

Semi-supervised machine transcription of spoken audio into phonemes (speech units).

MIT license

1 star 0 forks

Star

Notifications

Code Issues Pull requests Actions Projects Wiki Security Insights

master

81

[View code](#)

README.md

# speech2phone

## TODO

Mark when you've finished them.

- (Kyle) Preprocessor caching ✓
- (Kyle) Preprocessor returns categorical distribution ✓
- (Kyle) Embedding baseline
- (Seong) Python *scripts* (not notebooks) that use grid search, `mag` and save the plots in `/visualizations` for the following models: ✓
  - Random Forest
  - XGBoost
  - Gaussian Discriminant Analysis
  - Naive Bayes
  - Logistic Regression
  - Principal Component Analysis

- Support Vector Machine
- K-Nearest Neighbors
- K-Means
- Gaussian Mixture Model
- (Jared) Semi-supervised learning *scripts* (not notebooks) with fully-connected layer and 1-D CNN
  - Self-training
  - Co-training
  - Pi-model
  - Label propagation
  - Label gradient alignment
  - Using your model against itself
- (anyone)

## 🔗 Directory Structure

---

- *embedding/*: Learned embeddings, applied after preprocessing. For example, PCA.
- *experiments/class/*: *mag* experiments on classification (phoneme boundaries given to model).
- *experiments/seg\_class/*: *mag* experiments on segmentation and classification (phoneme boundaries produced by model).
- *models/*: custom model classes we've built.
- *preprocessing/*: Loads data from files, caches it, and returns NumPy arrays.
- *results/*: Assorted images/ plots that are interesting and could be useful in the final report. For example, a PCA .png
- *temp\_{jared, kyle, seong}/*: The equivalent of branches. Put work-in-progress here, and bring it out into the main system when it's done.
- *visualizations/*: Examples of how to plot a Mel spectrogram, etc.

## 🔗 Testing

---

- Run `pytest test_main.py`.
- Add additional tests there. We'll use a single test module for now. `pytest` uses simple assert statements.

## 🔗 Rules

---

- The directory containing `speech2phone` must be on the environment variable `PYTHONPATH`.
- To append it, run `export PYTHONPATH="${PYTHONPATH}:/my/other/path"`.
- For example, if I have `Users/jarednielsen/Desktop/speech2phone`, then I must have `Users/jarednielsen/Desktop` on my `PYTHONPATH`.
- If that doesn't work because of conda, Add a `.pth` file to the directory `$HOME/path/to/anaconda/lib/pythonX.X/site-packages`. This can be named anything (it just must end with `.pth`). A `.pth` file is just a newline-separated listing of the full path-names of directories that will be added to your path on Python startup. For example, `/anaconda3/envs/py36/lib/python3.6/site-packages/path.pth` has the line `/Users/jarednielsen/Desktop` in it.
- Use absolute imports everywhere. For example, `import speech2phone` or `import speech2phone.preprocessing`.
- See `speech2phone/__init__.py` and `speech2phone/preprocessing/__init__.py` for examples of how to set up subpackages.
- `/preprocessing` applies classic data processing methods (i.e. not learned) to the data, while `/embedding` applies learned methods. For example, Mel spectrogram stuff should be handled in `/preprocessing`.

## ↻ boundary recognition

---

### Approaches

- Recurrent network
- Merging (like piecewise linear regression) with the criterion over a metric using dynamic time-warping

## ↻ `/embedding`

---

Options for embedding include:

- spectrum
- cepstrum
- single linear layer (we could try [this](#) or just SGD)
- more complex learned network
- autoencoder
- UMAP
- t-SNE

These will all be specifiable by importing from the embedding module. The spectrum works pretty well as an embedding space, as we found by doing some PCA (see [/visualizations/pca\\_embedding.png](/visualizations/pca_embedding.png)). I think we'll use it as a baseline.

## 🔗 Things to try (add ideas here)

---

- trinemes
- dynamic time-warping
- reapply models to TIMIT to quantify results (quantified semi-supervised learning)
- use Mel spectrogram but give some time dependence (80 freq x 10 time)
- using the activations from the neural network to try to predict the speaker, and then consider the ethical implications (a la "voiceprint" technology)

## Releases

No releases published

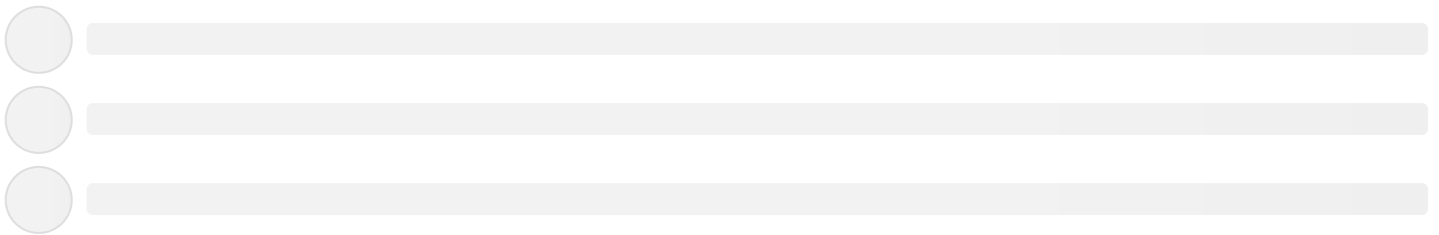
---

## Packages

No packages published

---

## Contributors 3



## Languages

