Syllabify and Conquer: A report on preparing and analysing speech text corpora

Author: Michael Haaf ***(michael.haaf@mail.mcgill.ca)***

Supervisor: Morgan Sonderegger ***(morgan.sonderegger@mcgill.ca)***

Table of Contents

1. Introduction

Natural language is a spoken phenomena, while the most common tools for computational natural language processing are predominantly text based. Tools that represent, store, and process audio signals associated with spoken natural language have not always existed and are not trivial to implement relative to comparable text-based processing tools. The reasons for this are straightforward: text-based data is comparably smaller in storage size and less demanding in memory to process. Moreover, speech-based data have much more complicated metadata and preprocessing requirements than text-based data; many of these complications put productive research out of the reach of amateur or non-technically proficient researchers. Speech transcription, aligning transcription with speech audio, audio signal processing, and non-uniform metadata/data storage standards are just some of the complications of speech-based natural language processing compared to text-based natural language processing. Given these obstacles, it is not surprising that more tools are researched, written, and developed for text-based corpora than for speech-based corpora in both natural language processing research and industry.

While the availability of speech-based corpora and automated tools to make use of them is increasing[10], there remains a significant gap between the availability of speech data corpora and their general usability for researchers across disciplines. While libraries like NLTK[1] have bridged this gap with large accessible text-based natural language corpora and comprehensive tutorials for their usage, the same does not yet exist for speech data corpora.

This report documents work undertaken over a semester in the MCQLL Lab to perform experiments and develop software to help bridge this gap, particularly to the preparation problems of corpora storage and speech-text alignment. This project focused on the tools PolyglotDB[10] and Montreal Forced Aligner[9], which respectively address the storage and alignment problems described above. The aim of this project is to experiment and enhance these tools to extend their usability among non-expert researchers. Throughout the project, tutorials of the experiment progress were reproduced and updated.[1] Additionally, experiments using PolyglotDB to analyze formants in a publicly available data set were performed successfully and documented in this report, to be published as tutorials in the near future. Overall summary of the deliverables and results of these efforts are elaborated in Section 5.

The goal of increasing the accessibility of speech-data corpora software was addressed by performing experiments using publicly available corpora, and documenting steps taken to use these tools for facilitation of a pipeline from these public corpora to concrete linguistic analysis. These experiments, which demonstrate the ability to prepare public corpora for investigation of interesting linguistic research questions, are verified experimentally and documented in Section 3 of this report. This report constitutes a technical description of work undertaken with these tools for reproduction, reuse, and as a basis for streamlining the investigation of more advanced linguistics problems that require similar speech data preparation and analysis techniques.

In Section 4, this report additionally describes the software design approaches taken throughout the experimental process to implement a generalizable system for performing analysis on new corpora while minimizing coding changes needed. These approaches culminated in the contribution of tools written by the author and made available in a public repository[2], that are technically documented in this report for future users, experimenters and developers. Alongside PolyglotDB and Montreal Forced Aligner, these tools facilitate the generalization of the experimental procedure taken across corpora and languages beyond those covered in the report experiments. This generalization is accomplished by abstracting the differences between corpora (varied in format and language) to a set of configuration files adaptable to the requirements of large sets of known public corpora. Differences in corpora that cannot be abstracted to configuration, especially syllabic differences in syllable representation across languages, are implemented using a Strategy pattern-based approach. This approach, dubbed "syllabifying" in this report, streamlines the introduction of new client logic for new corpora to the adaptation of a testable and flexible template class.

1. https://polyglotdb.readthedocs.io/en/latest/tutorial.html
2. https://github.com/michaelhaaf/mfa-workbook

This approach aims to contribute a code-base that can dynamically adapt to and reliably implement future client requirements when new corpora necessitate novel business logic in configuration and linguistic structure.

The design approach described emerged from the set of challenging problems produced by the varied and rich nature of speech data corpora. These problems surround the automation of various steps in corpora preparation for storage, alignment, and analysis: transcript encoding, audio signal processing, and pronunciation dictionary convention generalization. These problems and their relation to the problems of speech data alignment, storage, and analysis are described in Section 2, alongside discussion of the software contributed to resolve these problems and results indicating success in resolving them.

2. Speech Data Preparation Pipeline

While speech corpora are comprised of substantially the same basic elements across corpora and languages, speech corpora are also complex and heterogeneous in a variety of their features: metadata, directory structure, annotation files, documentation, for just some examples. Dozens of formats have been used to store speech corpora over the past several decades. These variations guarantee that workflow of getting general corpora ready for linguistic analysis is not homogeneous – rather, complications introduced by these variations make manual workflows for general corpus work enormously labour-inducing. Researchers are often required to write scripts in order to make their workflow tenable.

For the research questions in this project, namely speech analysis of verb formants in spoken language across many speakers, it is assumed that an orthographic transcription of an audio file, time-aligned to the correct units of speech, is available to reliably associate sounds with known phonemes. This is not true of most publicly available speech corpora. Tools like Montreal Forced Aligner (MFA)[9] exist to force-align orthographic transcription files to their corresponding audio files. While the experiments conducted in Section 3 were already aligned at the beginning of the project (using MFA), there exist many other corpora that could be prepared for linguistic analysis in a similar manner.

Currently, preparing data for usage with MFA usually involves manual scripting and organization of files. This is because the lexicon used by MFA to perform force alignment, and the lexicons documented in publicly available corpora, are never necessarily the same format, invoking the need to transform the metadata from one format to another. This creates classic parsing disambiguation problems to determine which token corresponds to which phoneme across pronunciation lexicon formats. To resolve these dynamic complications, and to pave the way for easier and more reliable extension to new corpora in the future, this project used Emergent Design Principles[4] to adapt the generalizability of the code written as the complication of the requirements became more clear, revealing the natural application of the Strategy Pattern[7] to "syllabify" pronunciation lexicons using configuration and extendable template classes. This work is documented in Section 4.

The result is a speech data preparation pipeline from publicly available corpora to aligned speech-text corpora ready for analysis. Some examples demonstrated in this project are documented in this section. We make use of the publicly available IARPA BABEL data set, particularly the Cantonese[3] and Lithuanian[5] corpora, to demonstrate the procedure undertaken to prepare non-aligned corpora for linguistic analysis necessitating alignment.

The IARPA Babel Cantonese corpus contains scripted conversations recorded as audio files along with text file transcriptions of the conversations. Notably, the text files are not time-aligned with the .sph files, as they contain no time metadata at all. MFA is capable of time-aligning this corpora, but the file specifications for MFA are different than those for IARPA Babel; these differences are summarized in the table below.[3]

The beginning task of the preparing corpora for alignment, then, is in part to convert corpora files to formats which MFA can recognize. Since natural language processing typically relies on large sample sets to draw reliable generalizations for automated tools, we are usually interested in large corpora, so these conversions should be implemented with an eye to perform efficiently at increasing scale.

3. It should be noted that MFA does support automatic sampling frequency adjustments, and is not limited to 16kHz frequencies. 16kHz is the default, however, and we found that MFA performed better when the resampling took place in advance of the alignment.

Corpus Parameter	MFA	IARPA Babel
audio format	.wav	.sph
sampling frequency	16kHz	8kHz
transcript format	Praat[2] TextGrid	.txt

Table 1: This table summarizes some example configuration differences that occur between speech data corpora formats. Praat TextGrids are transcriptions which contain hierarchical time metadata. Additionally, this table does not include the difference in lexicon formats between the two corpora, though this is the most significant difference to adapt to – these file types and analysis are developed further on in this section.

In addition to file conversions, speech data corpora can differ significantly in their directory structures. The sample corpora chosen, the IARPA Babel Cantonese corpus, is compared below with the directory structure used by MFA.

```
IARPABabelCorpus/                      MFACorpus/
|-- scripted/                          |   pronunciation_dictionary.txt
|   |-- reference_materials/           |-- textgrid_corpus/
|   |   |   lexicon.txt                |   |    recording1.wav
|   |   |   lexicon.sub-train.txt      |   |    recording1.TextGrid
|   |-- training/                      |   |    recording2.wav
|   |   |-- audio/                     |   |    recording2.TextGrid
|   |   |   |   recording1.sph         |   |...
|   |   |   |   recording2.sph
|   |   |   |   ...
|   |   |-- transcript_roman/
|   |   |   |   recording1.txt
|   |   |   |   recording2.txt
|   |   |   |   ...
```

That is, the following steps need to be taken, while the resulting files should be stored in an MFA compatible directory structure:

- Convert generic audio files to 16kHz .wav files
- Convert .txt transcripts to .TextGrid files, with all text contained in a single time hierarchy
- Convert corpus lexicon to MFA-ready pronunciation dictionary

This pipeline prepares generic speech-data corpora with orthographic audio recordings for conversion to an MFA ready format. The following subsections deal with each step of this pipeline in turn, making reference to code used and written throughout, and containing example results within each section.

2.1 Speech Corpora Acquisition and Organization

Two corpora sources were used in this study: IARPA Babel[5][3], a US federal program to develop large-scale speech-data corpora for non English languages, as well as Librispeech test-clean[11], a large-scale collection of dictated English speech. The corpora used range from 350MB to 890MB in total size, making these studies tractable for common development machines; part of the goal of this investigation is to determine if reliable and interesting analysis can be performant using data sets this small.

In this project, the IARPA Babel corpora have not been force aligned, and the goal of this project is to align them in preparation for linguistic analysis. For the Librispeech test-clean data set, a subset of the speakers have had this alignment preparation take place at the outset of the project; with some shell scripting, these speakers were filtered into a new data set for the experiments performed in Section 3. To permit working in parallel on both tasks at the same time, this project made use of the already aligned subset of Librispeech, while preparing the two IARPA Babel corpora for analysis.

The rest of this section concerns the concrete preparation steps required to align the two IARPA Babel corpora.

2.2 Transcript to TextGrid Conversion

Generally, transcripts for audio corpora are not made available in a particular format. Most often, they are text files unadorned with particular metadata. Since force-alignment is a time-series categorization of data, however, there does need to be some representation of time in speech-text corpora for alignment to take place. One common known file specification that resolves this problem is the Praat[2] TextGrid format. TextGrids allow for structuring ‘tiers’ into a text based file, allowing for a meaningful hierarchy for storage; for example, each phone token belongs to a corresponding word, and all phone tokens are considered events which take place within a given time range.

The conversion from general transcript files to TextGrids does not exist, since metadata and formatting can vary from transcript to transcript. This project makes use of the Python library praatio[12] which has useful utilities for performing this conversion. This project contributes a script, `transcripts_to_textgrids.py`[4], which uses the praatio library to convert, in bulk, a directory of text transcript files. This approach also allows tuning the transcript file to remove unwanted tokens and to perform other useful preprocessing steps.

2.3 Audio Signal Processing

In addition to the text transcript files, the audio files of speech-text data corpora can vary in important properties. To align generic audio recordings with generic text transcriptions, there are two prerequisites: a known sampling frequency, and a known file format. As discussed in the preamble to this section, the Montreal Forced Aligner is used to perform force-alignments, which runs optimally using 16kHz sampling frequency and the .wav audio standard. While many other combinations of frequency and format are possible in principle, some standard must be applied in order for measurement to take place, in turn allowing automated alignment to take place. This section concerns the problem of transforming a large set of audio files to a common sampling frequency and a common file format standard.

For example, the IARPA corpus stores audio files using the .sph file format standard, with audio recorded at an 8kHz sample rate. This corpus therefore demonstrates a key example of a corpus needing transformations in both metrics. Conversion from this format to the MFA standard above requires scripting: while there exist many tools to convert and resample audio formats, there are none that, crucially, (1) convert arbitrary audio formats to a chosen standard (in this case, .wav); (2) resample .wav from arbitrary sampling frequency to a chosen standard (in this case, 16kHz) without altering the pitch nor speed of the audio

4. https://github.com/michaelhaaf/mfa-workbook/blob/main/scripts/transcripts_to_textgrids.py

file; (3) handle gigabytes of audio files in bulk without running into RAM issues on ordinary machines (16GB RAM).

By point of comparison, the Praat tool can do (1) and (2) but not (3), since Praat scripts, in an effort to produce a non-technical user experience, do not offer memory management or lazy/dynamic loading of audio files at runtime which would be required to fulfill requirement (3).

This project developed a method for performing all three tasks, by combining the Praat library, the sph2pipe[5] library, and intermediate shell scripting. The result is a script with user-friendly prompts and error messages which allows for specifying a directory of audio files to be converted and resampled according to specification at the command line. The shell script uses POSIX specified shell idioms (the `find -exec` command pattern) to manage running expensive memory options on large-scale corpora without running out of memory while maximizing parallelization in processing. This project contributes a script[6] which can be reused to perform similar audio signal processing on large directories in the future.

2.4 Pronunciation Dictionary Production

As discussed in a previous section, speech-text alignment matches sound to text tokenized upon phonemes. That is, an accurate alignment depends upon reliable decomposition of word pronunciations into constituent phonemes in the text layer. As such, MFA pronunciation dictionaries are two-column files, where the columns represent a one to many mapping from words to pronunciations, and where pronunciations are decomposed into syllables and constituent phonemes by token boundaries.

Speech data corpora under study in this project, such as the IARPA Babel corpus set, typically include a lexicon with a similar mapping as MFA. In the general case, however, the symbols and ordering of phonemes, and of syllabic structure, are nonstandard across corpora. A typical example follows:

5. https://github.com/burrmill/sph2pipe
6. https://github.com/michaelhaaf/mfa-workbook/blob/main/scripts/bulk_sph_resample

IARPA Babel Cantonese Lexicon	MFA Pronunciation Dictionary
`meng2 m E: N _2`	`meng2 m E:2 N`
`m4chi5 m _4 . ts i: _5`	`m4chi5 m4 ts i:5`

That is, the word/pronunciation mapping from the IARPA Babel corpus is transformed to an MFA-compatible word/pronunciation mapping for each entry in the IARPA Babel lexicon. This is the general task of the pronunciation dictionary production phase of the speech data preparation pipeline.

In the example shown above, the problem appears to reduce to simple text manipulations to a standardized character set. The problem is inherently more complicated than this: close inspection of the examples above show that the ordering of tokens cannot be strictly assumed, especially when considering tones. Without phonological theory, the ordering of tokens cannot be determined accurately. A brief summary of the necessary syllable structure assumptions follows below.

Across languages, words are composed of one or more syllables, which are in turn composed of one or more phonemes. Every syllable has a nucleus, which is itself composed of one or more phonemes. Syllable nucleus may be preceded by one or more phonemes grouped together as an onset, and nucleus may be followed by one or more phonemes grouped together as a coda. Syllables may also have tone/stress indicators, paired with the nucleus vowel.[8]

Further complications arise in determining the boundary between syllabic units. The nucleus may be the first token or the last, and it may itself be composed of more than one token. The bounds of the onset and the coda are dependent on the bounds of the nucleus. A general scheme for syllable unit determination follows:

- Coda: all phonemes that occur before the first vowel in the syllable
- Nucleus: the first vowel in the syllable, followed by
    - nothing

  - a second vowel
  - a sonorant

- Coda: all phonemes that occur after the nucleus
- Tone: see below

Tone, as indicated above, is more complicated. In this project, tone was often ignored in the early stages in order to produce results faster. However, tone (and related information for non-tonal language: stress and pitch) are often included in publicly available corpora and are useful for more accurate alignment results to disambiguate important semantic and pragmatic content.

To demonstrate variation in tone, we can consider the IARPA Lithuanian corpus, compared with the IARPA Cantonese corpus we saw previously:

IARPA Babel Lithuanian Lexicon

```
Adomaitis   a . d o: . " m a_F I . t' I s
```

MFA Pronunciation Dictionary

```
Adomaitis   a1 d o:1 m a5 I t' I1 s
```

While we saw tone was marked explicitly in the Cantonese lexicon (although with an extra underscore token relative to the MFA format), in the Lithuanian lexicon tone is implicit – this reflects that Lithuanian is not exactly a tonal language, but that there are 5 distinct stress markers which can be applied to Lithuanian syllables. These tone/stress markers are documented in the IARPA Babel Lithuanian corpus documentation.[5]

In the example above, the third syllable is: `" m a_F I`. This syllable is of immediate interest for demonstration. The `_F` character indicates a falling accent, which combined with a long nucleus (composed of the vowels `a` and `I`), indicates that this syllable is a long vowel stressed with a falling accent, which according to the IARPA Babel Lithuanian corpus documentation, corresponds to the discrete tone category 5. This is just one example demonstrative of just some of the variation in complexity for tone and syllabic unit determination.

While the structure of syllables is finite and tractable in scope, the variation in possible ordering and boundaries of phoneme tokens raises complications in simply reading tokens left to right to assign onset, nucleus, coda, and tone indicators. The software design approach to "syllabifying", that is, translating pronunciations stored as text formats into data structures representing the phonetic structure of syllables, is described in Section 4: Software Design.

The steps to produce a pronunciation dictionary for unifying the corpus formats (IARPA Babel and MFA) automatically, with potential easy extension for future corpora, is the primary contribution of the software written for this project. While more complicated than anticipated, these efforts were ultimately successful: the software reliably creates MFA-ready pronunciation dictionaries for the corpora under study (IARPA Lithuanian and IARPA Babel) from their input lexicons.

Ultimately, the pronunciation dictionaries we created now allow us to use MFA to produced aligned speech-text corpora.

2.5 Alignment Production

The goal of the force-alignment procedure is demonstrated by Figures 1 and 2. Particularly, phonemes should be visibly matched with the correct corresponding audio signals. While there are automatic tools to detect out of vocabulary words and statistical measurements to the quality of alignment, a representative sample of alignments should be manually checked (and listened to) in order to verify that the force-alignment was carried out correctly.

In this project, the entire Cantonese IARPA set was aligned, and the Lithuanian set was aligned as well pending verification. For the Cantonese data set, out of roughly 16000 speaker files, there were roughly 650 files whose alignments failed, leaving just over 15000 which were aligned successfully according to available metrics and visual inspection of a

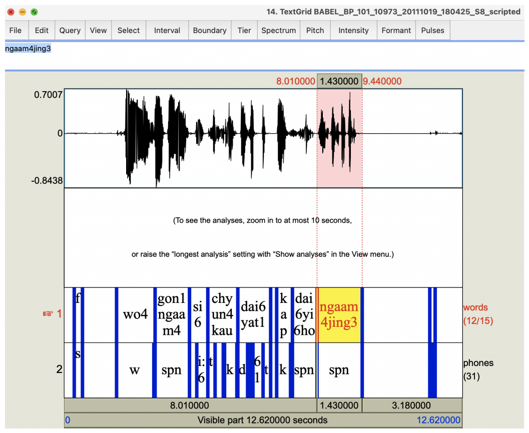
14. TextGrid BABEL_BP_101_10973_20111019_180425_S8_scripted
File Edit Query View Select Interval Boundary Tier Spectrum Pitch Intensity Formant Pulses
ngaam4jing3
words (12/15)
phones (31)
Visible part 12.620000 seconds

Figure 1: An unsuccessful forced-alignment of a 12 second scripted recording

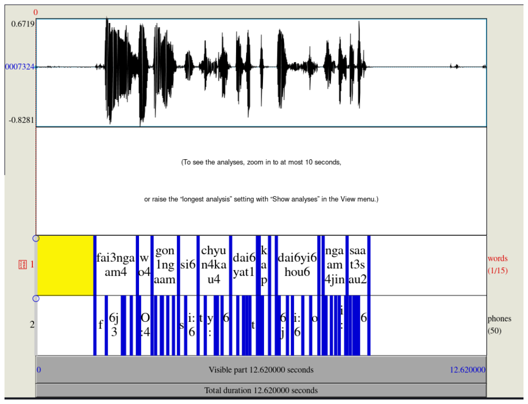
words (1/15)
phones (50)
Visible part 12.620000 seconds
Total duration 12.620000 seconds

Figure 2: A successful forced-alignment of the same 12 second scripted recording.

subset of the data. This is important: losing more data to alignment errors prevents us from building a significant enough data set using which we can perform experiments.

3. Experiments

The previous section established a pipeline for turning publicly available corpora into aligned speech-text data sets ready for linguistic analysis. In parallel to producing this pipeline and applying it to the IARPA Babel Cantonese and Lithuanian corpora, this project also sought to reproduce basic linguistic analysis on aligned corpora to experiment with and extend the usability of the PolyglotDB platform.

This section outlines the process for taking aligned speech-text data sets. These data sets could be the aligned data sets produced in the pipeline (IARPA Cantonese and Lithuanian). In this case, the Librispeech test-clean data set, which had already been aligned prior to the experiment phase of this project, was used for these experiments. The process for acquiring the Librispeech test-clean aligned corpora is described in Section 2.1: Speech Corpora Acquisition. The choice of this corpus both permitted parallel work on the data preparation pipeline, and facilitated easier comprehension of the linguistic analysis for general audiences, including the author himself.

This section begins with reproductions of existing PolyglotDB tutorials to demonstrate the basic mechanics of working with aligned corpora. It is followed by concrete investigation into basic linguistic research questions about the data set: for example, does the data set exhibit formant mappings typical of similar English speaker corpora? The production of these experiments, and discussion of their results, points the way to future more advanced linguistic analysis on these data sets and others that could be prepared by the speech data set pipeline discussed previously.

3.1 PolyglotDB Package Management

Before we move to the experiments performed, a quick note can be made about the development environment required to use these packages. PolyglotDB is a python package available for installation via pip, however, it does contain core dependencies that live outside of the python ecosystem; in particular, the NoSQL Java database library, InfluxDB. These dependencies can be varied in their support by general developer operating system platforms, and prior to this project, there was no one process that would automatically resolve these dependencies for a given PolyglotDB installation.

This project contributes a small procedure to automatically resolve system dependencies on all platforms that support Conda. The operation of this procedure is now described on the PolyglotDB website.[7]

This project investigated a transition of the PolyglotDB platform from pip to modern python installation environments, such as condaforge[6], where Montreal Forced Aligner is already available. Time constraints prevented this transition from being completed by the end of the semester, but progress is ongoing.[8]

Condaforge support for PolyglotDB would allow for automatic management of python and Java dependencies on any machine that supports conda. Future work could investigate

7. https://polyglotdb.readthedocs.io/en/latest/getting_started.html#conda-development-environment
8. https://github.com/conda-forge/staged-recipes/pull/18313

installation of PolyglotDB on new server architectures, such as the increasingly available ARM64 platform. For the time being, the procedure now provided on the PolyglotDB website linked in the footnote above is the procedure taken for the following experiments, and should be reproducible on most machines just like an automatic Conda installation.

3.2 Reproduction of Existing Experiments

The problem of abstracting away manual scripting from performing linguistic analysis on an aligned speech-text corpora is addressed by PolyglotDB. To begin analysis, the corpora should be stored in a manner that allows reliable read/write querying for corpus investigation and enrichment. To facilitate this storage, this project followed existing PolyglotDB tutorials.[9] This section overviews this process and summarizes the results.

As described in the Data Acquisition section of this report, a forced-aligned Librispeech test-clean English data set was acquired from previous alignment efforts at MCQLL with basic shell scripting. While this data set is force-aligned with phonemes in time, there is no other interesting metadata associated with speakers. Associating speaker metadata (age, gender, residence, etc.) with speech data is an important component of speech data research. This section described the procedure for storing aligned data sets, and enriching them for linguistic analysis, by importing them into a PolyglotDB instance.

PolyglotDB works by creating NoSQL time-series database in its development environment for speech data, and uses a SQL python library to wrap queries to speaker data stored in text files. Speech corpora can be imported and queried, with read/write permissions, using NoSQL under the hood, with queries wrapped in a python library for ease in interfacing with scripts and other software. Therefore, importing and enriching a corpora in a PolyglotDB database, then exporting the enriched corpora for analysis, can be done in a few lines of python, as shown in the tutorial. This tutorial was reproduced, with some typical examples of the kinds of queries that PolyglotDB supports.

9. https://polyglotdb.readthedocs.io/en/latest/tutorial.html

Now that it has been established we can import, enrich, and make simple queries to a PolyglotDB instance, more interesting linguistic analysis can be performed.

3.3 Formant Analysis

Following the analysis preparation performed in the previous steps, in this section we process the output of the same Polyglot database we enriched, and show how it can be used to address basic linguistics research questions involving variation between speakers in vowel space.[10] For example, we could verify basic aggregate properties of our data set by comparing vowel formant plots across speakers in our data set, or apply statistical analysis to investigate more interesting questions about properties of speaker dialects.

In this project, we carried out two such experiments: one to verify an expected property (do male speakers, on average, have lower vowel frequency utterances than female speakers?), and another to answer an interesting question about the speakers' dialects (which speakers exhibit the "cot"–"caught" merger in their dialect?). This section explains our approach to addressing those questions with our corpus.

We begin with the exported data from the previous step: a .csv containing vowel formant measurements for each phoneme in the data set, associated with each speaker and speaker metadata. Some debugging in this step is required: forced-alignment is an automated, probabilistic process and like signal detection in general, will not make perfect predictions. Such alignment issues can carry over and be made visible by investigating timing/speech-rate calculation errors in particular. We carried out some sanitization measures data set manually: correcting columns with invalid values, filtering duration/F1/F2/standard deviation outliers, in order to a reliable subset given noisy measurements. These steps require some domain knowledge and scripting capability, but this is the beginning of the scientific investigation, so that is to be expected.

The filtered data set was reduced to about 3/5 its original size, to 43758 measurements from 74001, increasing the coherence of the formant measurements greatly. The filtered

10. Most of the analysis in this section was adapted from analysis done by Morgan Sonderegger (McGill), which itself is analysis is partly adapted from course materials by Márton Sóskuthy (UBC).

data set was happily significant enough to produce interesting results: we were able to observe coherent distinctions in speaker frequencies by sex, and in "cot–caught" mergers by speaker dialect. These results are documented in the figures below.

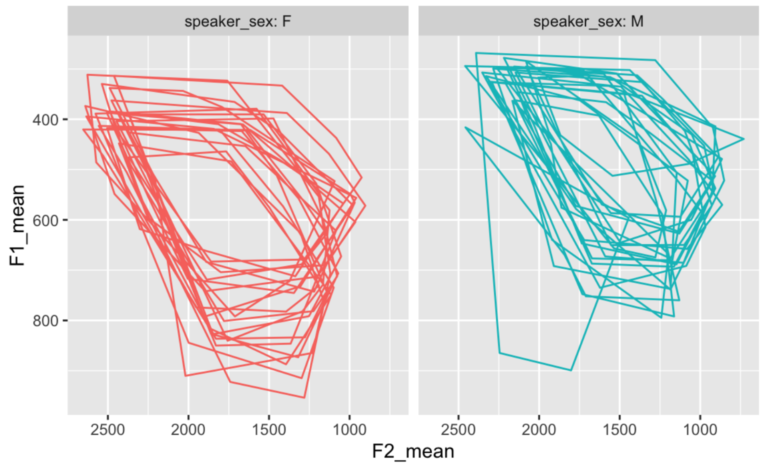

Figure 3: Comparing the vowel formant plot of male and female speakers, the vowel space is noticeably shifted to lower F2 and F1 on average for male speakers compared to female speakers, verifying an expected result. Figure produced using code written by Morgan Sonderegger (McGill).

We can see that the Polyglot database we enriched can be used to address basic linguistics research questions – and these are just the tip of the iceberg. The following section indicates areas of future work, which could be done on this corpus, or any other force-aligned speech data corpora.

3.4 Future Work

The following tasks, some taken up partially, others planned but abandoned to project scoping, would be interesting lines of research using the tools and methods established in this report:

- As indicated in a previous section, completing the setup for PolyglotDB to be made available on condaforge would more reliably support this type of research on a wider array of development architectures.

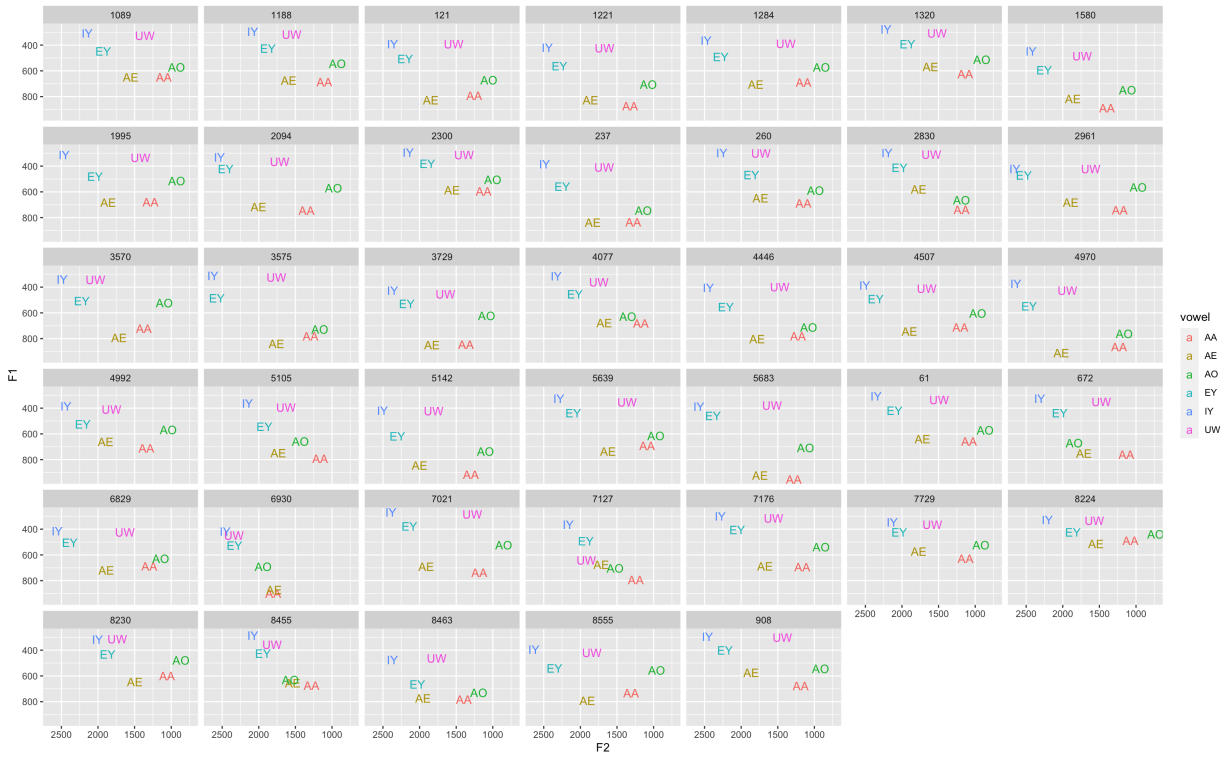

Figure 4: The “Cot–caught merger” is a well-known drift in some English dialects, particularly American English. In this graph, we can measure the vowel space distance between AO (“caught”) and AA (“cot”) to determine which speakers have the cot–caught merger in their dialect. For example, speakers such as 237, 2830, and 6829 may have AO == AA, indicating a merge, while speakers 1221, 2094, and 8555 may have distinct AO and AA. Figure produced using code written by Morgan Sonderegger (McGill).

- The IARPA Babel corpora set has many other languages documented aside from Cantonese and Lithuanian. Languages such as Mixtec, Totonac, Yoruba, and Burmese in the IARPA Babel set is also more easily available for this type of research following the methods developed in this project.

- The Formant analysis performed in the previous section is just one example of an interesting linguistics question to investigate with corpora like these. Previous work has been done to demonstrate Pitch analysis[11] using PolyglotDB, which provides further interesting data for analysis. These basic measurements open a wide range of research questions for amateur and expert researchers alike.

11. https://memcauliffe.com/playing-around-with-polyglotdb.html

- Relatively more advanced linguistic analysis research could be performed with data sets like these, or even more general speech aligned data sets. One such question of particular interest to the author would be investigation into time-differentiated data sets to measure dialect change in time.

- A bit more speculative, but is certainly more room for automated software tools and paradigms for speech data sets. Word-vector analysis has seen incredible results in text-data natural language processing, and a similar paradigm applied to general speech-data corpora could see similar results.

4. Software Design

This section is a technical description of the software contributed to this project. The set of requirements covered in previous sections constitute a dynamic and varied set of requirements for such a system that can generalize to future corpora. These complications necessitate software design. The code written for this project made use of practices and principles like the Emergent Design Principle [4]. These practices and principles are important in development generally, given that software requirements are always in a state of change, but are particularly important to this project where a major part of the utility of the software is to save time adapting its purposes to new corpora and new languages.

This section will explain where relevant design principles were applied to the code base to aid future users and developers in generalizing the software to new contexts. This section will be of particular interest to developers who wish to extend the functionality of the software contributed, for example, to extend the number of languages supported for alignment preparation.

We saw in previous sections that speech data corpora vary in format. Character bounds between words, pronunciations, syllables, and phonemes can vary, in fact even between corpora published by the same author (we observed this with the IARPA Lithuanian and IARPA Cantonese corpora). Furthermore, various properties of languages can vary in predictable ways: languages will have different sets of characters for important lexical properties like vowels, sonorants, and tone markers. These sets of variations can all be encapsulated in a single data class, as shown in the figure below:

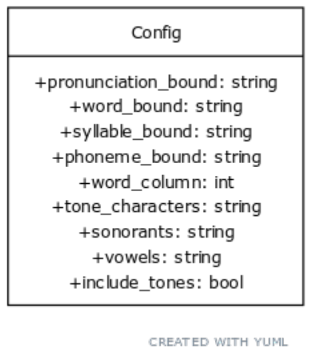

Figure 5: A UML diagram of the Config class used to abstract common features that vary across corpora in the syllabify system.

Configurations are implemented as `.yaml` files in this project. The configuration for the IARPA Cantonese corpus, for instance follows:

```
pronunciation_bound: "\t"
word_bound: ""
syllable_bound: " . "
phoneme_bound: " "
word_column: 1
tone_characters: ‘_[0-5]’
sonorants: ""
vowels: "aeoiu69"
include_tones: true
```

These configuration files allow for variation in high level abstract properties of corpora without the need to known anything about the underlying implementation of the code. This is useful for two reasons: to prevent new code from needing to be written for new corpora, and to make the addition of new corpora easier for researchers who are not familiar with programming themselves.

Variation between corpora does not end, however, with changes in the format of the file nor simple properties of languages. We also saw in previous sections, particularly the Pronunciation Dictionary Production section, that variation between languages and corpora is also observed in determining syllabic units for data representation. These variations are high-level abstractions themselves, more complicated than in-order processing: for one example, there is no inherent ordering to where tone makers are placed, as seen with the IARPA Cantonese and IARPA Lithuanian corpora. The method for which the syllabic units nucleus, coda, onset and tone are determined from language necessitates higher-order logic than configurational variation.

One way we can consider this problem is that there is a single behavior, "syllabify", that is common across all corpora. This syllabify behavior should varying in implementation between corpora, but across corpora should have a common interface (a word/pronunciation pair) and reliably decompose the pronunciation into syllabic units so that these units can be used to produce uniform output. Moreover, we would like to decouple clients of this behavior from any one implementation – at runtime, clients should be able to specify which implementation of syllabify they need (dependent on the corpus chose) without having to change any code. Finally, developers of the software should not have to change anything about the syllabify interface, common across all corpora, in order to add new implementations.

These requirements naturally evoke the Strategy pattern[7], a classic behavioral pattern for allowing variation in algorithm to be encapsulated by a generic Strategy class.

The design is encapsulated and extensible because "the various versions of the behavior, how many variations there are, which variation will be used under a given circumstance, and the design of each varying implementation are all hidden from the client"[4]. That is, the varying corpora language properties and format distinctions for speech-data experiments can be abstracted to algorithms that are variable at run-time, decoupled from the generic behavior required to "syllabify", and easily extensible to development in the future.

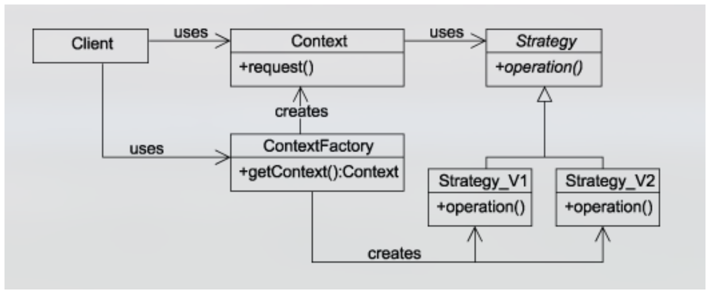

Figure 6: The variation of the Strategy Pattern used in this project, with creational logic is encapsulated in a factory class. Generic classnames shown here (e.g. Strategy) are specified precisely in Figure 7. Source: Figure B.54 in Bain, 2008[4]

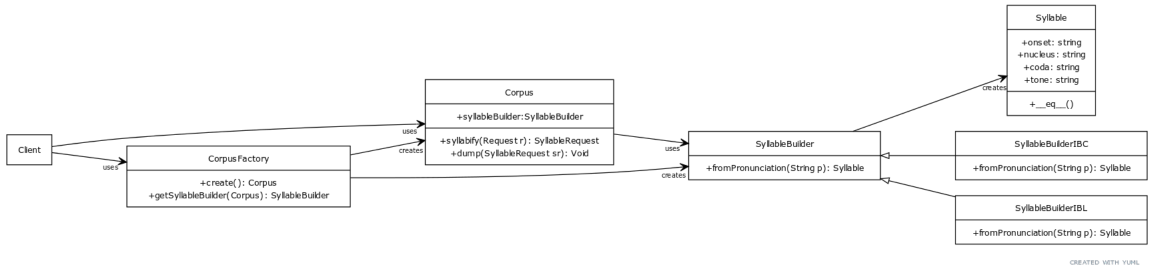
Syllable
+onset: string
+nucleus: string
+coda: string
+tone: string
+__eq__()
creates
Corpus
+syllableBuilder:SyllableBuilder
+syllabify(Request r): SyllableRequest
+dump(SyllableRequest sr): Void
uses
creates
Client
uses
CorpusFactory
+create(): Corpus
+getSyllableBuilder(Corpus): SyllableBuilder
uses
creates
SyllableBuilder
+fromPronunciation(String p): Syllable
SyllableBuilderIBC
+fromPronunciation(String p): Syllable
SyllableBuilderIBL
+fromPronunciation(String p): Syllable
CREATED WITH YUML

Figure 7: The actual pattern implemented for the syllabify system developed for this project, with some classes redacted to make the pattern clearer. See the Appendix, Section 6.1 for the full UML diagram. The SyllableBuilder classes stand in as the variable strategies: adding linguistic support for building syllables for a new corpora only involves creating another SyllableBuilder subclass, without modifying any part of the rest of the system.

Further details about design choices are elaborated in a complete UML diagram of the syllabify system in the Appendix, Section 6.1.

5. Conclusions

This report outlined some novel complications introduced by the adaptation of speech-text corpora varying across language and formatting conventions for concrete linguistic analysis. While tools exist for the preparation problems of corpora storage and speech-text alignment, there remains manual scripting involved to adequately create a development environment for interesting linguistic research. This project developed a procedure, applied to two corpora of varying format and language, which enabled automation in the preparation of speech corpora for forced-alignment, as well as the procedure for performing basic linguistic analysis on aligned speech corpora.

This procedure surrounds the automation of various steps in corpora preparation: corpora acquisition and normalization, transcript encoding, audio signal processing, and lexicon convention generalization. While many of these variations can be abstracted to configuration, the problem of parsing lexicon files to create phonetically valid data representations for conversion between formats proved to be higher-order abstraction, requiring designed software to perform reliably across variation in linguistic features. The software contributed in this report aims to fill just that gap: to provide automation general enough to extend to corpora varying in language and format, to further the accessibility of research on corpora requiring alignment.

The required generalization is accomplished by abstracting differences between corpora to both configuration and to a Strategy pattern-based python library. Using this approach, new corpora can be supported by modifying a template configuration file and adding a single subclass, rather than modifying existing scripts or code. The software developed to perform this task is made available for study, use, and extension in a public repository.

The upshot of corpora preparation was also analysed in this report. Experiments demonstrating the usage of PolyglotDB with similar corpora verified known experimental results, namely regarding: (1) formant frequency measurements aggregated across a large sample of speakers, and (2) dialect feature detection through measurement of variation in pronunciation across a large sample of speakers. These experiments show that investigation into interesting linguistic questions can be performed on publically available corpora in general, and the procedure for preparation automated across corpora is possible with existing speech-text software tools.

Some of the aims set at the outset of the project were not reached: the procedure developed here invites investigation into even more profound linguistic questions that can be studied through publicly available corpora beyond Librispeech. This report outlined some possible avenues for future work, as well as some suggested improvements to the software contributed in this project.

These are humble and imperfect contributions, but hopefully they can play a role in the extension of engaging speech data corpora research.

5.1 Acknowledgments

The completion of this report would have been impossible without the support of my supervisor Morgan, my advisor Ann, my mentor Jasmine, my partner Maïa, and many cherished family, colleagues, comrades, teachers, and friends who are invaluable parts of my life and have uplifted me in innumerable ways along the long way.

6. Appendix

6.1 Syllabify System UML Diagram

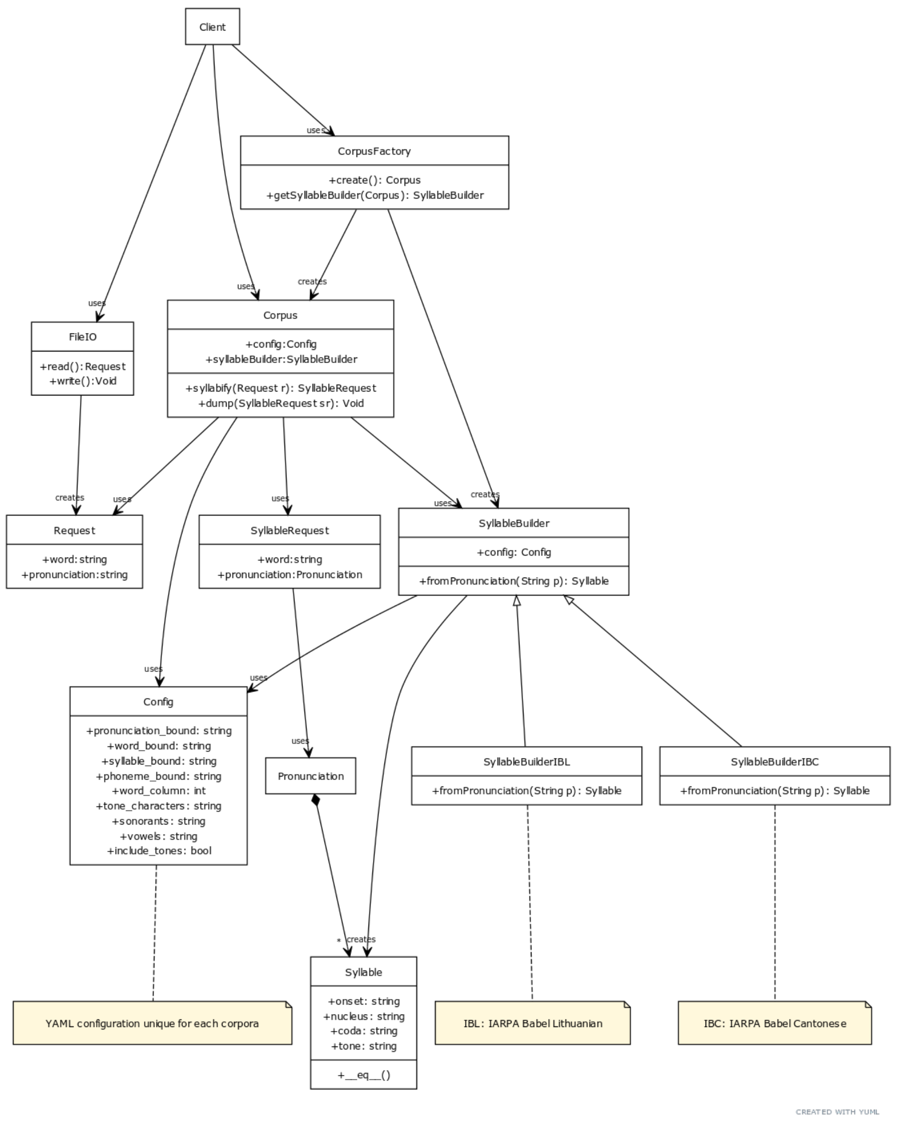

Figure 8: The complete UML diagram for the syllabification system produced during this project. The Strategy Pattern components can be seen in the Corpus/SyllableBuilder/Client relationships: the client can choose syllabification at runtime using configuration (for elements common to all languages) and dynamic algorithmic support (for elements unique to each language requiring higher-order logic) provided by the pattern.

References

[1] NLTK :: Natural Language Toolkit, 2022. URL https://www.nltk.org/.

[2] Praat: doing Phonetics by Computer, 2022. URL https://www.fon.hum.uva.nl/praat/.

[3] Tony Andrus and et al. IARPA Babel Cantonese Language Pack IARPA-babel101b-v0.4c LDC2016S02, 2016. URL https://catalog.ldc.upenn.edu/LDC2016S02. Web Download.

[4] Scott L. Bain. *Emergent design : the evolutionary nature of professional software development*. Net Objectives product development series. Addison-Wesley, Upper Saddle River, N.J., 2008. ISBN 978-0-321-55372-0 0-321-55372-1 81-317-3606-7 978-81-317-3606-7. URL http://www.myilibrary.com?id=264869.

[5] Daniel Benowitz, et al. and et al. IARPA Babel Lithuanian Language Pack IARPA-babel304b-v1.0b LDC2019S03, 2019. URL https://catalog.ldc.upenn.edu/LDC2019S03. Web Download.

[6] conda-forge community. The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem, July 2015. URL https://doi.org/10.5281/zenodo.4774216.

[7] Eric 1965 Freeman, Elisabeth. Robson, Kathy. Sierra, and Bert. Bates. *Head First design patterns*. Head first series. O'Reilly, Sebastopol, CA, 1st ed. edition, 2004. ISBN 0-596-00712-4 978-0-596-00712-6. URL http://catdir.loc.gov/catdir/enhancements/fy1305/2005280819-t.html. Section: xxxvi, 638 pages : illustrations ; 24 cm.

[8] Victoria Fromkin, Robert Rodman, and Nina Hyams. *An introduction to language*. Wadsworth/Cengage Learning, Boston, MA, 3e. edition, 2006.

[9] Michael McAuliffe. Montreal Forced Aligner 2.0.0 documentation, 2022. URL `https://montreal-forced-aligner.readthedocs.io/en/latest/index.html`.

[10] Michael McAuliffe, Elias Stengel-Eskin, Michaela Socolof, and Morgan Sonderegger. Polyglot and Speech Corpus Tools: A System for Representing, Integrating, and Querying Speech Corpora. In *INTERSPEECH*, 2017. doi: 10.21437/INTERSPEECH.2017-1390.

[11] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, South Brisbane, Queensland, Australia, April 2015. IEEE. ISBN 978-1-4673-6997-8. doi: 10.1109/ICASSP.2015.7178964. URL `http://ieeexplore.ieee.org/document/7178964/`.

[12] Tim Mahrt. PraatIO, 2016. URL `https://github.com/timmahrt/praatIO`.