The Journal of Open Source Software

audiomate: A Python package for working with audio datasets

Matthias Büchi[1] **and Andreas Ahlenstorf**[1]

1 ZHAW Zurich University of Applied Sciences, Winterthur, Switzerland

DOI: 10.21105/joss.02135

Software

- Review
- Repository
- Archive

Editor: Yuan Tang

Reviewers:

- @mulhod
- @faroit

Submitted: 25 February 2020
Published: 05 August 2020

Summary

Machine learning tasks in the audio domain frequently require large datasets with training data. Over the last years, numerous datasets have been made available for various purposes, for example, (Snyder, Chen, & Povey, 2015) and (Ardila et al., 2019). Unfortunately, most of the datasets are stored in widely differing formats. As a consequence, machine learning practitioners have to convert datasets into other formats before they can be used or combined. Furthermore, common tasks like reading, partitioning, or shuffling of datasets have to be developed over and over again for each format and require intimate knowledge of the formats. We purpose Audiomate, a Python toolkit, to solve this problem.

Audiomate provides a uniform programming interface to work with numerous datasets. Knowledge about the structure or on-disk format of the datasets is not necessary. Audiomate facilitates and simplifies a wide range of tasks:

- Reading and writing of numerous dataset formats using a uniform programming interface, for example (Snyder et al., 2015), (Panayotov, Chen, Povey, & Khudanpur, 2015) and (Ardila et al., 2019)
- Accessing metadata, like speaker information and labels
- Reading audio data (single files, batches of files)
- Retrieval of information about the data (e.g., number of speakers, total duration).
- Merging of multiple datasets (e.g., combine two speech datasets).
- Splitting data into smaller subsets (e.g., create training, validation, and test sets with a reasonable distribution of classes).
- Validation of data for specific requirements (e.g., check whether all samples were assigned a label)

Use Cases

To illustrate Audiomate's capabilities, we present two typical applications where Audiomate significantly simplifies the task of a developer: Training a speech recognition model with Mozilla's implementation of DeepSpeech and training a deep neural network to recognize music.

Converting Datasets

In this example, we illustrate how to employ Audiomate to convert the LibriSpeech dataset (Panayotov et al., 2015) into the CSV-format expected by Mozilla's implementation (https://github.com/mozilla/DeepSpeech) of DeepSpeech (Hannun et al., 2014) which can, in turn, be used to train an automatic speech recognition model.

```
import audiomate
from audiomate.corpus import io

# Download LibriSpeech corpus
downloader = io.LibriSpeechDownloader()
downloader.download('/local/data/librispeech')

# Read LibriSpeech
reader = io.LibriSpeechReader()
librispeech = reader.load('/local/data/librispeech')

# Save in DeepSpeech format
writer = io.MozillaDeepSpeechWriter()
writer.save(librispeech, '/local/data/librispeech_ds')
```

Knowledge of the on-disk formats of the datasets is not required.

Some datasets contain invalid or corrupted files. If those are known, Audiomate tries to rectify the problems or automatically excludes those files before processing any data.

Merging and Partitioning Datasets

Another area where Audiomate excels is mixing datasets and partitioning them into training, test, and validation sets. Assume that the task is to train a neural network to detect segments in audio streams that are music. MUSAN (Snyder et al., 2015) and GTZAN ("GTZAN music/speech collection," n.d.) are two suitable datasets for this task because they provide a wide selection of music, speech, and noise samples. In the example below, we first download MUSAN and GTZAN to the local disk before creating `Loader` instances for each format that allow Audiomate to access both datasets using a unified interface. Then, we instruct Audiomate to merge both datasets. Afterwards, we use a `Splitter` to partition the merged dataset into a train and test set. By merely creating views, Audiomate avoids creating unnecessary disk I/O and is therefore ideally suited to work with large datasets in the range of tens or hundreds of gigabytes. Ultimately, we load the samples and labels by iterating over all utterances. Audio samples are numpy arrays. They allow for fast access, high processing speed and ensure interoperability with third-party programs that can operate on numpy arrays, for example TensorFlow or PyTorch. Alternatively, it is possible to load the samples in batches, which is ideal for feeding them to a deep learning toolkit like PyTorch.

```
import audiomate
from audiomate.corpus import io
from audiomate.corpus import subset

musan_dl = io.MusanDownloader()
musan_dl.download('/local/data/musan')

gtzan_dl = io.GtzanDownloader()
gtzan_dl.download('/local/data/gtzan')

musan = audiomate.Corpus.load('/local/data/musan', reader='musan')
gtzan = audiomate.Corpus.load('/local/data/gtzan', reader='gtzan')

full = audiomate.Corpus.merge_corpora([musan, gtzan])

splitter = subset.Splitter(full, random_seed=222)
```

```
subviews = splitter.split(proportions={
    'train': 0.8,
    'test': 0.2,
})

for utterance in subviews['train'].utterances.values():
    samples = utterance.read_samples()
    labels = utterance.label_lists[audiomate.corpus.LL_DOMAIN]
```

Implementation

Audiomate was designed with extensibility in mind. Therefore, it is straightforward to add support for additional data formats. Support for another format can be added by implementing at least one of three available abstract interfaces.

- Reader: A `Reader` defines the procedure to load data that is structured in a specific format. It converts it into a Audiomate-specific data structure.
- Writer: A `Writer` defines the procedure to store data in a specific format. It does that by converting the data from the Audiomate-specific data structure into the target format.
- Downloader: A `Downloader` can be used to download a dataset. It downloads all required files automatically.

Rarely all interfaces are implemented for a particular format. Usually, `Reader` and `Downloader` are implemented for datasets, while `Writer` is implemented for machine learning toolkits.

Audiomate supports more than a dozen datasets and half as many toolkits.

Related Work

A variety of frameworks and tools offer functionality similar to Audiomate.

Data loaders Data loaders are libraries that focus on downloading and preprocessing data sets to make them easily accessible without requiring a specific tool or framework. In contrast to Audiomate, they cannot convert between formats, split or merge data sets. Examples of libraries in that category are ("Mirdata," 2020), ("Speech corpus downloader," 2020), and ("Audio datasets," 2020). Furthermore, some of these libraries focus on a particular kind of data, such as music, and do not assist with speech data sets.

Tools for specific frameworks Various machine learning tools and deep learning frameworks include the necessary infrastructure to make various datasets readily available to their users. One notable example is TensorFlow (Abadi et al., 2016), which includes data loaders for different kinds of data, including image, speech, and music data sets, such as (Ardila et al., 2019). Another one is torchaudio ("TORCHAUDIO," 2020) for PyTorch, which not only offers data loaders but is also capable of converting between various formats. In contrast to Audiomate, those tools or libraries support a specific machine learning or deep learning framework (TensorFlow or PyTorch, respectively), whereas Audiomate is framework agnostic.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (pp. 265–283). Savannah, GA: USENIX Association. ISBN: 978-1-931971-33-1

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., et al. (2019). Common voice: A massively-multilingual speech corpus. Retrieved from http://arxiv.org/abs/1912.06670

Audio datasets. (2020). https://github.com/mcfletch/audiodatasets.

GTZAN music/speech collection. (n.d.). http://marsyas.info/downloads/datasets.html.

Hannun, A. Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *CoRR*, *abs/1412.5567*. Retrieved from http://arxiv.org/abs/1412.5567

Mirdata. (2020). https://github.com/mir-dataset-loaders/mirdata.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An asr corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5206–5210. doi:10.1109/icassp.2015.7178964

Snyder, D., Chen, G., & Povey, D. (2015). MUSAN: A Music, Speech, and Noise Corpus.

Speech corpus downloader. (2020). https://github.com/mdangschat/speech-corpus-dl.

TORCHAUDIO. (2020). https://pytorch.org/audio/.