Interaction Style Modeling Toolset

an extension of the Midlevel Prosodic Features Toolikit

Version 1.0

Nigel Ward, University of Texas at El Paso

August 6, 2020

1 Overview

This toolset supports the analysis of interaction styles in spoken dialog. To quote from the abstract of my as-yet-published paper:

> In spoken dialog, people clearly vary in their interaction styles, but a comprehensive model of the space of variation has been lacking. To address this need, I applied Principal Component Analysis, using features designed to capture aspects of interaction-related prosodic behaviors, to tens of thousands of conversation fragments from the Switchboard corpus of American English telephone speech.

This toolset has so far been used only for the analyses reported in the paper, but is ddesigned to generally support computational analyses of interaction styles, for both scientific and practical purposes, as discussed in the paper.

Specifically, this code enables one to:

- derive a vector-space representation of interaction styles from a corpus of stereo-recorded spoken dialogs.
- given any new dialog, compute where it lies in an existing space of interaction styles.
- given a corpus with metadata (for now just Switchboard), compute various correlations and statistics on the factors affecting interaction styles .

This document serves mostly to overview the workflow and name the specific Matlab functions to call. It should be read after getting the big picture from the paper, and can be followed by reading the comments in the code. It is a work in progress; comments and suggestions are welcome.

2 Getting the Code

This code was written in Matlab and runs on Matlab version 2019.

The code is at `https://github.com/nigelgward/TBD` . It requires the Midlevel Toolkit, which is available at `https://github.com/nigelgward/midlevel/` , complete with documentation.

3 Terminology

The term "dimension" in the comments may refer either to a prosodic construction dimension or to an interaction style dimensions.

4 Corpus Preparation

Find a disk with 60 GB available, create a directory, refered to below as `motherDirectory`, and copy the Switchboard data discs there one by one, naming the subdirectories `disc1` ... `disc4`.

Create `wav`-format copies of each audio file using `code/sph-to-wav.sh` .

Create the pitch files using `midlevel/src/reaperize.sh` .

Decide how to split the data into subsets, and in *splits/* create an index file for each. How I did this for Switchboard is described in `labnotes.txt`. Each index file contains a one-per-line listing of the audio files in that set.

Run `prepMetadata.sh`.

5 Building the Model

Open Matlab and change to the `code` subdirectory (since that's where `rotationspec.mat` is found).

Run `computeStyleParams.m('splits/trainset.txt', 'trainStats.csv')` to create the features for all files in the training set. The output, here `trainStats.csv`, is sometimes called a PCB file, as it contains features based on prosodic-construction bin counts.

This will time (12 hours on a 8GB 3.6GHz machine), so if it crashes, be prepared to look at `trainStats.csv` to see how far it got, and edit the code to just process the conversations not already processed; the function will conveniently append stats for those new conversations.

The `computeStyleParams` function has three hidden dependencies, in which it reads data from hardcoded filenames. The first two, `fsfile` and `rsfile` specify how to compute the prosodic dimensions, as described in the midlevel documentation. The third is `sifile`, listing the standard deviations of the prosodic dimensions, as derived from the pbook data, and is used to determine the ranges of the bins. Usually this should be held constant for all analyses, unless you want to define not only a new interaction space but also to redefine the basic features used.

Now change to the `stats` subdirectory and run `deriveISspace('trainStats.csv', true, 'trainIsNormRot.mat', 'trainIStyles')` (or `'train-30sec.csv'`). This will create the model and save its parameters in the `mat` file. It will also print out many interesting statistics that can be copied into the paper. It also pops up a figure that will go in the paper (when `compareWithSubsets` is uncommented).

This will also write several files, notably:

- `loadingsTables.txt`, written to `stats`, shows, for each dimension (each interaction-style dimension) the loadings on the behavioral features (the prosodic-constructions-binned features).
- many files with names starting with `isratios`. These can be examined, for example by `sort -n isratios1hi.txt | more`. Once computed, it's nice to save them in a safe place, for example by moving them all to `wordstats`.
- many files with names starting with `idimWords`. These can be examined with LIWC. They also should be moved to `wordstats`.

- `sox-commands.sh`, a script which can be run with `bash` to generate various audio fragments. These can then be listened to to help understand the meaning of the various dimensions. The script also creates some anchor stimuli, to use in the human-perceptions experiment. Once created, it's nice to move these all somewhere safe, e.g. `exemplars`.

6 Applying a Model to New Data

This is to be done, for example, when choosing clips to use for the experiment, which should, of course, be taken from data not used in training or interpreting the dimensions: thus from held out testset data.

In `code` run `computeStyleParams.m('splits/testset2.txt', 'test2Stats.csv')`. This will take 3–4 hours.

** not tested yet **

In `stats` run `deriveISspace('test2Stats.csv', false, 'trainIsNormRot.mat', 'testIStyles')`, importantly specifying `false` for the second parameter.

This will not change the model, but it will apply it to the testset data, most usefully generating a `sox-commands.sh` file to use to generate fragments to use as stimuli, and the predicted scores to compare to the turker-assigned stores.

7 Acknowledgments