Interaction Style Modeling Toolset

an extension of the Midlevel Prosodic Features Toolikit

Version 1.1

Nigel Ward, University of Texas at El Paso

April 30, 2021

1 Overview

This toolset supports the analysis of interaction styles in spoken dialog. While people clearly vary in interaction styles, a comprehensive model of the space of variation has been lacking.

While this toolset has so far been used only for the analysis in one paper, as yet unpublished, it is designed to generally support computational analyses of interaction styles, for both scientific and practical purposes.

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
- given a new corpus or sub-corpus, characterize its mean and variation on each of the dimensions **pending**
- output stimuli and predictions to use in validation experiments

This document serves mostly to overview the workflow and to name 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

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

This code is mostly written in Matlab and runs on Matlab version 2019.

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 subset. 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
- src – source code, mostly matlab, with some bash and awk
- reaper – David Talkin's pitch tracker

Documentation
- doc – this documentation
- ../papers/istyles/ – 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 them
- experiment – plans for the experiment, also `scales-to-dims-v4.txt`

Data and data products
- /cygdrive/f/nigel/comparisons/nigel/swbd — the Switchboard data itself, at least on my machine
- pcdstats – the statistics on the prosodic construction distributions over a subcorpus
- wordstats – lexical occurrence statistics for each pole of each training-data dimension
- 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, miscellaney
- shortTests – sample wav files etc., mostly for testing `computStyleParams`
- f0reaper – a few F0 files for testing reaper and the way its called
- 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 `istyles` directory.

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

`computeStyleParams` function three hidden dependencies, in which it reads data from hard-coded 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.

`computeStyleParams` will take time (12 hours on a 8GB 3.6GHz machine). If it crashes before completion, don't start over. Instead, 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.

After `computeStyleParams` completes, 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 showing the effects of training set size, if `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`, which show which words are characteristic of each pole of each dimension. These can be examined, for example by `sort -n isratios1hi.txt | more`.

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

`deriveISpace.m` also 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 may be nice to move these all somewhere safe, e.g. `exemplars`.

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

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

This will, instead of making a new model, apply the existing model (the `IsNormRot` file) 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.

9 Validating the Model by Human-Subjects Experiments

The core of the experiments are the stimuli and the system's predictions for those stimuli. These both are generated in sets of 16; 4 such sets are generated. Both are specified by `deriveISpace.m`, specifically, it writes a script `mtSoxCmds.sh` which can then be run to generate all the stimuli, and it writes `mtPredictions.csv` file to document its predictions for those stimuli. Recent versions of both are found in `trainset-out` and `testIStyles`.

Judgments are obtained through QuestionPro. To create a survey in QuestionPro, first copy all the stimuli for the appropriate set up to its Media Library. As each 30-second clip is about 500KB, this is not onerous. In addition, from `experiment/clips-for-experiment/`, copy up `attention-check/full-attention-check.wav`, instructing them to mark "3" for all items, and the five wav files from `stereo-check`. 1 and 4 play on the right, 2 and 3 on the left.

Now build the audio into the survey. Start by copying an existing survey. Open two browswer windows, one for the new survey, and one for QuestionPro's Media Library (accessible via the 3rd-from-left icon in the top toolbar seen when editing the survey). For each file needed

- in the Media Library, click on the audio you need
- click on the 3 dots to expand the menu, then click "get html" to copy the code to the clipboard
- then go back to the window where the survey is open
- locate the relevant question and click on the up/down arrows to expand it
- click somewhere on the instructions and a floating toolbar appears
- click on the rightmost icon in that toolbar, a box with a northeast arrow, to open the editor
- click on the "Source" option at the upper left to be able to edit the html
- paste the clicked link in the html
- save

Next, build the first survey: Perceptions of Interaction Styles in Spoken Dialog@NGW:

(Pilot) (using stimulus set 1). The audio is all in `testIStyles`, for example as `stimulus-1-14.wav`.

There will be 3 other surveys: Perceptions of Interaction Styles in Spoken Dialog: 2 (respectively 3 and 4). To create these, copy an existing survey and rewire the audios.

**Test the survey ...

...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.

Then you'll link to mTurk and get human judgments ...

procedure for downloading results and testing the hypotheses

10 Acknowledgments

Jonathan Avila designed the questionPro integration.

istyles/doc/howto-is.tex