Requirements for the Automated Audio Captioning Python Package, Emuu

K. Drossos, S. Lipping, and T. Virtanen
Audio Research Group, Tampere University
{firstname.lastname}@tuni.fi

1. Purpose

1.1 Definitions and acronyms

AAC:	automated audio captioning
API:	application programming interface
caption:	a textual description of an audio signal
CLI:	command line interface
Clotho:	the audio captioning dataset called Clotho
DNN:	deep neural network
Emuu:	the name of the AAC python package
metadata:	metadata of audio files, available in Clotho, consisting of textual data
OOP:	object oriented programming
SW:	software

1.2 Background

AAC is a recent and novel research task, where a method/system gets as an input some audio data or features extracted from these audio data, and outputs a caption of the audio data. To do this, the methods/systems consist of DNNs that are trained using as input the audio or the extracted features and as a targeted output different ground-truth captions. The audio data and the captions are part of an AAC dataset. The performance of the AAC system is evaluated using again as an input some audio data (or features) and comparing the predicted captions with some ground truth ones. During the evaluation of the performance, the difference of the predicted and the ground truth captions is performed using a set of captioning metrics. These metrics are already implemented in Python and require as an input the predicted caption and a set of ground truth captions for the same audio data.

Being a recent and novel task, there are no available Python packages to standardize the above processes for AAC. Additionally, there is some data pre-processing that has to happen in order to use the AAC data with an AAC system, for which pre-processing also does not exist a standardized approach.

This document is about a Python SW package that will offer a common and standardized API for the above described processes. The automated AAC package will be mostly used by the research community, and specifically by the researchers that are focusing on AAC. Apart from the research community, there is a good possibility that the package will be used by developers who want to integrate the AAC functionalities in their code.

1.3 Overview and use cases of Emuu

The goal of this package is to offer a structured, standardized, and pythonic application programming interface for all the above. The package will be made available through standard Python SW channels, e.g. PIP and Anaconda/Conda.

The developed package will be employed in different use cases, with most common the following ones.

1.3.1 Use case 1: Downloading AAC dataset

1. User has already installed Emuu
2. User wants to download Clotho dataset
3. Use downloads Emuu by either
    a. Using the CLI of Emuu, or
    b. Using the API of Emuu in their code
4. Emuu expands the archive files of Clotho and moves the data to either a default location, or a location specified by the user.
5. Emuu registers the dataset in its settings, so it can be found from other processes of Emuu

1.3.2 Use case 2: Listen to audio

1. User has already installed Emuu and has downloaded Clotho
2. User wants to listen to a specific audio file, or part(s) of the audio file
3. User uses either:
    a. CLI of Emuu and specifies which file will hear and what part(s), by default is the whole audio, or
    b. API of Emuu in their script and specifies which file will hear and what part(s), by default is the whole audio
4. Emuu reproduces the specified audio file through available audio interface

1.3.3 Use case 3: See caption(s)

1. User has already installed Emuu and has downloaded Clotho
2. User wants to read the caption(s) of a specific audio file
3. User uses either:
    a. CLI of Emuu and specifies from which file the user will read the caption, or

    b. API of Emuu in their script and specifies from which file the user will read the caption
4. Emuu prints the caption(s) on the screen

1.3.4 Use case 4: See metadata

1. User has already installed Emuu and has downloaded Clotho
2. User wants to read the metadata of a specific audio file
3. User uses either:
    a. CLI of Emuu and specifies from which file the user will see the metadata, or
    b. API of Emuu in their script and specifies from which file the user will read the metadata
4. Emuu prints the metadata on the screen

1.3.5 Use case 5: Get audio data, corresponding captions, and metadata

1. User has already installed Emuu and has downloaded Clotho
2. User wants to use audio, captions, and/or metadata of the audio data
3. User uses API of Emuu in their script and specifies the audio files that wants to use
4. Emuu API returns the audio data, the captions, and (if specified) the metadata

1.3.6 Use case 6: Pre-process audio data, captions, and metadata

1. User has already installed Emuu and has downloaded Clotho
2. User wants to use pre-process audio, captions, and/or metadata of the audio data
3. User uses API of Emuu in their script and specifies the audio files that wants to use
4. User either:
    a. employes default pre-processing functionalities of Emuu, or
    b. uses user-defined processes for pre-processing the data
5. Emuu API returns the pre-processed audio data, the captions, and (if specified) the metadata

1.3.7 Use case 7: Extract features from audio data, corresponding captions, and metadata

1. User has already installed Emuu and has downloaded Clotho
2. User wants to use pre-process audio, captions, and/or metadata of the audio data
3. User uses API of Emuu in their script and specifies the audio files that wants to use
4. User either:
    a. employes default feature extraction functionalities of Emuu, or
    b. uses user-defined processes for feature extraction from the data
5. Emuu API returns the features extracted from the audio data, the captions, and (if specified) the metadata

1.3.8 Use case 8: Create splits for training and evaluating AAC methods

1. User has already installed Emuu and has downloaded Clotho
2. User wants to create data splits for training and/or evaluating AAC methods

3. User uses Emuu API in their script and specifies pre-processing, feature extraction processes, and files to be used for validation split
4. Emuu creates training, validation, and testing splits
5. Emuu exports these splits at default or user-specifies locations

1.3.9 Use case 9: Create PyTorch DataLoader objects or PyTorch Lightning DataModule objects

1. User has already installed Emuu and has downloaded Clotho
2. User wants to create data splits for training and/or evaluating AAC methods and use PyTorch or PyTorch Lightning classes
3. User either:
    a. uses Emuu API in their script and specifies pre-processing, feature extraction processes, and files to be used for validation split, or
    b. specifies previously created splits by Emuu
4. Emuu returns the data splits wrapped in the specified classes

1.3.10 Use case 10: Plotting statistics of dataset

1. User has already installed Emuu and has downloaded Clotho
2. User wants to see statistics of the dataset, either from audio, captions, metadata, or any combination of these
3. User employs Emuu and either:
    a. employs available statistics functions from Emuu, or
    b. uses user-defined functions for calculating statistics
4. User employs plotting functionality from Emuu to plot the data

1.3.11 Use case 11: Export splits

1. User has already installed Emuu, has downloaded Clotho, and has created splits
2. User wants to export the splits that already has created
3. User uses either:
    a. CLI of Emuu, or
    b. the API of Emuu
4. Emuu exports the data to a specified location

1.3.12 Use case 12: Update dataset

1. User has already installed Emuu and has downloaded Clotho
2. User wants to update Clotho to a newer version (if available)
3. User uses either:
    a. CLI of Emuu, or
    b. the API of Emuu
4. Emuu updates data of Clotho and issues a warning that new splits/features should be created (if applicable)

1.3.13 Use case 13: Evaluate predicted captions

1. User has already installed Emuu, has downloaded Clotho, has created splits, and has acquired predicted captions on the splits that has created
2. User wants to evaluate the predicted captions using available metrics
3. User either:
    a. provides a .csv file with audio file information, split information, and predicted captions for each of the audio files, or
    b. employs the API of Emuu, using variable having the information about audio data, split information, and predicted captions
4. User also specifies which of the metrics wants to use/calculate
5. Emuu returns the values for the specified metrics

1.3.14 Use case 14: Query evaluations of predicted captions

1. User has already installed Emuu, has downloaded Clotho, has created splits, has acquired predicted captions on the splits that has created, and has evaluated the predicted captions
2. User wants to get the top-K, low-K, or captions in a range of values according to a specific metric
3. User employs the API of Emuu, specifies the query (e.g. top or low K, what the K), and retrieves the audio data, the ground truth captions, and the corresponding metadata

2. Requirements of Emuu

2.1 Functional requirements

Emuu has the following functional requirements

- Installed in existing Python environments, using typical Python package managers, e.g. PIP and Anaconda/Conda
- Upgraded using the functionalities for the corresponding Python package managers
- Used in scripts or with a CLI
- Download, expand, and handle Clotho dataset
- Pre-process the AAC data
- Extract features from AAC data
- Provide API for using user-defined functions for pre-processing and feature extraction
- Reproduce and display the AAC data
- Evaluate predicted captions and handle the results
- Plot the AAC data and the results from the evaluations of captions
- Integrated with PyTorch and PyTorch Lightning

2.2 Usability requirements

Emuu has the following usability requirements

- Organized in meaningful packages, so users can import the absolute minimum functions and/or classes
- Provide standard documentation for the user
- Has meaningful names for functions, classes, methods, and attributes
- Provide meaningful error and warning messages

2.3 Technical requirements

Emuu has the following technical requirements

- User should be able to import bare minimum functions and classes in a script
- Usage of OOP paradigm
- Usage of common API for the classes, as much as possible
- Methods must be able to be cascaded, if applicable
- Use custom string representation of Python classes and objects
- Use Emuu specific errors and warnings
- Support Python version greater than 3.6
- Built with the least possible dependencies