BAT: An open-source, web-based audio events annotation tool

Blai Meléndez-Catalán
Music Technology Group,
Universitat Pompeu Fabra
Roc Boronat 138, 08018
Barcelona, Spain
bmelendez@bmat.com

Emilio Molina
Barcelona Music and Audio
Techonologies
Bruniquer 49, 08024
Barcelona, Spain
emolina@bmat.com

Emilia Gómez
Music Technology Group,
Universitat Pompeu Fabra
Roc Boronat 138, 08018
Barcelona, Spain
emilia.gomez@upf.edu

ABSTRACT

In this paper we present BAT (BMAT Annotation Tool), an open-source, web-based tool for the manual annotation of events in audio recordings developed at BMAT (Barcelona Music and Audio Technologies[1]). The main feature of the tool is that it provides an easy way to annotate the salience of simultaneous sound sources. Additionally, it allows to define multiple ontologies to adapt to multiple tasks and offers the possibility to cross-annotate audio data. Moreover, it is easy to install and deploy on servers. We carry out an evaluation where 3 annotators use BAT to annotate a small dataset composed of broadcast media recordings. The results of the experiments show that BAT offers fast annotation mechanisms and a method to assign salience that produces high agreement among annotators.

CCS Concepts

•Information systems → Open source software; Web applications; Information extraction;

Keywords

Annotation tool, audio events, web-based, open-source, salience annotation

1. INTRODUCTION

In most tasks related to Music Information Retrieval (MIR) and Speech Recognition, systems are trained and evaluated on audio material that has been previously annotated by humans. The annotation process is usually tedious and time consuming; therefore, many annotation tools have been developed to facilitate this process. Some of these tools cover a large variety of tasks [2, 3] while others focus on a small set of them [6]. In this sense, BAT is exclusively designed for the annotation of audio events. However, an event can be a chord, a bird's chirp, words, the parts of a song or the notes of the instruments that play it; thus, BAT is able to generate annotations for many different tasks, allowing the definition of an appropriate ontology for each of them. Additionally, it includes the possibility for several annotators to cross-annotate the audio to capture the ambiguity that its content might produce in relation to the defined ontology.

The main motivation behind the development of the present tool is to advance the research on the task of music detection in broadcast recordings, i.e., detecting which parts of a recording contain music even though it is mixed with other type of sounds. Specifically, we are interested in the investigation of the benefits of using information about the salience of music with respect to other possible simultaneous sound sources. For this reason, BAT is designed to offer an easy way to annotate the salience of sound events that coincide in time. This research is aligned with the interests of BMAT, a company based in Barcelona that monitors music across televisions, radios, clubs and digital services of all over the world, and reports it to more than 100 clients, processing tens of thousands of hours of audio a day.

This paper includes 6 sections: in Section 2, we summarize the requirements behind this development; then, in Section 3, we review the most relevant audio annotation tools. Section 4 is devoted to a thorough description of our tool and, in Section 5, we detail its evaluation process and discuss the obtained results. Finally, in Section 6, we expose our conclusions.

2. REQUIREMENTS

We state three requirements regarding BAT features: (1) to allow for an easy annotation of the salience of simultaneous sound sources; (2) to permit the creation of fitting ontologies for the multiple tasks it can be applied to; and (3) to enable the cross-annotation of audio data by several annotators.

In another sense, our tool also needs to be convenient to the potential users, in terms of annotation time and efficiency, while providing a simple, clear and agile annotation environment and also helpful functionalities to ease the annotation process. Finally, we want to facilitate the manipulation of the data stored in the tool's database and the access to detailed information about the annotation process.

3. RELATED WORK

To date, a variety of tools have been created for the an-

[1] www.bmat.com

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

notation of audio events. A recent one is CrowdCurio[2], which is a JavaScript web interface for the annotation of audio events that uses and extends the Regions plugin of wavesurfer.js[3]. BAT employs this extended plug-in as it includes useful features such as labeled regions or the possibility to switch the sound visualization between its waveform and its spectrogram (see Section 4.2). Another web-based tool is I-SED[4], which includes machine learning techniques to reduce the annotation time by automatically annotating some of the audio content and presenting it to the annotator for revision. SoundScape[5] also uses machine learning, and it allows the annotation of specific time-frequency regions of the spectrogram.

WaveSurfer[7][4] is a tool initially designed for speech annotation but deliberately made flexible and extensible to different tasks. Other tools that were originally meant for speech are Praat[1][5] and HAT[6]; however, Praat has been also used for music annotation. WaveSurfer was used to create MUCOSA[3], an environment for the annotation and generation of music metadata at different abstraction levels that incorporated a collaborative annotation subsystem. Sonic Visualizer[2] is another example of related tool, better known for its analysis and feature extraction applications, which are added to the tool using vamp plug-ins. However, it can also be used as an event annotation tool and can incorporate salience information using textual labels. Finally, ELAN[8] is a full-featured and complex tool that allows the annotation of both audio and video.

4. DESIGN

BAT is an open-source, web-based tool programmed in Django and JavaScript and dockerized for an easy deployment on servers. For the front-end part, we have integrated the extended version of wavesurfer used in the CrowdCurio project. Regarding the back-end part, we offer the possibility to interact with the content of the database externally, using Ipython and Jupyter. It currently accepts only audio in WAV format and works, at least, with Mozilla Firefox, Google Chrome and Opera. BAT can be found at its github repository[7] with a GNU AGPL v3.0 license. The repository contains the source code, documentation about the installation and the annotation process, and the necessary docker files.

4.1 Annotation process

The first step to start annotating audio with BAT is to create a project. Projects represent a particular task we want to annotate and should be named after it, for instance, we could name our project 'MusicSpeech'. Each of these projects can be assigned an unlimited number of classes, which ensures ontological flexibility. Additionally, when creating a project, it is possible to decide whether it allows its classes to overlap in time (for audio event detection tasks such as instrument recognition, music/non-music detection, etc.) or not (for audio segmentation tasks such as chord recognition, structural segmentation, etc.).

Once the project and its classes are ready we need to upload the data we want to annotate. The uploaded audio files are linked to the selected project and automatically split to generate segments. These segments are the pieces that the annotators will annotate independently and consecutively. Their length must be specified before uploading the audio file. We have decided to work with fixed length segments instead of the full audio files because it allows for an annotation with constant time precision, while avoiding actions such as zooming and scrolling, which introduce a cost in time and increase the complexity of the annotation process. Additionally, this setup favors the usage of crowdsourcing strategies by allowing the distribution of the workload among several annotators. Nevertheless, it is also possible to generate only one segment per audio file for tasks such as global key estimation, genre recognition, etc.

We define two different types of users based on the tasks they perform throughout the annotation process: the administrator and the annotator. The administrator is the user that creates projects, defines ontologies, uploads the audio data and chooses the segments' length. To this kind of user, the tool provides an environment to search and filter the content of all projects, which includes its classes, audio files and segments. It also allows the visualization of all annotations, but not its modification or generation. The other type of user, the annotator, has access only to its own annotations, but can modify them or generate new ones as long as there are still unannotated segments.

Annotating the content for some tasks may be an ambiguous exercise and the annotator might have to incorporate a certain amount of subjectivity. In these cases, the tool is prepared to manage the annotations of more than one user, i.e, to allow for cross-annotation. Having multiple annotations is a way to express this ambiguity and to detect which are the parts that produce more disagreement between annotators. Notice that all the actions taken during the annotation process are saved as logs, which store the action taken, the time when it was taken and other optional values permitting a detailed analysis.

When an annotator loads a segment, an annotation is immediately created linked to that segment and that annotator. We only allow a single annotation per segment and annotator. BAT divides the annotation process into two sequential phases: the *event identification* phase and the *salience assigning* phase. In the first one, the annotator must draw regions over the waveform containing events that are relevant to the current task, be it a person speaking, the time interval during which a certain chord sounds in a song or a note of a soloing saxophone. Then, it must assign a class from the available ontology to each of these regions and optionally add a tag to them. There is no restriction in the tags that the annotator can use.

If the events drawn in the first phase overlap in time, it is mandatory to go through the salience assigning phase. In this phase, new regions are automatically created that divide the segments into the largest possible time intervals with a homogeneous number and type of classes according to the events drawn in the previous phase. A salience value indicating loudness needs to be assigned to every class for those regions that cover a time interval with more than one class. This value has 5 possible levels represented by the integers in the range from 1 to 5. The annotation is relative to the class or classes with the highest loudness, which

[2] https://github.com/CrowdCurio/audio-annotator
[3] https://wavesurfer-js.org/plugins/regions.html
[4] http://www.speech.kth.se/wavesurfer/
[5] http://www.fon.hum.uva.nl/praat/
[6] http://www.speech.kth.se/hat/
[7] https://github.com/BlaiMelendezCatalan/BAT

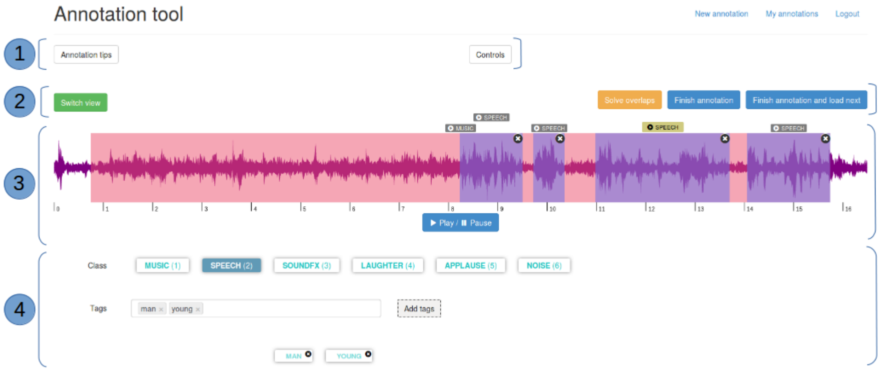

Figure 1: Layout of the annotation tool.

must always receive a salience value of 5. A relative system prevents inconsistencies due to the volume at which the annotator listens to the audio and can be approximated to an absolute annotation using the energy of the audio signal. After having assigned these salience values, the annotation of the current segment is finished and the next one can be loaded. The tool offers the possibility to go back and forth from one phase to the other; however, all salience information is lost when going back to the event identification phase, as events might be modified, and thus, produce different regions in the salience assigning phase.

The annotator is only forced to follow two rules during the annotation process: first, while in the event identification phase, it is not possible to finish the annotation or access the salience assigning phase if there are regions with no assigned class; and second, while in the salience assigning phase, it is not possible to finish the annotation if there are unassigned salience values. The tool will display a warning with the corresponding explanation every time the annotator incurs in one of these violations.

4.2 Annotation interface

We have designed the tool's graphical user interface with three concepts in mind: simplicity, clarity and helpfulness. Regarding simplicity, we want to highlight that, to start annotating, the annotator needs only to select a project and the tool will automatically deliver segment after segment until the annotator decides to stop or it has produced an annotation for every segment of the selected project. Also, the annotation process consists only of two well-defined steps, from and to which the annotator can easily transition. Moreover, the tool includes a set of notifications that warn and guide the user when it accidentally takes actions that would break one of the rules specified in Subsection 4.1.

Unlike other tools such as Sonic Visualizer, BAT has a specific purpose, which allows to promote clarity by displaying only the most essential elements in a balanced and colorful way. As you can see in Figure 1, the interface is organized in rows of elements: the first one (1) contains two buttons that expand a text box with either annotation tips or the list of controls. In the next row (2), at the left side we find the button to switch the audio visualization from waveform to spectrogram and vice versa (green), and at the right side, the button to transition between the two phases of the annotation process (orange) and the buttons to finish the annotation (blue). After that, (3) we have the audio visualization with a button to play and pause its playback. The figure shows one *music* region that overlaps with 4 *speech* regions. The speech region with the yellow label is currently the selected one. (4) The last row shows the classes and tags assigned to the selected region, which can be modified during the event identification phase.

Concerning helpfulness, the tool integrates a few functionalities in the form of keyboard shortcuts and automatic corrections that make the annotation process faster and more comfortable. Regarding keyboard shortcuts, BAT offers the possibility to set the class of the regions, to navigate through and play the audio and to expand the limits of a region to the boundaries of either another region or the entire segment. The helpful functionalities include: first, the prevention of overlaps if the project does not allow them; second, the unification of close region limits to avoid the creation of small overlaps, which are inefficient to solve; third, the inclusion, at each side of the waveform, of a padded zone that cannot be annotated in order to provide more information about the sounds at the beginning and at the end of the segment; and finally, the deletion of small regions when created accidentally by dragging the mouse after a click.

5. EVALUATION

As mentioned in Subsection 4.1, the annotation process has two phases. In each of these phases, the annotator can be either listening to the audio or taking annotation actions such as drawing or deleting events, assigning a class to them or setting the salience value of a region. On one hand, the evaluation is meant to assess the distribution of the total cost in time between the event identification and salience assigning phases, and between playback and annotation actions. On the other hand, we are interested in evaluating the agreement among the different annotators on the assigning of salience values.

We have not measured the total annotation time as it is dependent on the ontology used and the complexity of the selected audio. We could have compared BAT with other existing tools based on this measurement, but we have not done so for the following reason: this kind of evaluation would not be fair if not accompanied by an assessment of the performance of an algorithm trained with the annotations produced by each tool. In other words, what is really important is the trade-off between the annotation time and the algorithm's performance. This assessment is unfeasible

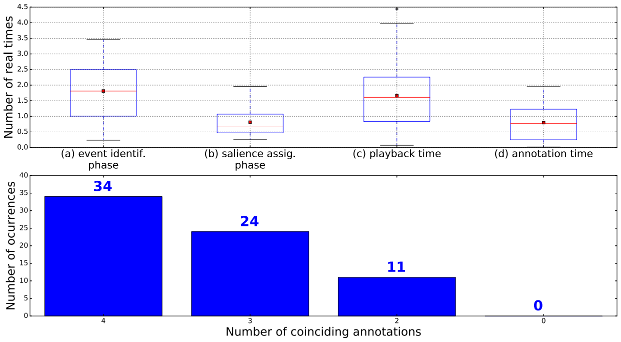

Figure 2: (top) Real times spent in (a) the event identification phase, (b) the salience assigning phase, (c) playback actions and (d) annotation actions. (bottom) Agreement in the assigned salience values.

as it requires the annotation of a training dataset with each tool.

5.1 Evaluation methodology

The dataset used for the evaluation contains 8 recordings with a duration of 1 minute. Each of these recordings belong to one of the following 8 types of broadcast media programs: news, documentary, music, children, series and films, sports, entertainment and debate. We load them to the database with a segment length of 30 seconds, thus producing a total of 16 segments.

For the first part of the evaluation, we had 4 annotators annotate the dataset. The task that they had to carry out was to annotate audio events and their salience using a simple and well-defined ontology containing the *music* and *speech* classes. Before the annotators could start annotating, we described the annotation process as well as the available controls, and let them annotate a few examples until they felt confident using the tool. To evaluate the annotators agreement on the salience values, we annotated the dataset in advance without introducing salience information and then had the same 4 annotators input it. The segments were delivered to the annotators already in the salience assigning phase.

5.2 Evaluation results

The top plot of Figure 2 presents the time distribution between annotation phases and between playback and annotation actions. The vertical axis represents time and it is expressed as number of real times, i.e., the ratio between the time spent annotating a segment and the segments' duration. The time spent in each phase or type of action is not that informative as it depends on the complexity of the ontology and the number of events contained in the segments that are relevant to this ontology. What we want to highlight is the relationship between these times: we observe that, on average, the annotation of the salience of the events takes less than 50% of the time that is devoted to the identification of these events. We also find approximately the same proportion between the time spent listening to the audio and its annotation.

Regarding the agreement on the salience annotation, the bottom plot of Figure 2 shows that for almost 50% of the regions the values assigned by all the annotators are identical. This percentage raises to 84% if we consider the regions where 3 of the annotators coincide and to 100% for the case of 2 annotators. There is no region that has been annotated differently by all annotators. We reach a total agreement of 83% if we average the highest percentage of identical annotations over all the annotations for each region.

6. CONCLUSIONS

In this paper we have presented BAT, an open-source, web-based tool for the annotation of audio events that has as its main feature the easy annotation of the salience of these events and also includes the possibility to generate ontologies and to cross-annotate data. We have focused its evaluation, first, in the distribution of the time cost between annotation phases and between action types; and second, in the agreement of the salience annotations produce by the annotators. The results show that BAT offers fast annotation mechanisms, as the majority of the annotation time is spent listening to the audio, and that its salience annotation system provides the requested information in an efficient and robust manner. In the future, we consider the incorporation of active learning methods to efficiently train algorithms, as well as automatic annotation formulas to reduce the cost of the annotation process.

7. REFERENCES

[1] P. Boersma. PRAAT, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345, 2001.

[2] C. Cannam, C. Landone, M. Sandler, and J. P. Bello. The Sonic Visualizer: A Visualization Platform for Semantic Descriptors from. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR-06)*, pages 324–327, 2006.

[3] P. Herrera, J. Massaguer, P. Cano, F. Gouyon, M. Koppenberger, N. Wack, and U. P. Fabra. Mucosa: a music content semantic annotator. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR-05)*, 2005.

[4] B. Kim and B. Pardo. I-SED : an Interactive Sound Event Detector. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages Pages 553–557, 2017.

[5] D. Krijnders and T. Andringa. Soundscape annotation and environmental source recognition experiments in Assen (NL). *Inter Noise*, 2009.

[6] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. P. Bello, and S. Dixon. Computer-aided melody note transcription using the Tony software: accuracy and efficiency. In *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR)*, 2015.

[7] K. Sjölander and J. Beskow. Wavesurfer - an open source speech tool. In *INTERSPEECH*, volume 4, pages 464–467, 2000.

[8] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes. ELAN: A professional framework for multimodality research. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 1556–1559, 2006.