‘superassp’ and the need for a sparse signal track data format

Fredrik Karlsson
Department of clinical science & Humlab, Umeå University

Abstract

The digital processing of phonetic data has long been based on the same basic data structures and consequently been stored in essentially the same manner. The transition to structured database-based workflows and the development of more complex analyses challenge the established data structures and storage models. I propose a sparsely defined data track model for structured storing of locally define signal information and suggest an implementation. The use of sparse data created by the amalgamating signal processing library ‘superassp’ is discussed within a speech database management system.

Background

The digital storage of information for use in phonetic research has remained stable for many years now, and this paper will develop the rationale for extending the existing models with an additional storage type, in order to afford consistant management of speech recordings and derived signals.

With time-aligned signal tracks being the primary format used for storing recordings obtained directly from a speaker or computed as indirect tracks, possibly by windowing of the original signal. While time-aligned, the sampled or derived signal tracks may certainly be multidimensional in nature, but the values are predominately of the same origin in terms of the analysis used. That is, while a spectrum is multimensional, each value shares the same origin, and we may define them continuously throughout a signal and store them as a spectrogram without considering how to label the individual values. This property may not hold in all cases in the future, as we will discuss later on.

Of further interest to us is how the signal is sampled. The sampling of the speech signal at a uniform rate is the predominant model for aligning measurements against the timeline, and while variable rate would be feasible, it could be argued that such an extension would offer little advantage for signal track data. We take the storage of electroglottography data as an illustrative example. Electroglottography is the investigation of the electrical impedance of the larynx as an indirect measure of vocal fold contact area. The impedance is sampled continuously, but of interest is, however, the dynamics of vocal fold closing and opening within a full cycle. The frequency at which the vocal folds describe an entire cycle varies over time when speaking, and the time frame for which derived measures (e.g., the closed quotient) offer a valid description of the activity will therefore vary in length. If sampled at a modern sampling rate, however (44100 Hz or above), the time resolution of glottal events will be > 70 samples per glottal cycle even for speakers with a high (600 Hz) f0. Since the output of analysis for a glottal cycle is relatively simple it may still be efficiently stored as identical values repeated throughout all analysis frames within the glottal cycle. An electropalatography signal may similarly be sampled relatively infrequently, but resampled to fit the timeline of an audio recording of speech since the number of stored channels is reasonably small and the sampling rate is known. It is observed, therefore, that variable rate low-frequency information may be shoehorned into a continuous track model without much loss in efficiency, provided that the data is relatively simple and analysis windows do not overlap.

Speech information with potential overlap and no periodicity

It has long been recognised that storage of information attached to the speech signal by a human as an annotation in some form requires a storage form that is separate from signal track. The reasons for differentiating between these two information forms are of particular interest for the argument developed in this paper. First, it is observed that the need for a separate data format for human added annotation is driven by the fact that the timing of where the response is started to be defined has no periodicity, and further that the portion of the timeline of a recording for which the where the response is valid is highly variable. In short, we can rarely predict exactly where a human would like to add a note related to their perception of a signal, and exactly where the perception ends. While it would be possible to encode the human perceptions as being present in a portion of a continuously defined signal track, it has long been recognized that a single start end end point noted in a text file along with a textual note (label) is a more effient encoding.

Second, human attached information may very well partially or completely overlap in relation to the recording session timeline. While one could have opted to find the solution to the problem of storing this information also in variable-rate signal track formats and multiple tracks (or multiple signal files), such an implementation would have been wasteful, and the solution has instead been the storage of human transcription data as time slices with a text label that is placed in different tiers in a collection.

One could note that the results of human perceptual experiments that use part of a speech signal as stimulus could be also stored in the same format, as they could be aligned with the timeline as soon as one could compute a time of a response. However, as perceptual experiment captures responses of many participants that likely will overlap, separate tiers will have to be constructed for to store participants' responses, which would make management unwieldy. It is observed that while new types of speech information may be shoehorned into one of the well established storage models within a database context, we are likely better served by acknowledging the lack of fit and reconsider our thinking of the data we store.

Sparsely defined signals and when we may need them

We have seen that we currently possess a data format for working efficiently with regularly defined signals, which may be multidimensional but remain sequential in their arrangement, and different formats for storing sparsly defined or data with overlap, but which handled multidimensionality less flexibly. I will illustrate that developing analysis techniques may result in data that, while possible to shoehorn into the best fitting of these two models, may demand a separate mode of storage when striving to take advantage of the possibility of structured and repeatable analysis offered by speech database management systems.

The VoiceAnalysisToolbox is an analysis package aimed at establishing possible acoustic markers of voice production deterioration due to a disease. It takes a single prolonged vowel as an input and computes 339 acoustic measures, including variants of jitter/shimmer, MFCCs, and their first and second-order derivatives, summarises a wavelet decomposition of the pitch track, computes closed quotient of the vocal fold cycle from inverse filtering, and so on. The analysis is implemented in compiled Matlab code and takes from 10 to 60 times the duration of the signal to complete. We have no evidence that the output of the procedure is identical across all samples of the same speaker, so we may assume that a well-defined research project would want to analyse either multiple prolonged vowels for the same speaker, multiple portions of the prolonged vowel in, or both.

So, after having up to 3 minutes to complete an analysis of a single prolonged vowel, where do we store the output so that it may be retrieved efficiently from a database later on? Having only 339 output values, the output itself could relatively easily be stored in manner similar to a spectrogram, but as the analysed regions may both vary in size and overlap, the management of the unpredictably many signal files may prevent efficient use of the information later on. Further, as the information that requires storage may me of different natures (e.g. MFCCs and wavelet features mixed) a labelling strategy may be required which signal files usually do not have. Therefore, the user will be burdened with the additional task of carefully indexing when retrieving the required information.

Similarly, the storage model that we now use for holding transcriptions is not well suited for storing the data. In their current form, files that we now use for holding human annotations (transcriptions) associated with a speech signal are not well suited for holding arrays of 339 values. Further, the serialization of values into a form that travels well between programming languages while remaining efficient becomes an issue, as does the task of making sure that the specifications of data stores and considered comparable actually does contain the same order columns and in the same order.

Listing 1. The definition of a table that holds slices. For each slice, the start and end sample of the slice needs to be defined, along with a checksum of the file content ('hash). The columns holding measurements to be stored for the slice are indicated as "[...]". The sample rate of the signal file is also required, which ensures that the sample number can be converted to time if needed.

```
CREATE TABLE slices (
    `start_sample` INTEGER NOT NULL,
    `end_sample` INTEGER NOT NULL,
    `samplerate` INTEGER NOT NULL,
    `hash` TEXT NOT NULL,
    […]
    PRIMARY KEY (start_sample, end_sample, sha)
);
```

The implementation of sparse signals superassp

'superassp' is an R speech signal processing library aimed at bringing together algorithm implementations of various sources and mold them so that the output could be used consistently by a speech database management system. It was originally contrived as an extension of the wrassp R package for use within the EMU Speech Database Management System (Emu). The `superassp` package aims to bring together a very heterogeneous collection of signal processing algorithms implemented in R (wrassp; Bombien, Winkelmann, Scheffers, 2021), C, Praat, Matlab™[1] and Python under a common interface. Regardless of origin, most algorithms will be made to produce an SSFF track when repackaged for use within superassp. For the output of the VoiceAnalysisToolbox procedure, however, a sparse signal track format is implemented in a way that deliberately makes it incompatible with the SSFF format and precludes direct use within the Emu system. Instead, the sparse track format is implemented as an SQLite database file. The choice of this database file format is arbitrary, but it may be observed that it is arguably the most used database, easily accessible across platforms and programming languages, and is reasonably performant.

In the SQLite format, the sparse collection of possibly overlapping slices for which data needs to be stored receives the simple definition presented in Listing 1. The simple definition allows for one sparse slice file to hold multiple slices with the same specifications (that is, the same set of measures). Identically structured measurements from partially overlapping slices may be inserted into the same sparse slice file, but the slices are uniquely defined by their start and end sample and the content of the file it was computed from, and duplicate definitions cannot be inserted. Instead, the application needs to determine at insert time how this situation should be handled. It is likely that most insert operations will be (SQL) `INSERT OR REPLACE` statements so that revised measurements will overwrite previously stored information for the slice.

It should be noted that, however, that slices are unique only for particular file content. Thus, it is possible to store the result of multiple assessments of the same portion of a signal timeline but with altered signal content (such as a preceding filtering operation) in the same sparse slice file, which may prove convenient in studies of algorithm robustness.

The use of sparse signals in speech databases

An implementation of sparse signals is, to my knowledge, not available in a speech database system currently. As 'superassp' is being developed as a complementary library for use in the Emu SDMS (Winkelmann, Harrington, & Jänsch, 2017), we will consider some aspects of sparse signals in that context. Emu has a client-server architecture, with the transcription interface being implemented as a web application and transcriptions transferred as JSON files. Since sparse signals are found above to be share properties with transcription elements in terms of time specification while demanding higher dimensionality, the mode of transfer and visualization of the data to the user should be considered. In Listing 2, a tentative JSON format for transferring a sparse signal containing parts of a voice report for two portions of the speech signal to the client is illustrated. How the data should be presented is, of course, up to the developer of the receiving client. In the context of the Emu SDMS, one could note that 2D panel of the web client could be set up to show the data efficiently to the user. In the case of overlapping slices, such as in Listing 2, multiple colors may

[1] We are currently investigating how we can expose algorithms implemented in Matlab™ within our license agreement and will do so if possible.

be used to differentiate between the first and additional slices. As the user moves the cursor into the prolonged vowel, the data for the larger slice is shown intially, and as the user moves the cursor into the region also covered by the second slice, additional and differently colored data points may be injected into the display. The 2D panel of the Emu web client is already well equipped to display just some columns of a specification and leave some undisplayed. Therefore, we see that just a portion of a slice, such as some MFCCs that are particular interest for the analys, may be displayed to the user when available in an efficient manner.

Listing 2. An example of selected measures of two temporally overlapping voice reports in JSON format for transfer between applications.

```
[
  {
    "start_sample": 44100,
    "end_sample": 132300,
    "data": {
      "Jitter (local)": 0.401,
      "Jitter (rap)": 0.171,
      "Jitter (ppq5)": 0.209,
      "Jitter (ddp)": 0.514,
      "Shimmer (local)": 2.131,
      "Shimmer (apq3)": 1.028,
      "Shimmer (apq5)": 1.347,
      "Shimmer (dda)": 3.084
    }
  },
  {
    "start_sample": 66150,
    "end_sample": 110250,
    "data": {
      "Jitter (local)": 0.469,
      "Jitter (rap)": 1.207,
      "Jitter (ppq5)": 0.233,
      "Jitter (ddp)": 0.624,
      "Shimmer (local)": 2.341,
      "Shimmer (apq3)": 1.207,
      "Shimmer (apq5)": 1.456,
      "Shimmer (dda)": 3.622
    }
  }
]
```

Conclusion

I have argued that acoustic data that are sparsely defined, potentially overlapping in time of definitiona but which may be multidimensional in nature, do not confidently fit in the data formats traditionally used in speech research and in speech databases. I have illustrated how they might be thought of in storage by presenting the implementation considered for the 'superassp' library. While visualization of exessivelly overlapping sparsely defined multidimensional data to the user will likely offer several challenges, I argue that a meaningful presentation is feasible for at least a subset of the data, depending on the fascilities made available by the receiving client software.

Acknowledgments

The research presented here has been enabled by the Swedish national research infrastructure Språkbanken and Swe-Clarin, funded jointly by the Swedish Research Council (2018–2024, contract 2017-00626) and the 10 participating partner institutions.

References

Bombien L, Winkelmann R, Scheffers M (2021). *wrassp: an R wrapper to the ASSP Library*. R package version 1.0.0.

Winkelmann, R., Harrington, J., & Jänsch, K. (2017). EMU-SDMS: Advanced speech database management and analysis in R. *Computer Speech & Language*, *45*(Supplement C), 392–410. doi: 10.1016/j.csl.2017.01.002

Tsanas, A. (2012). *Accurate telemonitoring of Parkinson's disease symptom severity using nonlinear speech signal processing and statistical machine learning*. Doctoral dissertation (DPhil), University of Oxford.