Interaction Style Modeling Toolset

an extension of the Midlevel Prosodic Features Toolkit

Version 1.1

Nigel Ward, University of Texas at El Paso

September 15, 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 designed 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 a corpus with metadata (for now just Switchboard), compute various correlations and statistics on the factors affecting interaction styles .
- given any new dialog, compute where it lies in an existing space of interaction styles.
- output information to support qualitative interpretation of the style dimensions
- give a new corpus or sub-corpus, characterize its mean and variation on each of the dimensions **pending**
- output stimuli etc. to use in validation experiments

This document serves mostly to overview the workflow and the specific Matlab functions to call for each step. It should be read after getting the big picture from the paper. You'll probably also want to skim the midlevel toolkit documentation. Then to learn how to actually run things, read the comments in the code. This document 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/istyles` . 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 Subdirectories

Code

code – source code, mostly matlab, with some bash and aw k
reaper – David Talkin's pitch tracker

Documentation subdirectories

doc – this documentation
paper – draft of a journal article

Metadata and model parameters

splits – files listing the data subsets: training, test ...
swbd-various – switchboard metadata and counts
pcdparms – parameters relating to the prosodic constructions: how to compute and gather stats over
experiment – plans for the experiment, also `scales-to-dims-v4.txt`

Data and data products

... — the switchboard data itself is on another drive

pcdstats – the statistics on the prosodic construction distributions over a corpus
wordstats – lexical occurrence statistics etc for each pole of the training-data
clips-for-experiment – audio files to be uploaded to questionPro. Includes hand-generated files for the instructions etc., plus copies of the various stimulus-sets
trainIStyles – all sorts of stuff on the training data
testIStyles – all sorts of stuff on the test data

Test data, etc.

shortTests – sample wav files etc., mostly for testing `computStyleParams`
f0reaper – a few f0 files for testing reaper and the way its called
stats – former working space for various intermediate files and results
precious – copies of things that took a long time to compute
illustrations – audio clips for use in talks
old — dead code and obsolete data
tmp — temporary files

6 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 PCDS file, as it contains "prosodic-construction distribution statistics" (specifically the bin frequencies).

This will take 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. These files are in `pcdparams`. 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. These 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, from the top-level `istyles` directory run `deriveISspace('train-30sec.csv', true, 'trainIsNormRot.mat', 'trainset-out')` . This will create the model and save its parameters in the specified `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).

7 Interpreting the Model's Dimensions

First examine `loadingsTables.txt` a human-readable file written by `deriveISpace`. This shows, for each dimension (each interaction-style dimension) the loadings on the behavioral features (the PCDS features).

Examine the various files with names starting with `isratios`, showing how which words are characteristic of each pole of each dimension. 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`.

Use LIWC on the word list files, like `idim7loWords.txt`, to learn which word categories ar most common for which poles.

`deriveIStyles.m` creates a script to generate informative audio fragments: run this with `bash manySoxCmds.sh`. These can then be listened to (training set only, of course) to help infer the meaning of the various dimensions. (The script also creates some anchor stimuli, at one time thought useful for the human-perceptions experiment.) Once created, it's may be nice to move these all somewhere safe, e.g. `exemplars`.

8 Validating the Model by Human-Subjects Experiments

`deriveIStyles.m` also a script `mtSoxCmds.shs` to generate stimuli, and `preds-for-mturk` files to document its predictions for those stimuli.

You'll then copy the stimuli into QuestionPro ...

Then you'll link to mTurk and get human judgments

Then you'll download the questionPro results, following the instructions in the comments of `anaPerceptions.m`, and process them with `anaPerceptions.m` to evaluate the extent to which the predictions match human judgments.

9 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.

10 Acknowledgments