary_path] output_directory
```

Training can be skipped if aligning with a pretrained model

```
bin/mfa_align [model_path] corpus_directory output_directory
```

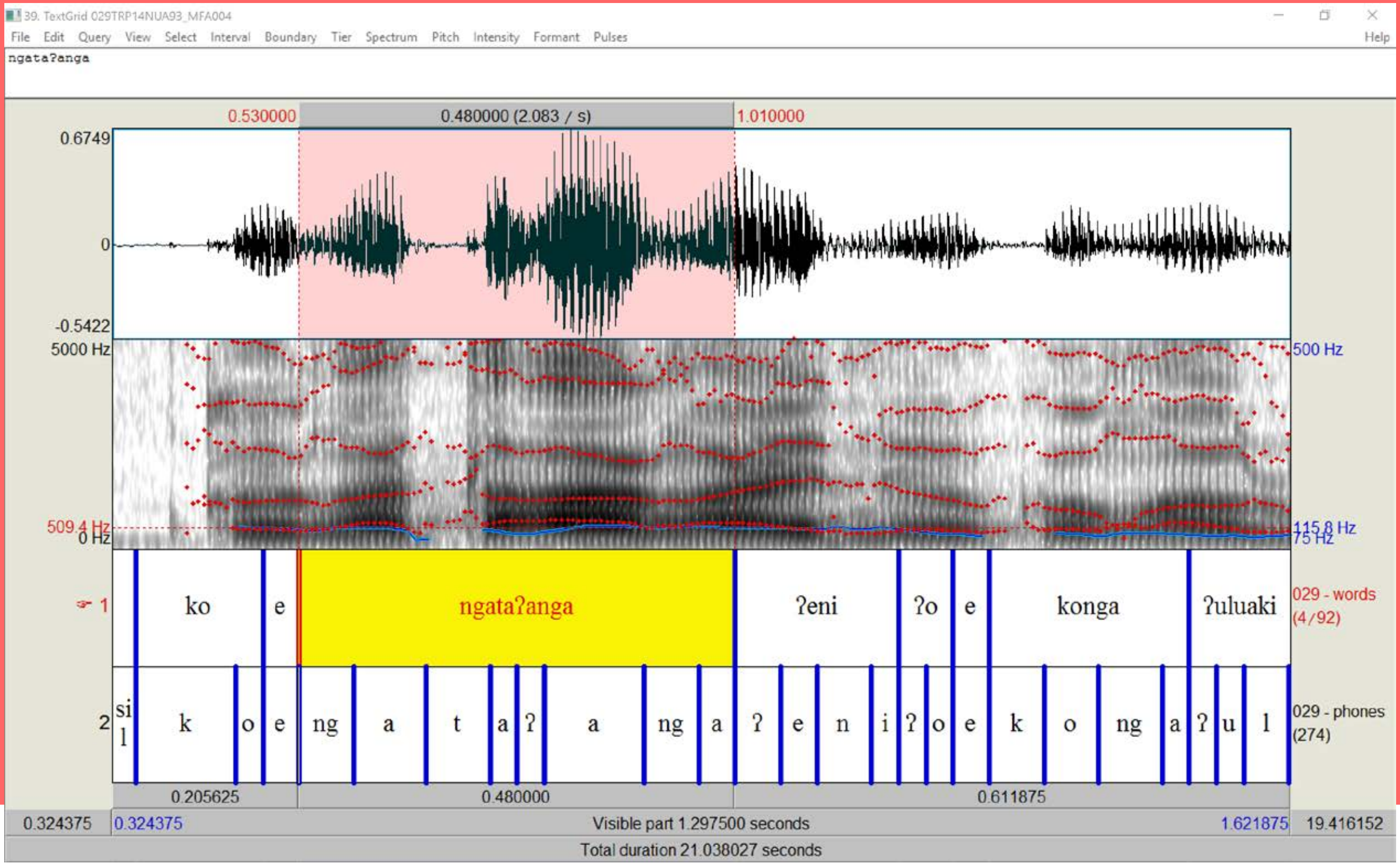
List of available options for both processes:
http://montreal-forced-aligner.readthedocs.io/en/stable/aligning.html

MFA TESTS

Model Training and Alignment

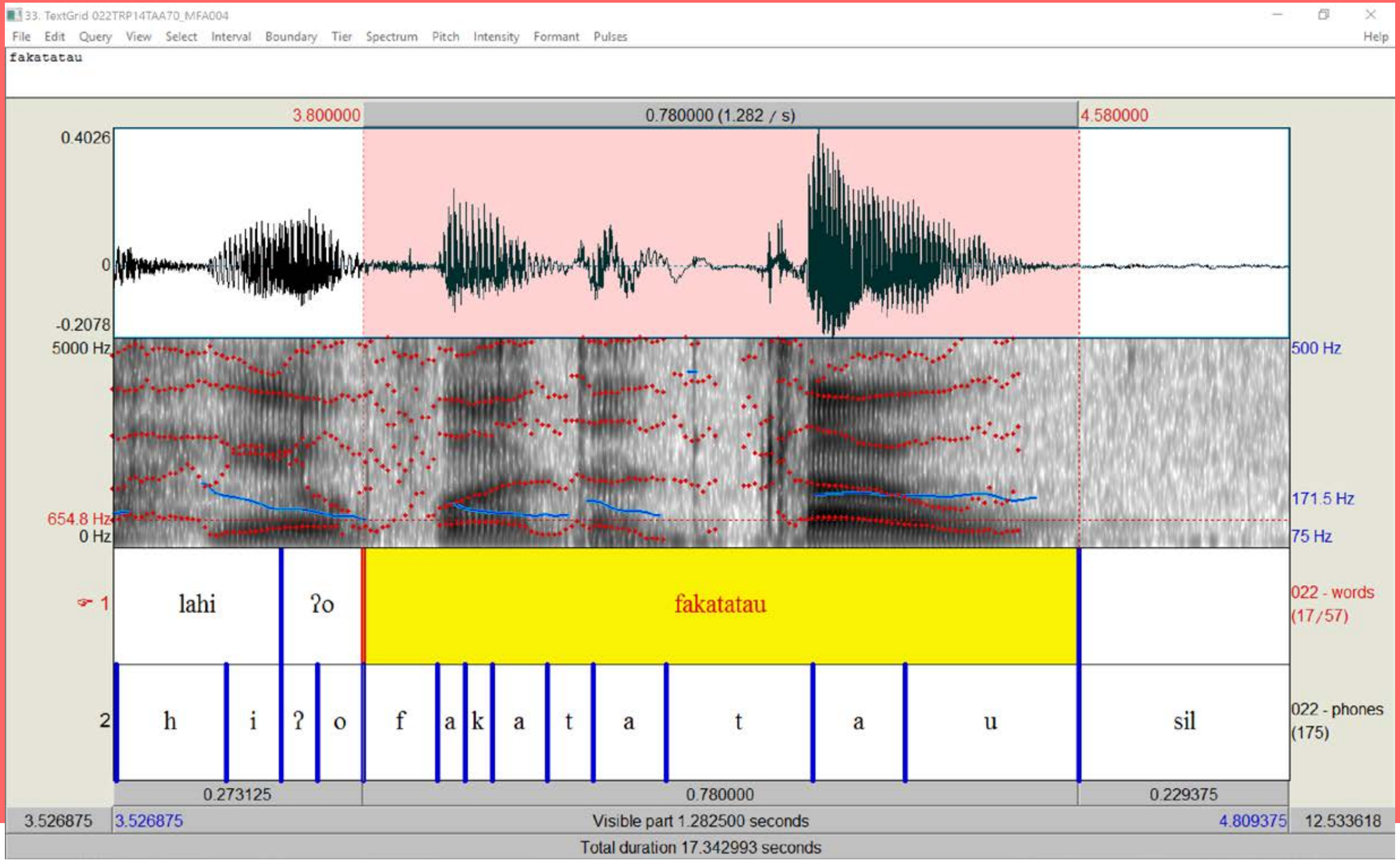
MFA TRAINING AND ALIGNMENT TESTS

Test ID #	Type and Number of Audio Files	Type of Transcription	Name of Acoustic Model Created	Runtime
MFA001	clean WL (same as PL-A) (22 files)	TextGrid	ton-001MFA.zip	:39
MFA002	clean WL (same as PL-A); dirty WL (20-second chunks) (38 files)	TextGrid	ton-002MFA.zip	1:14
MFA003	clean WL (same as PL-A); dirty WL (1-word chunks) (38 files)	TextGrid	ton-003MFA.zip	1:30
MFA004	clean (same as PL-A); dirty (1-word chunks); reading passage (59 files)	TextGrid	ton-004MFA.zip	1:30
MFA005	clean and dirty WL excerpts (10 files)	Text (PL-A)	(aligned only)	:02



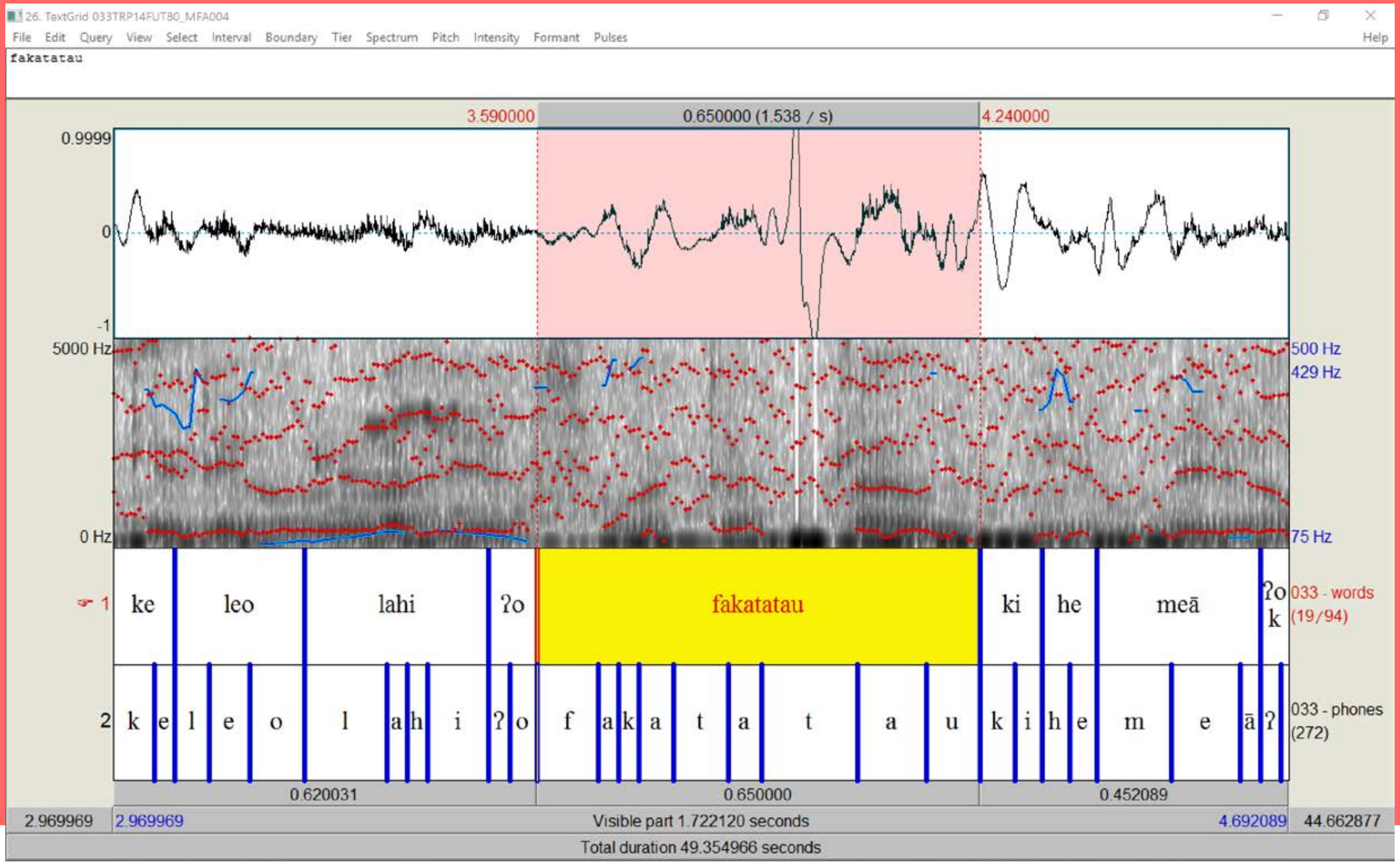
MFA ALIGNMENT

Connected Speech (Speaker 029)



MFA ALIGNMENT

Connected Speech (Speaker 22)



MFA ALIGNMENT

Connected Speech (Speaker 33)

MFA SUMMARY AND CONCLUSIONS

Quality

- Using MFA TextGrid input seems to eliminate the dirty file effects we saw with PL-A.
- MFA produced good alignments with long recordings, allowing us to preserve token context for analysis.

Efficiency

- In our experience, MFA file preparation was much more efficient than PL-A file prep.
- MFA's "no dictionary" option will save considerable time when we begin to analyze free conversation and interview speech. (Note, this may not be as effective for languages with less transparent orthography.)
- MFA's ability to process speech from multiple speakers in the same file will save prep time and preserve discourse context.

IMPLICATIONS AND APPLICATIONS

Feasibility and Efficiency

SUMMARY AND RECOMMENDATIONS

Efficiency

- Forced alignment can greatly reduce the time required to prepare files for acoustic analysis.
- It is **possible** and **efficient** to **force align field recordings,** even with background noise.
  - TextGrid input using MFA produces good alignments with less clean-up time
- The amount of time saved will vary by language and the type of analysis planned

Reliability and Validity

- Forced alignment **may improve general consistency and replicability**
- It's necessary to make **manual boundary adjustments of TextGrids** output from forced alignment
  - Positive: It allows you to dig deeply into an understudied language early on

ACKNOWLEDGEMENTS

Bell, Adrian V. and Marianna Di Paolo. 2014. University Research Council Grant, University of Utah. “Language, Ethnic Markers, and the Adaptation of Tongan Immigrants to Utah.”

Holt, Carter. 2015. “Boosting Phonetics Research through Technology.” Undergraduate Research Opportunity Program. Summer 2015 Research Assistantship. (Marianna Di Paolo, mentor)

Special thanks to Kyle Gorman, Michael McAuliffe, and Craig Johnson

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.

Boersma, Paul and David Weenink. 2015. Praat: doing phonetics by computer [Computer program] 5.4.22.

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.

Evanini, Keelan, Stephen Isard, and Mark Liberman. 2009. "Automatic formant extraction for sociolinguistic analysis of large corpora." INTERSPEECH.

Goldman, Jean-Philippe. 2011. "Esayalign: an automatic phonetic alignment tool under Praat." Interspeech-2011:3233-3236.

Gorman, Kyle, Jonathan Howell and Michael Wagner. 2011. Prosodylab-Aligner: A Tool for Forced Alignment of Laboratory Speech. Canadian Acoustics. 39.3. 192–193.

Kisler, Thomas, Florian Schiel, and Han Sloetjes. 2012. "Signal processing via web services: the use case WebMAUS." Digital Humanities Conference 2012.McAuliffe, Michael, Michaela Socolof, Sarah Mihuc, and Michael Wagner (2016). Montreal Forced Aligner [Computer program]. Version 0.5, retrieved 13 July 2016 from http://montrealcorpustools.github.io/Montreal-Forced-Aligner/.

Povey, Daniel, Arnab Ghoshal, Giles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. "The Kaldi Speech Recognition Toolkit." IEEE 2011 Workshop on Automatic Speech Recognition and Understanding.

Reddy, Sravana, and James Stanford. 2015. "Toward completely automated vowel extraction: Introducing DARLA." Linguistics Vanguard.

Rosenfelder, Ingrid. 2013. "Forced Alignment & Vowel Extraction (FAVE): An online suite for automatic vowel analysis." University of Pennsylvania Linguistics Lab, Last Modified December 8, 2013, accessed November 26. 2015. http://fave.ling.upenn.edu/index.html.

Rosenfelder, Ingrid, Joe Fruehwald, Keelan Evanini, and Jiahong Yuan. 2011. FAVE (Forced Alignment and Vowel Extraction) Program Suite.

Schiel, Florian, Christoph Draxler, Angela Baumann, Tania Ellbogen, and Alexander Steffen. 2012. "The Production of Speech Corpora."

Young, Steve, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Garth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. 1995-2006. The HTK Book (for HTK Version 3.4). edited by Microsoft Corporation (1995-1999) Cambridge Univerisity Engineering Department (2001-2006).