Tampere University

Samuel Lipping

MULTIMODAL AUDIO DATASET CREATION WITH CROWDSOURCING

The case of annotating an audio captioning dataset with AMT

Faculty of Information Technology and Communication Sciences
Bachelors of Science Thesis
May 2019

ABSTRACT

Samuel Lipping: Multimodal audio dataset creation with crowdsourcing
Bachelors of Science Thesis
Tampere University
Signal Processing
May 2019

Creating large multimodal datasets for machine learning tasks can be difficult. Annotating large amounts of data for the dataset is costly and time consuming if done by finding and hiring participants. This thesis outlines a method for gathering multimodal annotations with the crowdsourcing platform Amazon Mechanical Turk (AMT). Specifically, the method in this thesis is made for annotating audio files with five captions and subjective scores for description accuracy and fluency for each caption. The durations of the audio files used in this thesis are uniformly distributed from 15 to 30 seconds. The method divides the whole annotation task into three separate tasks, namely audio description, description editing and description scoring. The editing and scoring tasks were introduced to attempt to fix errors from the previous tasks.

The inputs for the audio description task are the audio files that are to be annotated. The inputs for the description editing task are the descriptions from the audio description task, and the inputs for the description scoring task are the descriptions from the previous tasks. Each audio file is described five times, each description is edited once, and each set of descriptions is scored three times. At the end of the process there are ten descriptions for each audio file and three scores for accuracy and fluency for each description. The scores are used to sort the descriptions and the top five descriptions are used as the final captions for the files. This thesis creates an audio captioning dataset using this method for 5,000 audio files.

Keywords: audio, captioning, AMT, crowdsourcing

TIIVISTELMÄ

Samuel Lipping: Multimodaalisen äänitietoaineiston luominen joukkoistamisen avulla
Kandidaatintyö
Tampereen yliopisto
Signaalinkäsittely
Toukokuu 2019

Suuren multimodaalisen tietoaineiston luominen koneoppimista varten voi olla haastavaa. Suuren datamäärän annotointi on kallista ja vaatii paljon aikaa, jos tämä tehdään keräämällä ja palkkaamalla annotoijia. Tässä työssä esitellään joukkoistamismenetelmä multimodaalisen datan annotointiin käyttäen joukkoistamisalustaa Amazon Mechanical Turk (AMT). Tarkemmin ottaen tässä työssä esitelty menetelmä on luotu keräämään audiotiedostoille viisi kuvausta ja pisteyttämään jokainen kuvaus tarkkuuden ja kielen sujuvuuden perusteella. Tässä työssä käytetyt audiotiedostot ovat pituudeltaan välillä 15-30 sekuntia. Esitelty menetelmä jakaa annotoinnin kolmeen osaan: audion kuvaukseen, kuvauksien muokkaamiseen ja kuvauksien pisteyttämiseen. Muokkaamis- ja pisteyttämisosissa korjataan edellisissä osissa tulleita virheitä.

Audion kuvaukseen annetaan sisääntulona annotoitavat audiotiedostot. Kuvausten muokkaamiseen annetaan sisääntulona audion kuvauksesta saadut kuvaukset. Kuvausten pisteyttämiseen annetaan sisääntulona audion kuvauksesta saadut kuvaukset, kuvausten muokkaamisesta saadut muokatut kuvaukset sekä annotoitavat audiotiedostot. Jokainen audiotiedosto kuvaillaan viisi kertaa, jokainen kuvaus korjataan kerran ja jokaisen audiotiedoston kuvaukset pisteytetään kolme kertaa. Koko prosessin lopputuloksena on kymmenen kuvausta jokaiselle tiedostolle ja kolme pisteytystä jokaiselle kuvaukselle. Kuvaukset järjestetään pisteiden perusteella ja lopullisena tuloksena saadaan viisi parasta kuvausta jokaiselle tiedostolle. Tässä työssä luodaan tietoaineisto, jossa on 5000 audiotiedostoa.

Avainsanat: audio, kuvaus, AMT, joukkoistaminen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

AMT	Amazon Mechanical Turk
API	Application programming interface
CC licence	Creative Commons licence
cli	Command line interface
GUI	Graphical ised interface
HIT	Human Intelligence Task, a single task in AMT

1 INTRODUCTION

When thinking about the functionality of computer programs, one usually thinks of algorithms such as arithmetic and event-based functionality in applications. These are tasks where the parameters of the algorithms are defined by the programmer. Machine learning algorithms, however, are such that the programmer does not define the parameters of the algorithm directly. Rather, as the name implies, the machine learns the parameters on its own based on the data it is given [12]. The machine can learn to model very complex relationships.

Because of the ability to learn complex relationships from examples, machine learning offers the possibility of doing complex tasks, e.g. image captioning, object detection from an image, audio transcription and audio source separation. In object detection, the algorithm detects objects, e.g. humans or cars, from an image by drawing bounding boxes or outlines around the object in the image. In audio transcription, in turn, the algorithm provides subtitles to an audio signal containing speech. In source separation, the algorithm separates sources of sound, e.g. two separate speakers, from an audio signal. This thesis will focus on a topic related to captioning which is the automatic process of generating a description for a multimedia input. For example, image captioning is a task where a description is automatically generated describing the contents of the input image.

Machine learning can be divided into three categories, namely i) Reinforcement learning, ii) unsupervised machine learning, and iii) supervised machine learning. Reinforcement learning is a machine learning approach where a machine learns to act in an environment by exploring. The machine is given possible actions that it can perform. The machine evaluates a reward based on its state in the environment. When performing actions, the machine attempts to maximise the reward it gets. Reinforcement learning can be used e.g. in gaming artificial intelligence [14]. In unsupervised machine learning the machine learns to provide results without any output value examples. Unsupervised machine learning can be used for e.g. clustering or anomaly detection. Unsupervised clustering, where the algorithm identifies groups within some data, can be used to e.g. identify groups of similar customers [21]. Anomaly detection, where the algorithm detects unusual objects from some data, can be used e.g. in fraud detection [1]. Recently, unsupervised machine learning has been used to estimate depth of field, i.e. the distance from the viewer, from a single video stream [19]. Supervised machine learning is an approach where the algorithm learns its parameters based on existing examples consisting of input

values and target output values. The input of the machine learning algorithm could be e.g. an image and the output could be a tag or a sentence describing the image. There are a wide set of fields where supervised machine learning can be applied, even ranging to more traditional fields like physics [2].

Machine learning requires existing data that is used to train the algorithm. This set of training data is often called a dataset. More specifically in the case of supervised learning, the dataset must include both input and output values. Since the algorithm learns from existing examples, it is also the case that the more examples there are to learn from, the better the performance of the algorithm. This is why it is important to have good and large enough datasets when implementing machine learning algorithms. For example, the MNIST dataset of handwritten digits, which has been used in numerous previous research [16, 17], consists of 60,000 training images and 10,000 test images [15].

Other datasets include AudioSet [11] and ImageNet [4]. AudioSet includes YouTube[1] videos and label annotations for the audio content of the videos. ImageNet is a dataset of annotated images, where the annotations classify the images within a hierarchy. The annotations (outputs) in AudioSet and ImageNet represent individual objects or aspects, such as “cat”, “do” or “outside”, of the inputs, but do not say anything about the relationship between the objects. For example, the annotations “dog” and “frisbee” of a video do not say if the dog is catching the frisbee or if the dog has already caught the frisbee. A machine learning task with a more complex output is image or audio captioning. In image or audio captioning the outputs contain information about the relationship between the objects in the image or audio. Among others, the MS COCO caption dataset [3], which contains a set of images and image descriptions, is a dataset that can be used in image captioning. For audio captioning, however, there is no such dataset available. This thesis attempts to address that problem by building an audio captioning dataset and explaining the process by which the dataset was built.

One way to build an audio captioning dataset could be by gathering a group of participants to annotate the audio in the dataset. However, this approach is slow, costly, involves extensive time scheduling, and requires a physical space for the annotation experiment. In recent research, crowdsourcing has risen in popularity as a good way to gather large amounts of data [3, 7, 22]. The whole crowdsourcing task, e.g. gathering data for a dataset, is distributed to multiple participants and has the possibility of global reach. The task can be done by the participants with no restrictions on time or place. Crowdsourcing has also been seen to be a reliable source of data, in addition to being convenient and fast [22]. Inspired by this, we aim to build our dataset using crowdsourcing.

Amazon Mechanical Turk (AMT) is an online crowdsourcing platform used in previous research. It provides access to a global workforce for crowdsourcing tasks. Chen et al. used AMT to create a dataset for image captions [3]. Zaidan & Callison-Burch used AMT to retrieve Urdu-to-English translations [22]. In audio information retrieval, AMT has been used to produce transcriptions from audio clips [7].

[1] `https://www.youtube.com/`

For the audio files that are used in this thesis, the online audio database Freesound [10] was used, which has been used in previous audio processing research [8, 9, 20]. The rest of the document is organized as follows: Chapter 2 will provide the necessary background information for the rest of the thesis. In Chapter 3, the process of building the dataset is discussed. Chapter 4 presents the results and holds the conclusions of this thesis.

2 BACKGROUND

This chapter will go through the necessary background information and terminology used in this thesis. The following sections will explain each necessary component of this thesis and declare the purposes of using these components.

2.1 Dataset

For our purposes, a dataset is a collection of input data and target values. In other words, a dataset D is a collection of N input data (k_n) and target values (v_n),

$$D = \{k_i, v_i\}_{i=1}^{N}$$

For example, in the MSCOCO dataset [3], the input values k_n are digital images and the output values v_n are captions of those images. The MSCOCO dataset is also an example of a multimodal dataset where the input and target values are different modes of media, i.e. image and text. The aim of this thesis is also to create a multimodal dataset from audio files to textual descriptions.

2.2 Crowdsourcing

Gathering data for datasets involves planning the gathering experiment, finding a group of participants to annotate your data, scheduling the annotation procedure, and then having a physical environment where the annotation procedure can take place. To do all of this manually would require a significant amount of time, effort, and money. Planning the experiment, gathering the participants, and scheduling the experiment all add up to the expenses of building the dataset.

Crowdsourcing is a way to eliminate the need for a physical environment to host the annotation procedure. Without the need for a physical environment, the pool of participants is broadened to an essentially global reach, and the need for scheduling is also diminished. In crowdsourcing, the task is delivered to the participants. This means having a digital platform where the task is hosted and then sending participation invitations to the participants, so that they can participate in the experiment at a time which they find most

proper. The hosting platform can be a ready-made one, e.g. AMT or custom-made, as was done by Drossos et al. [6]. With a custom-made platform, however, the group of participants and experiment environment still need to be established.

Using an existing crowdsourcing platform, such as AMT or CrowdFlower[1], eliminates the need for establishing a digital environment. Existing platforms have also been tested and deemed functional by others. Moreover, these platforms come with their own established groups of users and thus eliminates the need for hand-picking a group of participants for the experiment.

2.3 Amazon Mechanical Turk

Amazon Mechanical Turk (AMT) is a crowdsourcing platform that has been estimated to have 100,000-200,000 workers, of which 75% are from the USA [5]. A single self-contained task in AMT, e.g. describing a single audio file, is called a Human Intelligence Task (HIT). The AMT users who publish HITs are called requesters. The AMT users who work on the HITs and provide submissions for them are called workers. A HIT can have multiple assignments associated with it, meaning that multiple distinct workers will work on that specific HIT. For example, if an audio description HIT has five assignments associated with it, the requester of that HIT will receive a submission for the HIT from five distinct workers.

Other parameters associated with HITs include the title and description of the HIT, HIT reward, and qualifications. The title and description of a HIT describe the contents of the HIT to a worker who is browsing the HITs on AMT. The reward of a HIT is the amount of money a worker can gain for working on a single HIT. The worker will receive the reward if their submission is approved by the requester of the HIT. The interaction between HITs, AMT, workers and requesters is displayed in Figure 2.1.

A qualification is an attribute of a HIT that determines who can work on the HIT. Qualifications can be used to control who can participate in the HIT. AMT has their own qualifications for e.g. location, approval rating and number of approved HITs. Requesters can also make their own qualifications. There are also third party services that can be used to limit the amount of HITs one worker can participate in[2].

The layout of a HIT, which is what the worker sees when doing the HIT, is designed with HTML and JavaScript. AMT provides a HIT editor for this purpose. The HITs can be managed via the AMT web GUI, a CLI or an API for various programming languages. In this thesis, the Python API was used because of the limitations of the AMT GUI.

[1] `https://www.figure-eight.com/`
[2] `https://uniqueturker.myleott.com/`

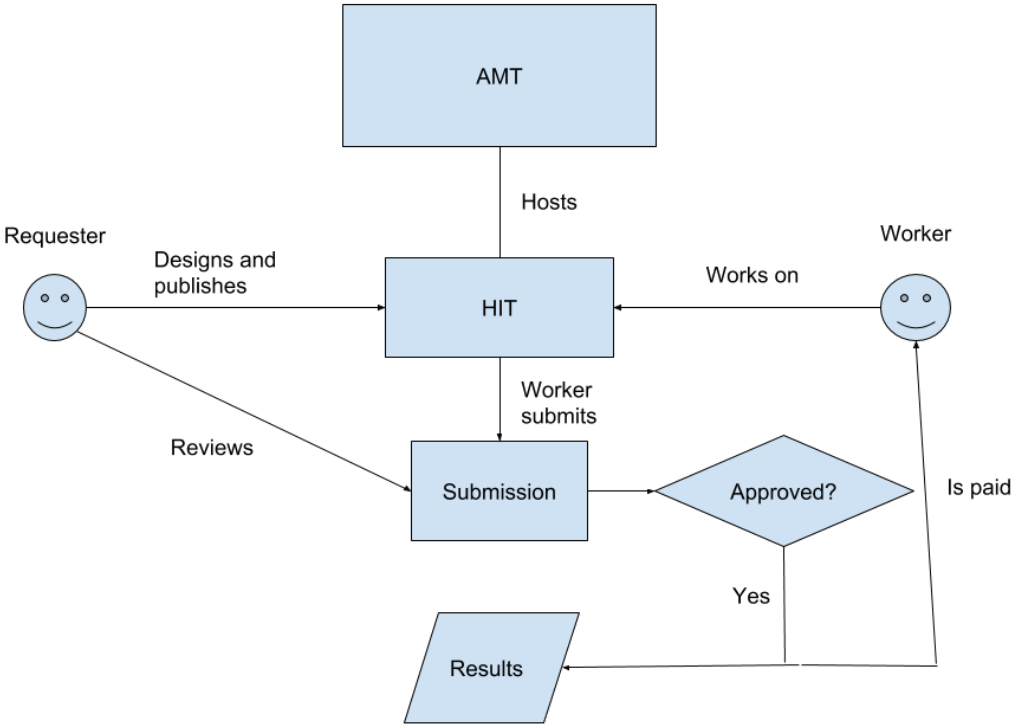

Figure 2.1. *Visual representation of the AMT platform. HITs are designed and published by the requesters. Published HITs are hosted by AMT. Workers can work on and submit submissions for HITs. Submissions are reviewed by requesters. If a submission is deemed to have good quality by the requester, it is approved and the submission becomes a result of the HIT, and the worker of that submission receives payment. If the submission is of bad quality, the submission is rejected, the worker does not receive payment, and the HIT has to be republished to receive a better submission.*

2.4 Audio Captioning

Audio captioning is an example of multimodal translation. In multimodal translation, information is derived from one mode of media to another. In the case of audio captioning, information is transformed from audio to text. Another example of multimodal translation is image captioning where a caption is generated to describe the input image. Another example of multimodal translation from audio is audio transcription. However, audio transcription can be viewed as mapping from speech features to text. In captioning, the algorithm is required to model more abstract relationships, e.g. counting (e.g. a clock striking five times), and enclosures and sizes (e.g. talking in a big room or hall).

2.5 Freesound

Freesound is a database of user-provided sounds that was started in 2005 [10]. The database has active users that have uploaded 36,000 audio files only in 2018[3]. Each audio file in Freesound has the following attributes:

- Username of the user who uploaded the audio file
- Name of the audio file
- Description of the audio file describing the contents of the audio file
- A set of tags indicating the semantic attributes of the audio file (e.g. water, people, crowd, rain)
- A unique ID integer that can be used to identify the audio file in Freesound
- Technical description of the audio file, e.g. sampling frequency, duration, and file size

The tags and descriptions are written by the user who uploaded the audio file. The descriptions provided by the users vary in length and quality, so they cannot be used as audio captions in research. Most importantly for research, every sound in the database has one of the following CC licences attributed to them[4]:

- zero[5]
- attribution[6]
- attribution noncommercial[7]

If a sound file has a zero license, it can be used freely. If it has an attribution license, credit has to be given to the creator of the file, and a link to the license has to be provided. In the case of a noncommercial attribution license the same limitations apply as with the attribution license, but the file may not be used for commercial purposes. Because of these licences, credit is given for all the sounds in the dataset.

[3] https://blog.freesound.org/?p=942
[4] https://freesound.org/help/faq/
[5] https://creativecommons.org/publicdomain/zero/1.0/
[6] https://creativecommons.org/licenses/by/3.0/
[7] https://creativecommons.org/licenses/by-nc/3.0/

3 BUILDING THE DATASET

In this chapter, the process of building the dataset is described. The methods for selecting the audio files is described in Section 3.1 and gathering the captions is described in Section 3.2. The dataset created in this thesis will have 5,000 audio files. Each file ranges from 15 to 30 seconds in duration. Each file will have five descriptions describing their contents.

3.1 Gathering Audio Data Samples

We gathered 12,000 audio files from the Freesound online database and they varied in duration and content. We wanted to have high diversity of the contents of the audio files. For each of those audio files, we gathered the technical information, descriptions, names, and tags from Freesound. To describe the audio file contents, we used the tags of the file. In order to diversify the content in the audio files as much as possible, we attempted to make the distribution of tags as uniform as possible.

Before optimizing the tag distribution, we removed non-descriptive tags from the optimization algorithm. We considered tags to be non-descriptive if they provided no information about the audio content. Such tags included tags describing time or recording equipment, e.g. “July”, “field-recording”, “binaural” , “autumn”, and “contact-mic”. The librosa python library[1] was used to normalize the audio files and to trim the silences from the beginnings and ends. We then removed all audio files with a duration less than 15 seconds from the optimization. After this we had 9,000 remaining audio files.

To make the distribution of tags as uniform as possible, we first randomly permuted our 9,000 files 10^6 times and selected the first 5,000. For each permutation, we calculated the distribution of tags and measured the kurtosis, defined as

$$g = \frac{m_4}{m_2^2}$$

where m_2 and m_4 are moments defined as

$$m_r = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^r$$

[1] `https://librosa.github.io/librosa/`

where $\bar{x}$ is the mean of the distribution [23], and entropy, defined as

$$S = -\sum_{i=1}^{n} P(x_i) log(P(x_i))$$

where $P(x_i)$ is the probability of x_i of the distribution [18]. The kurtoses and entropies were evaluated with the scipy python library[2]. We selected the 5,000 files that exhibited maximum the entropy as well as the 5,000 that exhibited the lowest kurtosis. From these two permutations we selected the one with more uniform distribution of tags. The resulting tag distribution of the dataset audio files is displayed in Figure 3.1. The tags corresponding to the tag indices in the figure are displayed in a table in Appendix B.

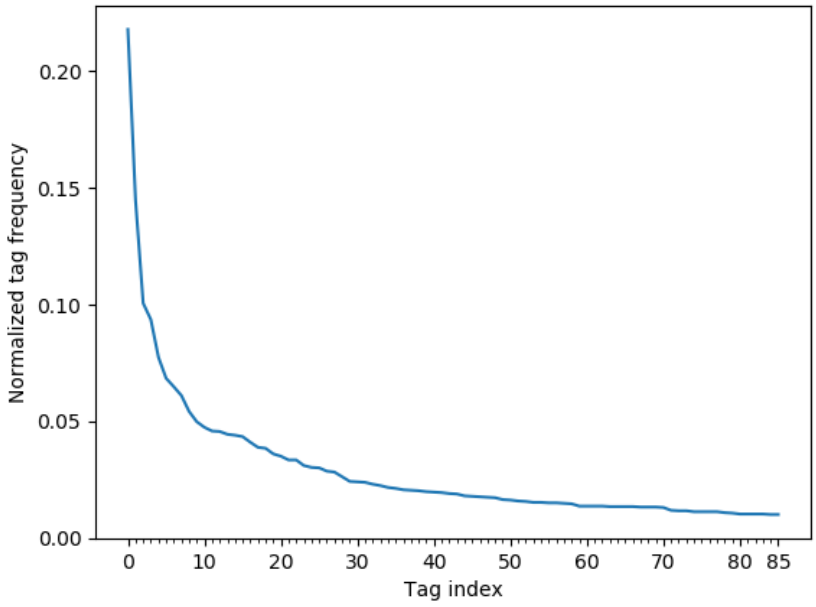

Figure 3.1. *Distribution of the frequency of tags for audio files used in our dataset. The figure shows only the descriptive tags that have a frequency greater than or equal to 0.01*

Once we had a set of 5,000 audio files with an optimized content diversity, we began to sample the audio files that were over 30 seconds in duration while leaving the files between 15 and 30 seconds as they were. We wanted to sample the files so that the resulting samples would contain actual audio content and not only silence. Additionally, we wanted the distribution of all our audio file durations to be uniform. Therefore, we predetermined the durations of the samples so that the duration distribution became even. Having the desired sample durations, we then sampled the files by taking a window of the desired duration from the audio file that maximised the energy of the sample. To measure the energy of the sample we used the RMS value of the samples of the audio signal. Because the sample locations were only based on energy, the results may begin or end in the middle of an audio event. This could result in artifacts in the sample. To minimize the artifacts, we applied half of a 512-sample Hamming window to the beginnings and

[2] `https://docs.scipy.org/doc/scipy/reference/index.html`

ends of the samples as fade-ins and fade-outs. Finally, the librosa python library was used to normalize the audio files in order not to inconvenience annotators with sudden loud files. The duration distribution of the dataset audio file durations is displayed in Figure 3.2.

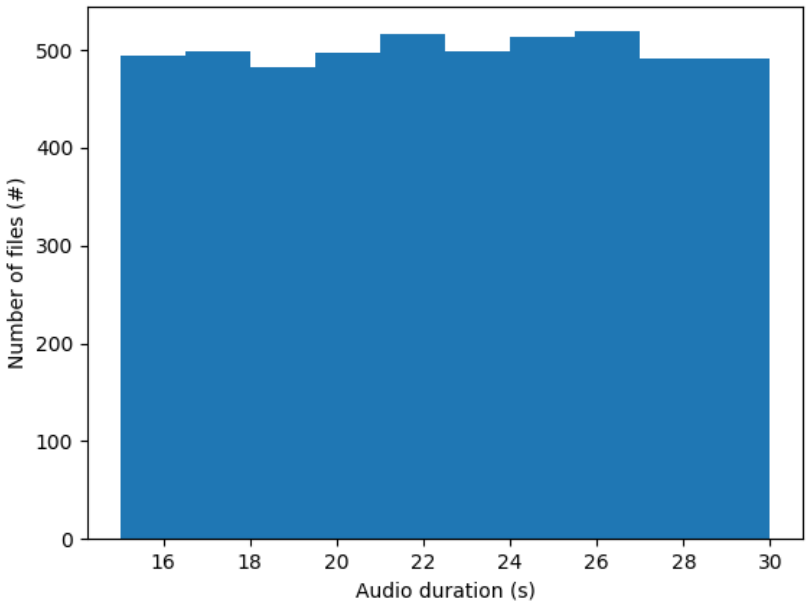

Figure 3.2. *Distribution of duration of audio files used in our dataset.*

3.2 Gathering Annotations with AMT

To gather the annotations for our dataset with AMT, we designed an experiment with three tasks: i) gather five initial descriptions for each audio file, ii) edit the five descriptions from i) for grammatical errors or rephrase the descriptions to get five more descriptions, and iii) score the ten descriptions from i) and ii) based on accuracy and fluency. Finally, gather three of both scores for each audio file. A visual representation of our experiment structure is displayed in Figure 3.3. In the first task of our experiment, we simply gather five descriptions for each audio file.

In the audio description task, there is a risk of receiving submissions with grammar errors, e.g. "*can* is driving by..", awkward sentence structures, e.g. "A *zing callow motion* readable of grinding control.", or similar problems that are easier for humans to detect than for machines. For this reason a second task was introduced to the annotation experiment where a worker reads a submission from the first task and corrects any errors in that sentence, e.g. editing "*An* car is driving.." to "*A* car is driving..". If the sentence does not contain any errors, the worker is tasked with rephrasing the sentence, e.g. editing "A dog is barking in the background as people murmur nearby and birds chirp." to "People chat quietly while a dog barks in the distance with chirping birds.". In this way we are not only

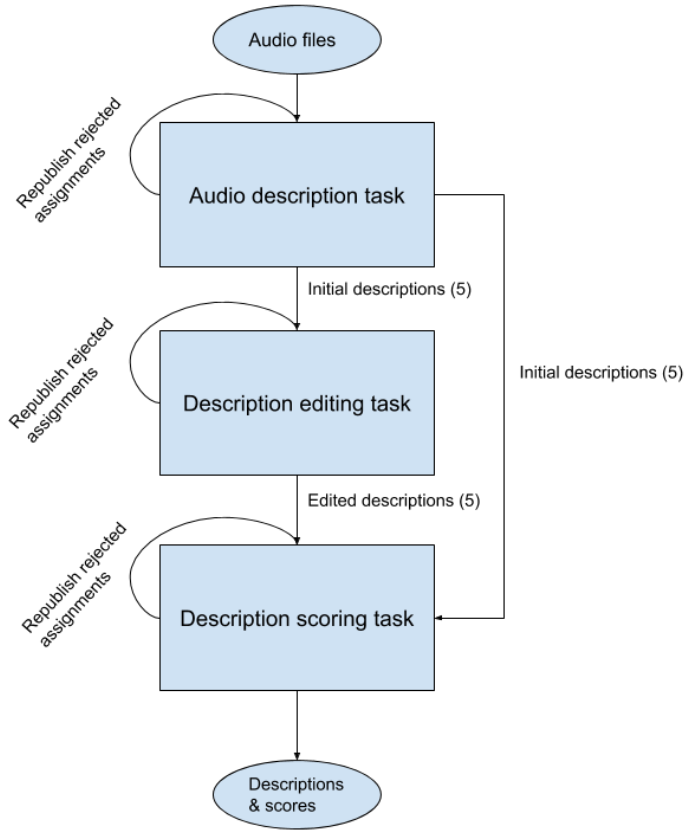

Figure 3.3. *Information flow between the three tasks of the dataset building process.*

likely to get fixed grammar, but also get more diversity in the descriptions for each audio file.

From the audio description and caption editing tasks, we now have multiple descriptions for each audio file. Some descriptions from the first task will still have grammar errors, and some descriptions might not describe the audio file as accurately as others. For this reason, a third and final task was introduced to the gathering process in which a worker listens to the audio file, reads all the descriptions from the previous tasks and scores each description based on accuracy and fluency. In this way the worst descriptions can be weeded out by sorting the descriptions based on the scores and discarding the descriptions with the lowest scores. This three task structure was inspired by previous research with AMT [22].

Since we used the three task structure described above in our experiment, we had to ensure that a worker participating in the editing task would not encounter their own submissions from the audio description task. To this end, we divided our audio files into batches of 500 audio files. The HITs concerning each batch of audio files were published as separate groups of HITs. For each batch, we then created custom qualifications for each batch. We then granted those qualifications to all workers that participated in a batch before publishing the next task for that batch. In this way we could use the qualification system of AMT to prevent participants of that batch from participating in that batch in the following tasks. We also used qualifications to blacklist some workers who con-

sistently made poor submissions or tried to cheat somehow. Additionally, workers were limited to 100 HITs per batch with a third party service. That is, a single worker could participate in a maximum of 100 HITs within a given batch. In this way no one worker would dominate the dataset and there would be diversity in the annotations.

To create and manage the HITs for each task, we used the boto3 Mturk API for Python [13]. The API was used to create HITs, approve and reject assignments, republish rejected assignments, and grant batch participation qualifications. The scripts used for this purpose are publicly available online[3].

In the HITs, the audio files were presented to workers with HTML audio elements. The elements required a streaming link to the audio file. For this reason, our audio files had to be hosted somewhere. To this end, we used Dropbox Professional. The Dropbox python API[4] was used to create the share links required for the annotation HITs.

In the following subsections these three tasks are explained and the layout of each HIT for each task is displayed. Each HIT layout has thorough instructions to improve submission quality and to establish grounds for rejection. Since rejecting submissions negatively affects the AMT workers, there is some javascript functionality in each HIT layout to attempt to prevent workers from making submissions that would lead to rejections. The HTML code for the HIT layouts are publicly available online[5]. Since there are also some criteria that are not easy to detect automatically, a manual review process is also included with each task.

3.2.1 Audio Description (Task 1)

In the first task of our experiment, a worker listens to one of our audio files and writes a description describing the contents of the audio file. We gathered five descriptions for each of our audio files. The inputs for this task are the audio files and the output is five descriptions for each audio file. The layout of this task, i.e. the GUI of the task in AMT, that is visible to the worker is displayed in Figure 3.4.

The layout can be divided into three sections. The first section contains the instructions that the worker has to follow, marked with blue in the figure. Failing to comply with these instructions results in the rejection of the submission from the worker. The two biggest sections of the task instructions are not to assume or add anything to the description that is not clearly present in the audio, and not to add non-descriptive padding phrases, such as “There is”, to the sentence. Since there is also a minimum amount of words for the description defined in the instructions, padding the sentence with such phrases would be counterproductive. The main parts of the instructions for the audio description task are also written in Appendix A. The instructions also contain two audio-description examples,

[3] https://github.com/lippings/amt_hit_management.git
[4] https://www.dropbox.com/developers/documentation/python
[5] https://github.com/lippings/amt_hit_management.git

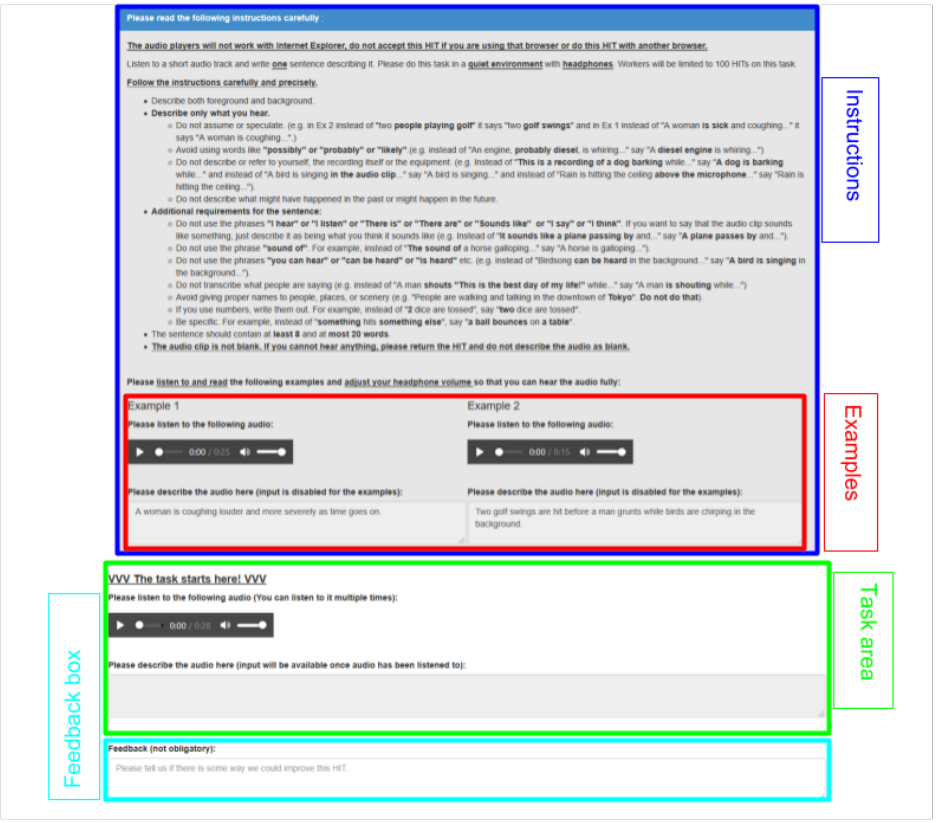

Figure 3.4. *The layout of the task 1 HIT*

marked with red in the figure. The second section is the task area, marked with green in the figure. This is where the worker can listen to the audio file and write the description for the file. The third section in the layout is the feedback box, marked with teal in the figure. Here the worker can submit feedback to us if they find there is something to improve on with the HIT layouts.

Additionally, there is an automatic control that checks for the fulfillment of some of the instructions of the HIT. This control is implemented with a client-side API, using the JavaScript programming language. If the script detects a violation of the instructions, the worker will not be able to submit their description. More specifically, the control detects the phrases disallowed by the instructions and checks if the length of the sentence is within the allowed limits set by the instructions. The control also disables text input until the worker has played the whole audio at least once. This is to ensure that the worker has listened to the audio and knows what to describe before writing anything. Some of the criteria for rejection are not straight forward to detect with a script in the HIT itself. Therefore, before approving submissions, the submissions were also manually reviewed. The manual review process included scanning through the submission descriptions, checking for obvious errors. Such errors included descriptions that made no grammatical sense, copying a phrase from the HIT layout as a description, and using the same description for multiple submissions even when not relevant.

3.2.2 Description Editing (Task 2)

In the second task of out experiment, a worker fixes any grammar issues in the descriptions gathered from the first task. If there is nothing wrong with the sentence, the worker rephrases the sentence. One edited description is gathered for each description from the first task. The inputs to the second task are the outputs of the first task, i.e. five descriptions per audio file in our dataset. The outputs of this task are five additional descriptions that are edited versions of the output descriptions from the first task. After the first and second tasks, we then have ten descriptions for each audio file, five descriptions from both tasks. The HIT layout for this task that is visible to the worker is displayed in Figure 3.5.

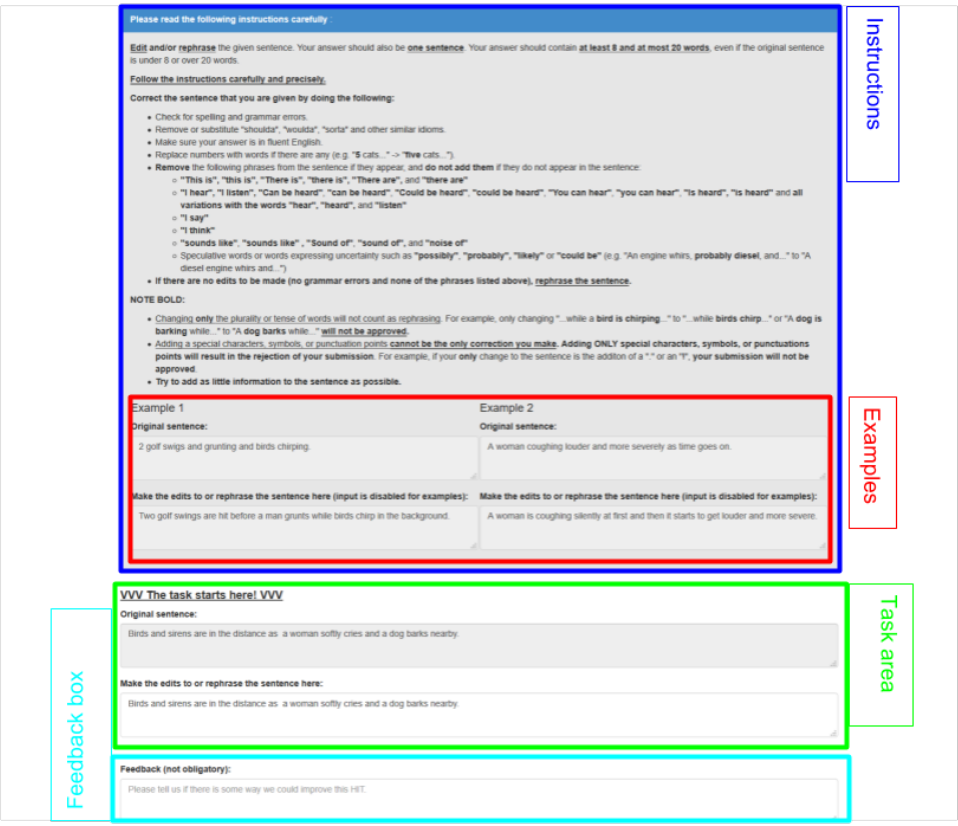

Figure 3.5. *The layout of the task 2 HIT*

The layout of the second task is structured in the same way as that of the first one: The second task layout consists of three sections. The first section is the instructions of the task, marked with blue in the figure. Failing to comply with the instructions resulted in the rejection of the submission of the worker. The two largest sections of the task instructions are to fix the grammar and fluency of the description, and to remove the non-descriptive padding phrases from the sentence. The instructions also included the requirements for rephrasing a description that does not need to be fixed. The main parts of the instructions for the description editing task are also written in Appendix A. This section also includes two description editing examples for the worker, marked with red in the figure. The second

section of the layout in the task are marked with green in the figure. This is where the worker can read the original description and provide an edited version of the description and submit their results. The third section of the layout is a feedback box, similar to the first task.

As was with the first task, this task also has an automatic control to check for the fulfillment of some of the task instructions. If the script detects a violation of the task instructions, the worker will not be allowed to submit their results. More specifically, the control checks for any of the phrases listed in the instructions. Additionally, the control checks if the original sentence and edited sentence are the same, and whether punctuation is the only edit to the sentence. As was with the first task, the results of the second task were manually reviewed before approval. To review the submissions for this task, a python script was first used to detect submissions where the only edit was a plurality or tense change. The script produced some false positives, so these had to be manually checked after running the script. All the submissions were also scanned through manually to check for the same kind of obvious errors as in the first task. Errors detected by the manual review included submissions where the only edit to the description was changing “a” to “an” and vice versa, even when the original description was grammatically correct.

3.2.3 Description Scoring (Task 3)

In the final step of our experiment, a worker scores the descriptions gathered from the first two tasks from one to four, with a higher score meaning a better description. The worker provides scores for accuracy and fluency separately. Three sets of scores were gathered for each set of ten descriptions per audio file. The inputs to this task are the outputs from the first two tasks, i.e. ten descriptions per audio file. The outputs of this task are three pairs of accuracy and fluency scores for each description. The scores were used to sort the descriptions, and the top five descriptions will be given in the final dataset. As a result, each audio file will have five descriptions. The layout for this task that is visible to the worker is displayed in Figure 3.6.

The third task can also be divided into three sections. The first section contains the instructions to the task, marked with blue in the figure. Failing to comply with the task instructions results in rejection of the submission of the worker. The main parts of the instructions are to score the fluency of the descriptions, not the audio, not to include patterns (e.g. 1 2 3 4 1 2 3 4 1 2) in the scores, and not to give a perfect fluency score to a description with typos in it. The second section of the layout is the task area, marked with green in the figure. Here the worker can listen to the audio file that the descriptions describe, and see and score the descriptions themselves. The third section is the feedback box, marked with teal in the figure, similar to that of the first and second tasks. The HIT layout contains short instructions, shown in blue in the figure. The instructions include also some grounds for rejection. Since the scoring is subjective, there is no objective and consistent reason to reject a submission based on the accuracy scoring. Therefore,

Figure 3.6. *The layout of the task 3 HIT*

objective reasons for the fluency scores are used as grounds for rejection, such as giving a description with typos a four in fluency.

As with the previous tasks, this task includes an automatic control to check the fulfillment of some of the task instructions. If the script detects a violation of the task instructions, the worker will not be permitted to submit their results. More specifically, the control checks if all the scores for accuracy or fluency are the same. This is because having the same scores for all descriptions does not provide any means to determine which descriptions are better. The control also prevents the worker from scoring the accuracies of descriptions before listening to the audio file fully at least once. This task also included manual reviewing of the submissions before approval. To review the submissions of this task, a python script was used to flag submissions that gave a four in fluency to a description with obvious typos. The flagged submissions were manually checked for false positives. The submissions were also manually checked for any patterns in the submissions of a single worker.

4 CONCLUSIONS

In this thesis, a scalable crowdsourced method for creating a multimodal dataset was outlined, using AMT. A unique dataset for audio captioning was created with this method. The dataset consists of audio files of environmental sounds, five captions for each sound, and three scores for accuracy and fluency for each caption. The audio files in the dataset are from Freesound, and thus have CC licences attributed to them. The licences only forbid commercial use.

The method described in this thesis divides the audio annotation task into three tasks: audio description, description editing and description scoring. In the audio description task, the AMT workers were tasked with writing a description for the audio file. Five descriptions were gathered for each audio file. In the description editing task, the AMT workers fixed the grammar or rephrased the descriptions from the audio description task. Each description was edited once, resulting in a total of ten descriptions for each audio file after this task. In the final task, namely the description scoring, the AMT workers assigned scores for each description of an audio file based on accuracy, i.e. how well the description describes the audio, and fluency, i.e. how fluent the English is in the description. Each set of descriptions was scored three times. At the end of this task, there were ten descriptions for each audio file, and three scores for accuracy and fluency for each description. The descriptions were sorted based on the subjective scores, and the top five descriptions are the final descriptions for the audio file.

Crowdsourcing offers a more scalable approach to multimodal dataset creation than the traditional approach to recruit annotators. Crowdsourcing brings its own challenges, such as the uncertainty of the quality of the results, but these challenges can be addressed by designing the annotation experiment carefully. Manual review can also be added to the process to further improve the quality of results.

REFERENCES

[1] R. Bauder, R. da Rosa and T. Khoshgoftaar. Identifying Medicare Provider Fraud with Unsupervised Machine Learning. *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. July 2018, 285–292. DOI: 10.1109/IRI.2018.00051.

[2] J. Baxter, A. C. Lesina, J. M. Guay and L. Ramunno. Machine Learning Applications in Plasmonics. *2018 Photonics North (PN)*. June 2018, 1–1. DOI: 10.1109/PN.2018.8438845.

[3] X. Chen, H. Fang, T. Lin, R. Vedantam, S. Gupta, P. Dollár and C. L. Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *CoRR* abs/1504.00325 (2015). arXiv: 1504.00325. URL: http://arxiv.org/abs/1504.00325.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR09*. 2009.

[5] D. Difallah, E. Filatova and P. Ipeirotis. Demographics and Dynamics of Mechanical Turk Workers. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. Marina Del Rey, CA, USA: ACM, 2018, 135–143. ISBN: 978-1-4503-5581-0. DOI: 10.1145/3159652.3159661. URL: http://doi.acm.org/10.1145/3159652.3159661.

[6] K. Drossos, A. Floros and A. Giannakoulopoulos. BEADS: A dataset of Binaural Emotionally Annotated Digital Sounds. *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications* (2014), 158–163.

[7] K. Evanini, D. Higgins and K. Zechner. Using Amazon Mechanical Turk for Transcription of Non-native Speech. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. CSLDAMT '10. Los Angeles, California: Association for Computational Linguistics, 2010, 53–56. URL: http://dl.acm.org/citation.cfm?id=1866696.1866704.

[8] J. Fan, M. Thorogood and P. Pasquier. Emo-soundscapes: A dataset for soundscape emotion recognition. *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. Oct. 2017, 196–201. DOI: 10.1109/ACII.2017.8273600.

[9] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons and X. Serra. General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline. *ArXiv e-prints* (July 2018). arXiv: 1807.09902 [cs.SD].

[10] F. Font, G. Roma and X. Serra. Freesound Technical Demo. *ACM International Conference on Multimedia (MM'13)*. ACM. Barcelona, Spain: ACM, Oct. 2013, 411–412. ISBN: 978-1-4503-2404-5. DOI: 10.1145/2502081.2502245.

[11] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal and M. Ritter. Audio Set: An ontology and human-labeled dataset for audio events. *Proc. IEEE ICASSP 2017*. 2017.

[12] I. Goodfellow, Y. Bengio and A. Courville. Deep Learning. http://www.deeplearningbook.org. MIT Press, 2016, 97. (Visited on 05/03/2019).

[13] A. Inc. *MTurk-Boto 3 Docs*. URL: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/mturk.html#MTurk.Client.create_additional_assignments_for_hit (visited on 03/01/2019).

[14] A. Jeerige, D. Bein and A. Verma. Comparison of Deep Reinforcement Learning Approaches for Intelligent Game Playing. *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. Jan. 2019, 0366–0371. DOI: 10.1109/CCWC.2019.8666545.

[15] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86.11 (Nov. 1998), 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

[16] J. Li, A. Ren, Z. Li, C. Ding, B. Yuan, Q. Qiu and Y. Wang. Towards acceleration of deep convolutional neural networks using stochastic computing. *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. Jan. 2017, 115–120. DOI: 10.1109/ASPDAC.2017.7858306.

[17] B. E. qacimy, M. A. kerroum and A. Hammouch. Handwritten digit recognition based on DCT features and SVM classifier. *2014 Second World Conference on Complex Systems (WCCS)*. Nov. 2014, 13–16. DOI: 10.1109/ICoCS.2014.7060935.

[18] T. Schneider. Information Theory Primer With an Appendix on Logarithms PDF version. (July 2013), 3. DOI: 10.13140/2.1.2607.2000.

[19] S. Shang, H. Wang, P. Zhang and B. Ding. Unsupervised Learning of Depth and Pose Estimation based on Continuous Frame Window. *2018 International Joint Conference on Neural Networks (IJCNN)*. July 2018, 1–8. DOI: 10.1109/IJCNN.2018.8489713.

[20] T. Su, J. Liu and Y. Yang. Weakly-supervised audio event detection using event-specific Gaussian filters and fully convolutional networks. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, 791–795. DOI: 10.1109/ICASSP.2017.7952264.

[21] S. Wu and L. Fu. High-dimensional data clustering for customers with duplicate attribute values. *2016 International Conference on Logistics, Informatics and Service Sciences (LISS)*. July 2016, 1–6. DOI: 10.1109/LISS.2016.7854441.

[22] O. F. Zaidan and C. Callison-Burch. Crowdsourcing Translation: Professional Quality from Non-professionals. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, 2011, 1220–1229. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002626.

[23] D. Zwillinger and S. Kokoska. *CRC Standard Probability and Statistics Tables and Formulae*. Chapman & Hall: New York, 2000.

A SOME INSTRUCTIONS FROM THE AMT TASKS

The instructions for the first task can be divided into two main parts:

- Describe only what you hear. E.g. Do not describe what might have happened in the past or might happen in the future.
- Do not add non-descriptive phrases in the description. E.g. Do not use the phrase “sound of”. For example, instead of “The sound of a horse galloping...” say “A horse is galloping...”.

The instructions for the second task can be divided into three main parts:

- Fix the grammar of the sentence. E.g. Check for spelling and grammar errors.
- Remove non-descriptive phrases from the sentence.
- Requirements for rephrasing the sentence if there is nothing to fix in the sentence. E.g. Changing only the plurality or tense of words will not count as rephrasing.

B AUDIO FILE TAG INDICES

Table B.1. *Tags corresponding to the tag indices on the x-axis in Figure 3.1. The indices visible in the figure are in bold in the table.*

Index	Tag	Index	Tag	Index	Tag
0	**ambient**	**30**	**weather**	**60**	**hit**
1	water	31	open	61	machinery
2	nature	32	bell	62	walk
3	birds	33	waves	63	dog
4	noise	34	field	64	bathroom
5	rain	35	close	65	outdoors
6	city	36	paper	66	squeak
7	wind	37	industrial	67	trees
8	metal	38	spring	68	morning
9	people	39	mechanical	69	ocean
10	**car**	**40**	**steps**	**70**	**metallic**
11	traffic	41	river	71	floor
12	engine	42	stream	72	drops
13	street	43	sea	73	hum
14	atmosphere	44	road	74	station
15	train	45	general-noise	75	leaves
16	machine	46	park	76	liquid
17	footsteps	47	voices	77	outside
18	bird	48	beach	78	white-noise
19	walking	49	running	79	town
20	**kitchen**	**50**	**urban**	**80**	**woods**
21	door	51	voice	81	shower
22	forest	52	insects	82	children
23	wood	53	splash	83	railway

24	crowd	54	radio	84	wet
25	storm	55	animal	85	snow
26	thunder	56	crickets		
27	cars	57	birdsong		
28	motor	58	talking		
29	glass	59	drip		