The following is a high level overview of the data for this competition. It is highly recommended that you read the supplementary materials found on the tabs of the Overview page.

File descriptions

- **train.7z** – Contains a few informational files and a folder of audio files. The audio folder contains subfolders with 1 second clips of voice commands, with the folder name being the label of the audio clip. There are more labels that should be predicted. The labels you will need to predict in Test are `yes`, `no`, `up`, `down`, `left`, `right`, `on`, `off`, `stop`, `go`. Everything else should be considered either `unknown` or `silence`. The folder `_background_noise_` contains longer clips of "silence" that you can break up and use as training input.

  The files contained in the training audio are not uniquely named across labels, but they are unique if you include the label folder. For example, `00f0204f_nohash_0.wav` is found in 14 folders, but that file is a different speech command in each folder.

  The files are named so the first element is the subject id of the person who gave the voice command, and the last element indicated repeated commands. Repeated commands are when the subject repeats the same word multiple times. Subject id is not provided for the test data, and you can assume that the majority of commands in the test data were from subjects not seen in train.

  You can expect some inconsistencies in the properties of the training data (e.g., length of the audio).
- **test.7z** – Contains an audio folder with 150,000+ files in the format `clip_000044442.wav`. The task is to predict the correct label. Not all of the files are evaluated for the leaderboard score.
- **sample_submission.csv** – A sample submission file in the correct format.
- **link_to_gcp_credits_form.txt** – Provides the URL to request $500 in GCP credits, provided to the first 500 requestors. Please see this clarification on credit qualification.

The following is the README contained in the Train folder, and contains more detailed information.

Speech Commands Data Set v0.01

This is a set of one-second .wav audio files, each containing a single spoken English word. These words are from a small set of commands, and are spoken by a variety of different speakers. The audio files are organized into folders based on the word they contain, and this data set is designed to help train simple machine learning models.

It's licensed under the Creative Commons BY 4.0 license. See the LICENSE file in this folder for full details. Its original location was athttp://download.tensorflow.org/data/speech_commands_v0.01.tar.gz.

History

This is version 0.01 of the data set containing 64,727 audio files, released on August 3rd 2017.

Collection

The audio files were collected using crowdsourcing, seeaiyprojects.withgoogle.com/open_speech_recording for some of the open source audio collection code we used (and please consider contributing to enlarge this data set). The goal was to gather examples of people speaking single-word commands, rather than conversational sentences, so they were prompted for individual words over the course of a five minute session. Twenty core command words were recorded, with most speakers saying each of them five times. The core words are "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", and "Nine". To help distinguish unrecognized words, there are also ten auxiliary words, which most speakers only said once. These include "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow".

Organization

The files are organized into folders, with each directory name labelling the word that is spoken in all the contained audio files. No details were kept of any of the participants age, gender, or location, and random ids were assigned to each individual. These ids are stable though, and encoded in each file name as the first part before the underscore. If a participant contributed multiple utterances of the same word, these are distinguished by the number at the end of the file name. For example, the file path `happy/3cfc6b3a_nohash_2.wav` indicates that the word spoken was "happy", the speaker's id was "3cfc6b3a", and this is the third utterance of that word by this speaker in the data set. The 'nohash' section is to ensure that all the utterances by a single speaker are sorted into the same training partition, to keep very similar repetitions from giving unrealistically optimistic evaluation scores.

Partitioning

The audio clips haven't been separated into training, test, and validation sets explicitly, but by convention a hashing function is used to stably assign each file to a set. Here's some Python code demonstrating how a complete file path and the desired validation and test set sizes (usually both 10%) are used to assign a set:

```python MAX_NUM_WAVS_PER_CLASS = 2**27 - 1 # ~134M

def which_set(filename, validation_percentage, testing_percentage): """Determines which data partition the file should belong to.

We want to keep files in the same training, validation, or testing sets even if new ones are added over time. This makes it less likely that testing samples will accidentally be reused in training when long runs are restarted for example. To keep this stability, a hash of the filename is taken and used to determine which set it should belong to. This determination only depends on the name and the set proportions, so it won't change as other files are added.

It's also useful to associate particular files as related (for example words spoken by the same person), so anything after '*nohash*' in a filename is ignored for set determination. This ensures that 'bobby_nohash_0.wav' and 'bobby_nohash_1.wav' are always in the same set, for example.

Args: filename: File path of the data sample. validation_percentage: How much of the data set to use for validation. testing_percentage: How much of the data set to use for testing.

Returns: String, one of 'training', 'validation', or 'testing'. """ base_name = os.path.basename(filename) # We want to ignore anything after '*nohash*' in the file name when # deciding which set to put a wav in, so the data set creator has a way of # grouping wavs that are close variations of each other. hash_name = re.sub(r'*nohash*.*$', '', base_name) # This looks a bit magical, but we need to decide whether this file should # go into the training, testing, or validation sets, and we want to keep # existing files in the same set even if more files are subsequently # added. # To do that, we need a stable way of deciding based on just the file name # itself, so we do a hash of that and then use that to generate a # probability value that we use to assign it. hash_name_hashed = hashlib.sha1(hash_name).hexdigest() percentage_hash = ((int(hash_name_hashed, 16) % (MAX_NUM_WAVS_PER_CLASS + 1)) * (100.0 /
```

MAX_NUM_WAVS_PER_CLASS)) if percentage_hash < validation_percentage: result = 'validation' elif percentage_hash < (testing_percentage + validation_percentage): result = 'testing' else: result = 'training' return result ```

The results of running this over the current set are included in this archive as validation_list.txt and testing_list.txt. These text files contain the paths to all the files in each set, with each path on a new line. Any files that aren't in either of these lists can be considered to be part of the training set.

Processing

The original audio files were collected in uncontrolled locations by people around the world. We requested that they do the recording in a closed room for privacy reasons, but didn't stipulate any quality requirements. This was by design, since we wanted examples of the sort of speech data that we're likely to encounter in consumer and robotics applications, where we don't have much control over the recording equipment or environment. The data was captured in a variety of formats, for example Ogg Vorbis encoding for the web app, and then converted to a 16-bit little-endian PCM-encoded WAVE file at a 16000 sample rate. The audio was then trimmed to a one second length to align most utterances, using the extract_loudest_section tool. The audio files were then screened for silence or incorrect words, and arranged into folders by label.

Background Noise

To help train networks to cope with noisy environments, it can be helpful to mix in realistic background audio. The `_background_noise_` folder contains a set of longer audio clips that are either recordings or mathematical simulations of noise. For more details, see the `_background_noise_/README.md`.

Citations

If you use the Speech Commands dataset in your work, please cite it as:

APA-style citation: "Warden P. Speech Commands: A public dataset for single-word speech recognition, 2017. Available fromhttp://download.tensorflow.org/data/speech_commands_v0.01.tar.gz".

BibTeX `@article{speechcommands, title={Speech Commands: A public dataset for single-word speech recognition.}, author={Warden, Pete}, journal={Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz}, year={2017} }`

Credits

Massive thanks are due to everyone who donated recordings to this data set, I'm very grateful. I also couldn't have put this together without the help and support of Billy Rutledge, Rajat Monga, Raziel Alvarez, Brad Krueger, Barbara Petit, Gursheesh Kour, and all the AIY and TensorFlow teams.

Pete Warden, petewarden@google.com