Gazi Rahman
QAC239
Prof. N&Y
May 20, 2023

QAC239 Final Project: The Laptop that Laughed

Introduction

This paper is an accompaniment to my QAC239 final project, in which I created an algorithm that learns to detect and mark laughter in John Mulaney stand-up comedy videos. At the end of the script, the algorithm functions on a completely fresh video and adds its own laughter clips to time segments where it detects audience laughter. Initially, the idea was inspired by my own interest in stand-up comedy. However, I was inclined to pursue it because of the lack of existing laughter-detection software; when I researched pre-trained models during my brainstorming phase I noticed that, for such a seemingly popular idea, existing algorithms were incredibly intellectually inaccessible. Software like VGGish from Google or similar models from MIT implemented this goal using incredibly convoluted means and a very limited opportunity to peek "under the hood" of the algorithm. Additionally, their descriptions did not have clear indicators of the models' strength at noticing the collective laughter of an audience. Thus, using the Sound Event-Detection unit from QAC239 as a base, I was inspired to ask, "How do we characterize the audio features of crowd laughter in a way that is simple and usable by a computer program? How do we use these characterizations to make audio-based predictions?" On the surface level, onlookers should be intrigued by the sheer wonder of the final product; it is, dare I say, wildly intriguing to listen to a computer begin to laugh. Beyond this, it may also be intriguing for data scientists and programmers to look into this project because it implements laughter detection in a way that is simple, RAM-efficient, and easy to understand for new students and beginners.

Data & Methods

The training-phase data I used was from two John Mulaney stand-up comedy videos; one about a horse in the white house and another about the weirdness of introducing your partner to your parents. I made these particular choices as they were among Mulaney's most viewed clips. At first glance, you might state that this data is rather limited; you'd be right. This is because I went through each video and manually hand wrote every instance of laughter that I deemed valid. A large portion of my time loss came from trying to understand what to flag as laughter in each video; with such limited resources, I had to wonder, "Would it confuse the algorithm if I included scattered chuckles into my classification of laughter? Should I include singular, loud laughs from the crowd? To what extent should I include audio clips that involve overlap in sounds, such as a crowd laughing and clapping at the same time?" These metrics consistently changed throughout the implementation of my project. In the end, I hand-marked occurrences of laughter (1) that involved more than one person, (2) where the laughter was louder than the clapping, and (3) that included occasions where the audience was laughing even while the comedian was speaking. In total, I included and labeled ~8 minutes worth of stand-up comedy. This initial data-marking process was a lesson in the difficulty of manually labeling data. In class, we often used data that was already collected and pre-processed for us. Here, I partook in the experience of playing a clip, rewinding, rewinding, marking it, doubting it, and so on. Once I obtained my data, I was prepared to move on to the actual implementation of my training phase.

The preprocessing phase of my script begins with simple audio extraction. From start to finish, I used yt-dlp to download each video, extracted the audio data from the resulting mp4 files, wrote the audio out into distinct wav files, used ffprobe to find the sampling rates of each file, and loaded them accordingly.

To proceed with my preprocessing phase, I performed feature extraction on each audio file. To begin, I used librosa to obtain the Mel Frequency Cepstral Coefficients, which is a mathematical model with which to characterize audio features. This choice was inspired by convention; the MFCC in audio is frequently used for speech recognition. I hypothesized that there must be overarching vocal patterns in large crowds based on the sum collective of each individual voice. Thus, the convention of using MFCC should be useful in analyzing the features of stand-up comedy audio, be it the comedian or the audience. After attaining the MFCC from each stand-up comedy audio file, I proceeded by extracting the delta of each feature. The delta of a given feature is an approximation of the local slope of the feature using least squares. This choice was similarly inspired by convention; where the MFCC would extract features, the delta offered a means to assess the features and create more data points to feed into my algorithm.

I concluded my preprocessing by labeling indices of laughter in each audio clip and splitting the collective data points into training and testing data. After marking start and end timestamps of laughter, I iterated through each occurrence to create a collection of 0's and 1's representing a binary classification of laughter, with 0 being an absence and 1 being the presence of laughter. This method of labeling was necessary because I needed the dimensions of the labels to line up 1-to-1 with the dimensions of the features. I concluded by using the laughter labels and extracted features to create an 80/20 split between training and testing. I made the decision to use the vast majority of my data to train because I wanted to afford more opportunities for my algorithm to learn; with such limited data and resources I was okay with having a smaller testing phase if it meant my algorithm was better trained. In addition, because I would be creating an output file that I could listen to and personally assess, I concluded that anyone could test my program simply by listening to the final file it outputs.

To conclude, I used a logistic regression model to learn to detect laughter in audio clips. Although I was initially inspired to do so based on my reference, which was the event-detection unit in class, I reinforced this choice with the knowledge that logistic regression would likely be optimal for binary classification. LogReg estimates the probability of a particular event either occurring or not, and it uses a set of independent variables, which in this project were the features I extracted from the audio. After training and testing using the data, I concluded by functioning on a fresh video and marking each predicted occurrence of laughter using the overlay function of pydub. I overlaid each predicted occurrence with a clip of Michael Scott from The Office laughing hysterically.

Results

Figure 1 - Video 1

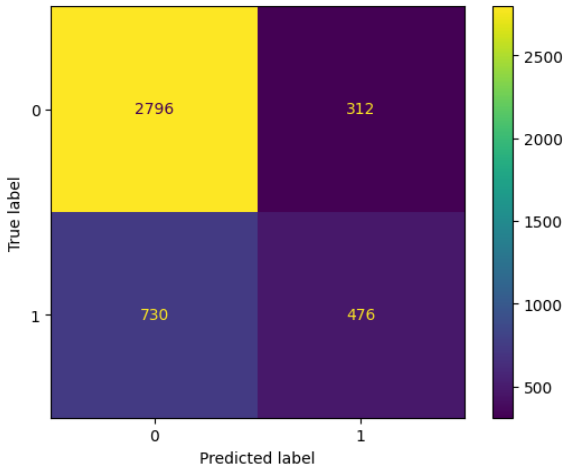
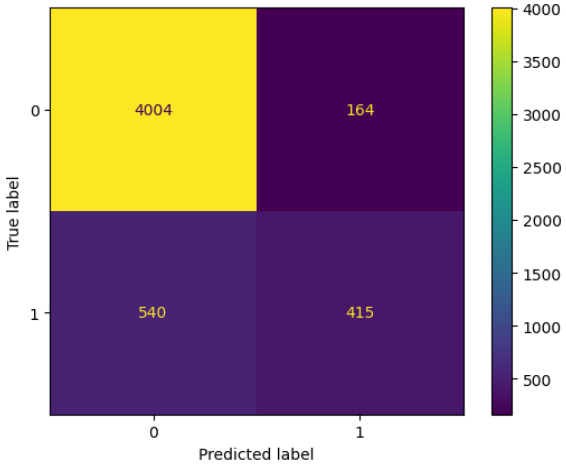


Figure 2 - Video 2



Between the two videos, the average accuracy of the model was 81%. On the first video, the model had an accuracy of 76% whereas on the second video, it had an accuracy of 86%. The 10% difference was striking; I hypothesize that this was caused by different conventions in audio-balancing for stand-up videos. Video 2 is older; in compliance with older conventions, the

audience is as loud as the comedian. Be it due to microphones or due to editing, this means there is a stronger presence of laughter in the audio data for Video 2. In contrast, Video 1 is more modern, it is a more professional production led by Netflix. As a result, using more modern microphones that are better at picking up individual voices, using modern conventions which focus on highlighting the voice of the comedian and muting the voices of the audience, there is a smaller presence of laughter in the raw audio. This is likely a large part of the distinctions in performance; in Video 2, with more vivid laughter, it was likely more easy to tell apart feature-wise from other sounds.

This hypothesis is backed up by the confusion matrices shown above, wherein the figure number corresponds to which video the matrix is representing. In Figure 2, we see a much smaller presence of true positives in contrast to Figure 1, even though Figure 2 had more raw data. Similarly, in spite of having more raw data, Figure 2 had fewer false negatives as well, offering insight about the difference in proportions of performance. While for both videos, the logistic regression model had a similar amount of true negatives, Figure 2 had a much greater amount of true positives. It was interesting to see that the model functioned better on older audio and outdated conventions.

With an 80% accuracy, I am fairly confident in the performance of my model. Regarding my research questions, the results demonstrate that the MFCC and resulting delta features are a strong reference point from which to characterize the resulting audio data of overlapping voices, particularly in the form of laughter. I originally lacked confidence, I thought that maybe the MFCC would end up being better for individual voices. However, we can see that it reveals a great amount of usable information, the likes of which helps an algorithm predict laughter with

notable accuracy. To answer the second part of my question, it is clear that a logistic regression model was a somewhat effective implementation of binary classification.

Throughout the course of this write-up, I've been alluding to the idea of overlaid audio. The climax of my project involves extracting features from a fresh stand-up video that the model hasn't seen. The model uses these features to predict laughter in this new video. Using these predictions, I had the program iteratively overlay the original stand-up comedy video with the laughter of Michael Scott in every instance where it thought laughter was going to begin. The end product offers onlookers an interactive, manual means of assessment by listening to a demonstration of the program's predictions; listening to the output feels like one is watching a computer laugh. I do recognize that I am biased as the creator; the algorithm notices gaps in laughter that don't actually exist, causing it to laugh more than once in one true interval of laughter. However, this unexpected feature could be a pro; the amount of times the algorithm laughs parallels the intensity of the crowd's laughter. When a greater proportion of the audience laughs in a video, the program laughs many times in succession, with the overlaps in laughter emulating a dystopian AI audience.

Conclusion

If this write-up should conclude on any note, it is that readers should refer to the outputted audio from the function, linked previously and here. I used the MFCC and delta features of two audio clips of John Mulaney's stand-up comedy to train and test an algorithm that learns to detect laughter. I'm energized by the completion of this project as it instantiates a simple, achievable, and intellectually accessible implementation of laughter detection. I look forward to this project being interacted with by future students, friends, and learners. In the

future, I'd be interested in ways to make this model more effective. I regret my inability to use more data in my project; I strongly believe that with more audio inputs, the model would be better trained and better able to detect laughter, especially seeing that just two videos taught it this much. I don't regret the lackluster performance of the model regarding Video 1; although it seemed disappointing, I think it is worth noting that an effective model needs to be fed a variety of data. It needs to learn from a diverse array of laughter, even cases where the laughter is muted, quiet, or harder to detect. Another possible direction for the project would be to do multi-class predictions; in addition to detecting laughter, I could potentially have my algorithm detect clapping and have my program overlay clips of comedy with its own clapping as if to say, "Bravo!" I would also be interested to train and test using more comedians. Although I'm not sure the difference it would make since the project was in detecting laughter, which should largely be consistent in its chaos, I think it would be interesting to see the results. Regardless, it was a delight to see the final output of the project. Most of all I'm energized by the capability of Machine Learning. Implementing it in audiovisual formats has been particularly exciting because of the element to which the algorithms interact with the senses of researchers and onlookers alike.

In case of technical malfunctions, the manual links for the code and output are shown below:

https://colab.research.google.com/drive/190BkfGk1uAwvf5vOuVl7GsXCsEBPtr1A?usp=sharing

https://drive.google.com/file/d/1-nZBKfVC4a-8DAt8cqNUjzDeqchD-7fJ/view?usp=sharing