EMU-SDMS: R CENTRIC SEMI-AUTOMATIC SPEECH DATABASE PROCESSING AND ANALYSIS

Raphael Winkelmann, Jonathan Harrington

Institute of Phonetics and Speech Processing, Ludwig-Maximilians University of Munich, Germany.
raphael | jmh@phonetik.uni-muenchen.de

ABSTRACT

We present a set of tools centred around the EMU Speech Database Management System for building, querying and analysing speech corpora. A new approach is to situate all stages between digitised waveforms and the final graphical and statistical analysis within the R programming environment. All these stages will be illustrated through a comparative analysis of the formants of the Australian and New Zealand English vowel spaces. For this purpose, emuR package functions will be used to directly invoke web services provided by the Bavarian Archive for Speech Signals (e.g. WebMAUS) for building linked hierarchical annotations between orthographic and phonological levels with time-stamps into the signals. The analyses presented employ both the emuR and other graphical and statistical R packages which are used well beyond the speech science community. This illustrates that the EMU-SDMS is part of the vast R package eco-system which includes state-of-the-art methods that are improved constantly.

Keywords: EMU-SDMS, corpus phonetics, speech databases, automatic annotation

1. INTRODUCTION

Throughout the digital age, the basis for most quantitative empirical phonetic research has been a digital collection of speech related signals (see amongst others [11], [12]). These signal collections, the most prevalent of which are collections of acoustic recordings, are often further enriched by adding meta information, orthographic transcripts as well as various forms of segmentation and labeling that can vary in granularity (e.g. word vs. syllable vs. phonetic segmentation). These additional resources complement the primary media files. As outlined in [6], there is a clear trend in corpus phonetics to produce ever larger datasets, either by collecting large amounts of new data or by accessing the ever growing number of speech corpora available in online repositories (see, for example, the Virtual Language Observatory (VLO) of the Common Language Resources and Technology Infrastructure (CLARIN) https://vlo.clarin.eu/ or the Bavarian Archive for Speech Signals (BAS) Repository http://hdl.handle.net/11858/00-1779-0000-0006-BF00-E).

The size of these datasets has already reached a point where tools are needed to either fully- or semi-automatically process various aspects of the additional complementing information mentioned above. Forced alignment (FA) in combination with grapheme to phoneme conversion (G2P) techniques, such as the ones developed by [10, 4, 9], provide efficient ways of processing the complementary data. These techniques can be employed to provide automatic segmentation and labeling, if an orthographic transcript is available. Even if, under certain conditions, the automated results lack the desired precision, the time-saving aspect of automatically locating and segmenting, for example, the phonemic structure, optionally followed by a manual correction step, usually outweigh the cost of a more precise fully manual annotation.

The EMU speech database management system (EMU-SDMS) [13] introduced the concept of having an all-in-one solution for working with speech databases in the R language and environment for statistical computing and graphics [8]. It allows users to generate, manipulate, query, analyse and manage speech databases all from within R. Recent developments of the EMU system offer users the additional ability to script annotation structures and call the BAS web services (http://hdl.handle.net/11858/00-1779-0000-0028-421B-4), for example, to perform the aforementioned G2P and FA. Both of these functionalities are available via a dedicated set of R functions that provide database annotation validity assurance.

The aim of this paper is to demonstrate some of these new features by describing a hands-on use-case that shows how these features can be used to compare the Australian (AE) and New Zealand English (NZE) vowel spaces. Only a handful of R commands are necessary to achieve such an analysis starting only with digitised waveforms and their

orthographic transcriptions. As automatic segmentation and labelling procedures will be employed, no manual annotation steps will be used.

2. COMPARATIVE VOWELS ANALYSIS

We present a similar but simplified version of two of the analyses in [11]. The goal is to visually compare formants of the AE and NZE vowel spaces using F1/F2 formant plots to showcase the tools that the EMU-SDMS provides. Both monophthongs and diphthongs will be analysed.

2.1. Data

We assume that the acoustic data have already been collected and orthographic transcripts generated; that is, the following file structure is present:

audio	ortho. transcript
NZE/NZE_1.wav	NZE/NZE_1.txt
NZE/NZE_2.wav	NZE/NZE_2.txt
...	...
AE/AE_1.wav	AE/AE_1.txt
AE/AE_2.wav	AE/AE_2.txt
...	...

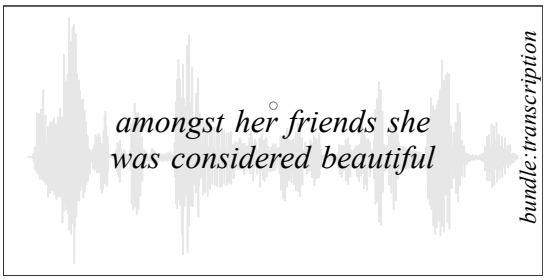
where `NSE/` and `AE/` are subdirectories containing the respective English varieties. File collections of the above form are referred to as `txtCollections` within the EMU system. These collections are commonplace in the speech science community, largely due to tools such as SpeechRecorder [2] being able to generate them synchronously with prompted speech data acquisition and other tools such as various BAS web services [5] expecting them as input. Further, these collections can easily be generated from other forms of annotation + audio file collections. Hence, a collection of this kind should be familiar to most phoneticians and the steps described in the following should be easily transferable to other similar and widely available data sets.

Compared with the recordings used in [11], and other similar studies (e.g. [12]) here a random set of single sentence utterances of each English variety is analysed (281 recordings in total). Furthermore, for the sake of simplicity the dataset will consist of recordings of male speakers only.

2.2. Data Preparation

To process the recordings and plain text files, an EMU database (emuDB) has to be created. This is achieved using the `convert_txtCollection()` function provided by the emuR package. The resulting emuDB will contain only a single timeless annotation level called *bundle* containing an additional attribute *transcription* that holds the content of the plain text files[1]. Figure 1 shows an example of this bare-bones annotation structure.

Figure 1: Single annotation item on timeless *bundle* level containing the content of the corresponding plain text file as is created by `convert_txtCollection()`



This single annotation item will later form the root node in a more complex hierarchical annotation structure (see Figure 2). The subdirectories `NSE/` and `AE/` will be placed into separate session folders by the `convert_txtCollection()` routine.

2.2.1. Calling the BAS web services

As of version 1.0.0 of emuR it is possible to call various BAS web services using specialized emuR functions that follow the naming convention: `runBASwebservice_*()`, where `*` is a placeholder for the name of the web service. After loading the generated emuDB (`db = load_emuDB()`), we call `runBASwebservice_all()`, which chains all available web services. The web service calls that are performed in consecutive order are:

1. `runBASwebservice_g2pForTokenization()`
2. `runBASwebservice_g2pForPronunciation()`
3. `runBASwebservice_chunker()` (optional depending on audio file length)
4. `runBASwebservice_maus()`
5. `runBASwebservice_minni()` (optional depending on argument)
6. `runBASwebservice_pho2sylCanonical()`
7. `runBASwebservice_pho2sylSegmental()`

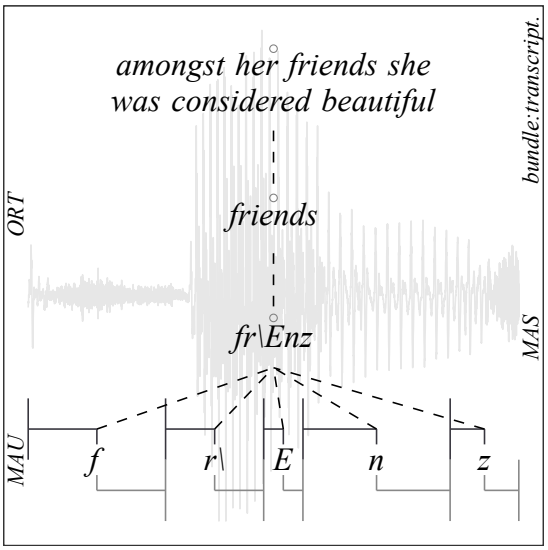
Figure 2 shows an excerpt of the automatically constructed hierarchical annotation structure.

This annotation structure can now be used to extract the segments of interest, including the formant tracks belonging to these segments. The emuR package provides two functions (`query()` and `get_trackdata()`) to achieve this.

2.3. Monophthongs

To compare the AE and NZE monophthong vowel spaces, the *heed*, *hid*, *hood* and *hud* (IPA: /i ɪ ʊ ʌ/; SAMPA: /i I U V/)) vowel subset will be extracted

Figure 2: Example of a resulting hierarchical annotation structure returned by `runBASwebservice_all()`



for both English varieties and the centroid of each vowel class plotted onto the F1/F2 plane.

2.4. Querying, data extraction and preparation

A new feature of the emuR package (as of version 1.1.0) is that both the `query()` and `get_trackdata()` functions implement a new `tibble` result type (see https://www.tidyverse.org/) which allows their output to easily be processed by other packages. The aim is to avoid using objects specific to the emuR package, which was the case with the legacy S3 classes `trackdata` and `emusegs`. This drastically improves the scope of usability of `query()` and `get_trackdata()` output in other R packages. In this example, we use the `tibble` result type in both the `query()` and `get_trackdata()` calls. The `query()` function extracts the vowel segments including time stamps into the signal, while `get_trackdata()` applies the formant estimation function (`forest()`) provided by EMU-SDMS's wrassp R package to the queried segments and extracts the calculated formant tracks.

```
# query vowels
sl = query(db,
  "MAU =~ [iIUV]",
  resultType = "tibble")
# get formant tracks for segments
td = get_trackdata(db, sl,
  onTheFlyFunctionName = "forest",
  resultType = "tibble")
```

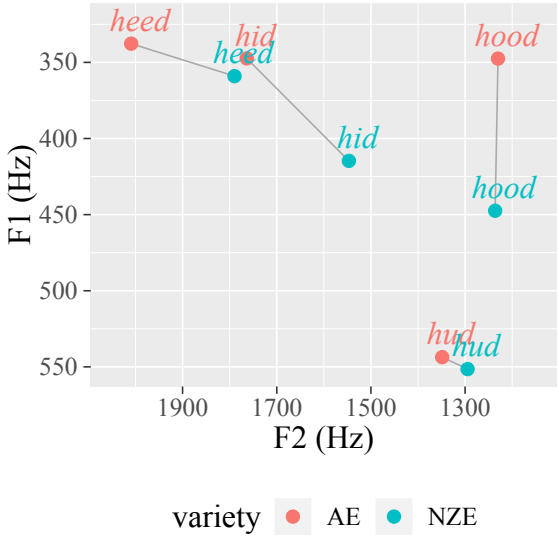
The resulting `td` object contains both the segment information returned by `query()` and formant sample times and values. Using the dplyr R package we are now able to perform post-processing such as outlier removal and vowel class centroid calculation:

```
# remove 0 values (T1 = F1, T2 = F2)
td = td %>% filter(T1 != 0 & T2 != 0)
# calculate centroids
centroids = td %>%
  group_by(session, labels) %>%
  summarise(F1 = mean(T1),
    F2 = mean(T2))
```

2.5. Visual comparison

Figure 3 shows the resulting centroids plotted on the F1/F2 plane for NZE and AE monophthongs *heed*, *hid*, *hood* and *hud*.

Figure 3: F1/F2 comparison of the available AE and NSE *heed*, *hid*, *hood* and *hud* vowels



As in [11], the *hid* and *hood* vowels of NZE are lowered relative to the AE vowels in Figure 3. Additionally, the *hid* vowel is more centralized in the NZE variety. The *hud* vowel is in line with the *hard* and *hod* vowel space region in [11], which is slightly lowered and retracted for NZE vowels. Compared to the findings in [11] in our simple use-case the *heed* vowels in AE are slightly raised and fronted compared to those in NZE.

Figure 3 was generated using the following code, which uses the previously calculated `centroids` object and the ggplot2 package:

```
# plot centroids onto F1/F2 plane
ggplot(centroids) +
  aes(y = F1, x = F2, col = session,
    label = labels, group = labels) +
  geom_point() +
  geom_text() +
```

```
  scale_y_reverse() +
  scale_x_reverse()
```

The above code is a simplified version of the actual code used to generate the plot. Label mapping, exact label placement and legend theming code snippets were omitted for the sake of brevity.

2.6. Rising diphthongs

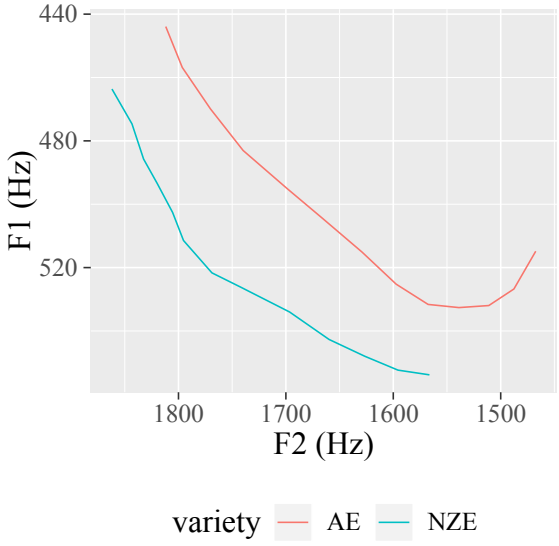
An analysis of the rising *hay* (IPA: /æɪ/, SAMPA: /{I/) diphthong will be performed. The brief R code below is all that is necessary to replicate the formant trajectory visualization in Section 3.3 of [11]. As no manual diphthong onset/offset target marking was performed on the available dataset (see Section 2.3 in [11]), only the central 60% of the diphthong is analysed to compensate for any potential coarticulatory effects and/or unprecise boundaries caused by the FA.

```
# query diphthong
sl_dip = query(db,
  "MAU == {I",
  resultType = "tibble")
# get formant tracks
td_dip = get_trackdata(db,
  sl_dip,
  onTheFlyFunctionName = "forest",
  resultType = "tibble")
# normalize length of segments
td_dip_norm = normalize_length(td_dip
  )
# extract central 60%
td_dip_norm = td_dip_norm %>%
  group_by(sl_rowIdx) %>%
  filter(times_norm >= 0.2 & times_
    norm <= 0.8)
# calculate mean trajectories
td_dip_norm_average = td_dip_norm %>%
  group_by(session, times_norm) %>%
  summarise(F1 = mean(T1),
    F2 = mean(T2))
# visualize
ggplot(formants_norm_average) +
  aes(x = F2, y = F1, col = session)
    +
  geom_line() +
  scale_y_reverse() +
  scale_x_reverse()
```

The final ggplot call produces Figure 4.

Unlike in [11], the *hay* diphthong was slightly lower in NZE (higher F1) compared to AE. It is worth noting that [11] point out that rising diphthongs are generally relatively similar in NZE and AE. Hence, our slightly deviating results are probably due to the size and design of our example data set.

Figure 4: F1/F2 trajectory comparison of the central 60% rising *hay* (IPA: *æɪ*, SAMPA: *{I*) diphthong in AE vs. NZE



3. DISCUSSION

In this short use-case we demonstrated how the tools provided by the EMU-SDMS can be used to efficiently generate, process and analyse speech databases. Only a handful of R commands were necessary for the visual analyses of both monophthongs and diphthongs. The steps to perform the described analyses are identical regardless of the size and design of the available dataset.

The modular design of the EMU-SDMS permits many other external tools to be used for the various stages of processing. For example, others tools such as the OpenSMILE feature extraction tool [3] or the Montreal Forced Aligner [7] could easily be integrated with minimal R scripting effort. As such, the EMU-SDMS provides a simple, flexible, efficient, all-in-one solution for state of the art phonetic analysis in the R environment.

4. ACKNOWLEDGMENTS

Research supported by the German Federal Ministry of Education and Research (BMBF) via the CLARIN-D project [1].

5. REFERENCES

[1] CLARIN, 2017. CLARIN-D web page. https://www.clarin-d.net last accessed: 2019-03-27.

[2] Draxler, C., Jänsch, K. 2004. SpeechRecorder - a Universal Platform Independent Multi-Channel Audio Recording Software. *Proceedings of the IV.*

International Conference on Language Resources and Evaluation Lisbon, Portugal. 559–562.

[3] Eyben, F., Weninger, F., Gross, F., Schuller, B. 2013. Recent developments in opensmile, the munich open-source multimedia feature extractor. *Proceedings of the 21st ACM international conference on Multimedia*. ACM 835–838.

[4] Kisler, T., Reichel, U., Schiel, F. 2017. Multilingual processing of speech via web services. *Computer Speech & Language* 45, 326 – 347.

[5] Kisler, T., Schiel, F., Sloetjes, H. 2012. Signal processing via web services: the use case WebMAUS. *Proceedings Digital Humanities 2012, Hamburg, Germany* Hamburg. 30–34.

[6] Liberman, M. 2019. Corpus phonetics. *Annual Review of Linguistics* 5(1). Published online on August 22, 2018.

[7] McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., Sonderegger, M. 2017. Montreal forced aligner: Trainable text-speech alignment using kaldi. *Interspeech* 498–502.

[8] R Core Team, 2018. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing Vienna, Austria. ISBN 3-900051-07-0.

[9] Reichel, U. D. 2012. PermA and Balloon: Tools for string alignment and text processing. *Proceedings Interspeech* Portland, Oregon. paper no. 346.

[10] Schiel, F. August 1999. Automatic Phonetic Transcription of Non-Prompted Speech. *Proceedings of the ICPhS* San Francisco. 607–610.

[11] Watson, C. I., Harrington, J., Evans, Z. 1998. An acoustic comparison between new zealand and australian english vowels. *Australian journal of linguistics* 18(2), 185–207.

[12] Wells, J. 1962. *A study of the formants of the pure vowels of British English*. PhD thesis University of London London, UK.

[13] Winkelmann, R., Harrington, J., Jänsch, K. September 2017. EMU-SDMS: Advanced speech database management and analysis in R. *Computer Speech & Language* 45, 392–410.

[1] see https://ips-lmu.github.io/The-EMU-SDMS-Manual/chap-annot-struct-mod.html for more information about the annotation structure modelling capabilities of the EMU-SDMS