The appendix is organized as follows: In the first three sections (A - C), we describe a set of experiments and their results. In Section D, we discuss the contrastive model and its training details. In Section E, we evaluate Pengi's performance if the audio encoder is kept frozen. Lastly, in Section F, we state our findings from error analysis and highlight different types of Pengi errors.

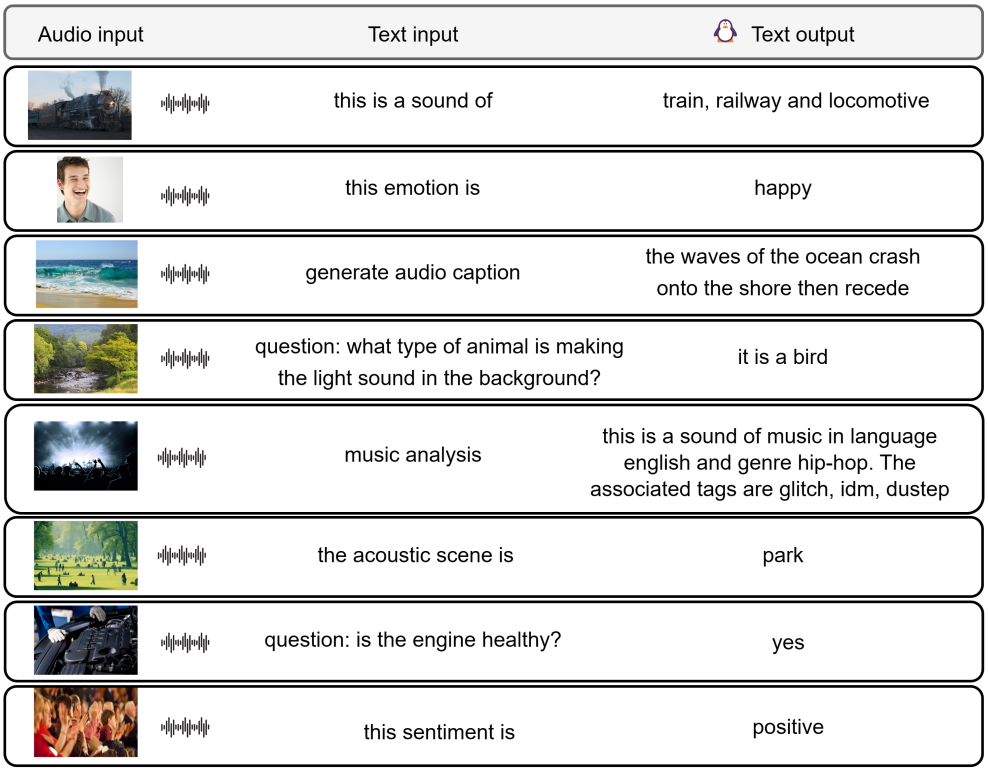

Figure 4: More examples of audio and text prompt input and their corresponding textual responses. Images are for illustration purposes only.

A Additional text input

Pengi takes as input, an audio recording and text, and generates free-form text as output. During inference, an audio encoder a_ϕ and a mapping network m_1 represent each audio recording as a sequence of continuous embeddings. Similarly, a text encoder g_ϕ and a mapping network m_2 does the same for the corresponding text input. Both sequences are combined as a prefix to prompt a pre-trained frozen language model f_θ. The language model generates tokens starting from the prefix.

The text input acts as task induction and helps guide the language model to produce the desired output. Let's take an example of human speech recording. A text input of "generate audio caption" will generate a caption like "a person speaking with a car moving in the background", while a text input of "this sentiment is" will produce a response like "negative". However, there are instances where we want to guide the language model further to answer or complete a specific query we had. We can do this by additional text input. This is depicted in Fig 5. The second text input gets tokenized by the frozen language model's tokenizer and converted into continuous embedding by the frozen language model's embedding function. Therefore, the new prefix consists of a sequence of embeddings associated with audio, first text input, and second text input which originates from the audio encoder, text encoder, and frozen language model's embedding function respectively.

Some examples and the effects of the second text input are shown in Fig 6. Empirically, we have seen the additional second input produces meaningful output only when used with text input of "generate metadata". The examples shown in Fig 6 are cherry-picked. The additional text input often causes Pengi to lose track of the audio data and hallucinate its own text or fall back to frozen language model behavior. It is not clear how to ground the output in audio information when additional text input

is provided. Further investigation in this direction will enable new scenarios including in-context learning.

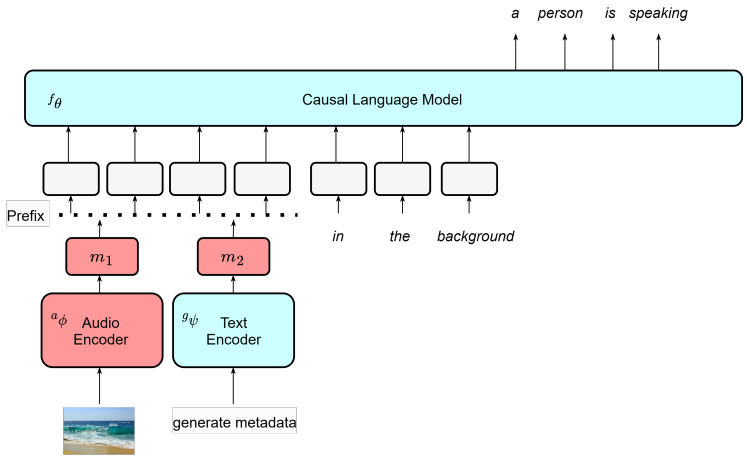

Figure 5: The user can also add an additional second text input and guide the output of Pengi. For example, the user can add "in the background" after the audio and text prefix and Pengi produces the output "a person in speaking". Compared to Fig 2, the output of Pengi changes to what the user has prompted in the second text input which is about background sounds.

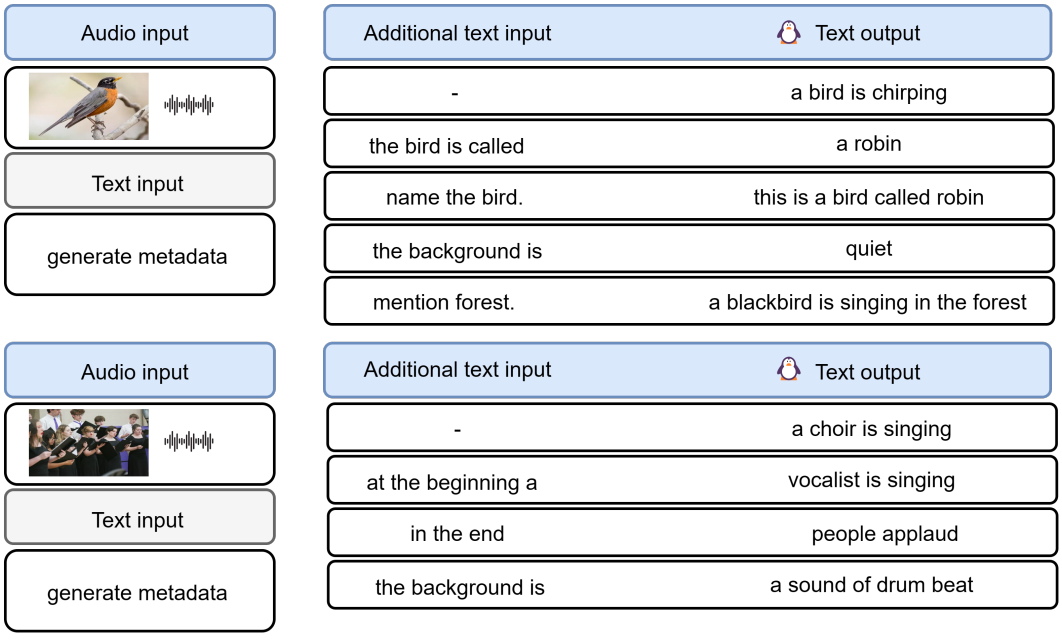

Figure 6: Examples of audio-text input with additional text input and their corresponding textual responses. Images are for illustration purposes only. The '-' symbol indicates additional text input was not used.

B Inferring audio prefix

The audio encoder and text encoder followed by mapping networks, jointly forms the prefix which prompts the frozen language model. To understand more about Pengi's natural language response, we try to interpret prefixes as a sequence of tokens or words. Each prefix embedding is mapped to the highest similarity token from the GPT2 vocabulary [38]. The similarity method used is cosine

similarity. This is possible as the prefix and GPT2 embeddings occupy the same latent space. We use this method on a few examples from the ESC50 dataset [40]. The examples of Pengi's generated output and the inferred audio prefix are shown in Table 10. The interpretations are hard to follow but do contain salient words that are related to audio content. For example, each inferred audio prefix contains words associated with content of audio like babies, thunder, chicken, etc which also appear in corresponding Pengi's natural language output.

One reason interpreted prefix does not have a clear structure is that the mapping network has to do two things at once - comprehend both the audio and text input and guide the fixed language model. Mokady et. al.[38] observed that the interpreted prefix is more comprehensible when GPT2 is also fine-tuned. A similar method can be followed to infer the text input prefix, but we didn't find any interpretable insights there.

Text output	Inferred audio prefix
a baby is crying loudly and loudly	and, the my's the first the and and fixme the the supern the. coma the BST in in improvis the **babies** in in the noises from noises in the (the the and innovative for
a thunder claps and then a thunderstorm hits	and- the bigHUD the the the and as"] the theth P the. **weather** the close andscape. **thunder** in- the Audiostorms interview click in the the and i unsettling,
a rooster is crowing loudly	and at the newone the new the and to OUR the theron the. **chickens** theities the in imperson the **chickens** to in the Audio sitcom. chickens in the (the the, Mumbai the
a bird is singing in the background	and, **the great bird** the first the and and OUR the the number La the in **bird** the one great and photography and **bird** that. in Audio **owl** interview **singing** being: the and I innovative,

Table 10: Examples of Pengi output and their corresponding inferred audio prefix. The input text prompt is "generate audio caption" for all examples. We bold the salient words relating to the input audio and text output.

C Effect of text prompts

The choice of input text prompt changes Pengi's downstream task performance. We analyze the performance of seven of the input text prompts defined in Section 4.1 for downstream tasks. For some tasks, only specific prompts are applicable, for example, *'question: {}'* prompt for Audio Q&A and *'this emotion is'* for emotion recognition. Pengi's performance on each downstream task corresponding to the different input text prompts is shown in Table 11). In summary, we see that the prompt *'generate metadata'* works well on average for close-ended downstream tasks.

D Constrastive Learning model details

We follow and train a CLAP [13] model for the choice of contrastive model used in our experiments. We use transformer-based audio and text encoder. The audio encoder is HTSAT [6] and the text encoder is from CLIP [42]. Both the encoders are followed by a linear transformation called the projection layer. We finetune both the encoder and their projection layers. After contrastive training, the audio encoder and text encoder are used in Pengi.

Consider a batch size of N. Let the audio and text embedding be represented by $E_t \in \mathcal{R}^{N \times d}$ and $E_a \in \mathcal{R}^{N \times d}$. Then the resulting similarity matrix C is:

$$C = \tau(E_t \cdot {E_a}^T) \tag{4}$$

We use the loss function ($\mathcal{L}$) of symmetric cross-entropy: projections

$$\mathcal{L} = 0.5(\ell_{text}(C) + \ell_{audio}(C)) \tag{5}$$

where $\ell_k = \frac{1}{N} \sum_{i=0}^{N} \log diag(softmax(C))$ along text and audio axis respectively.

implementation details. The audio is sampled at 44.1 kHz and is converted to a log Mel spectrogram with 64 Mel bins, a hop size of 320 secs, and a window size of 1024 secs in the range of 50-8000 Hz. We randomly truncate all audio files to 7 seconds in length for HTSAT. All models are trained with Adam Optimiser [28] for 45 epochs with a batch size of 1536 on 20 V100 GPUs. We use a linear schedule with 2000 warmup steps and a base learning rate of 1e-4.

Training data. We train CLAP on the same audio-text pairs (Section 4.1) collected for Pengi.

Results. To verify the training, we check our CLAP's performance on the ESC50 dataset. The results are shown in Table 12.

Downstream Dataset	Text prompts ↑ question: {}	generate audio caption	generate metadata	this is a sound of	this acoustic scene is	this music note is	this emotion is
Clotho Cap.	-	0.2709	-	-	-	-	-
AudioCaps Cap.	-	0.4667	-	-	-	-	-
ClothoAQA	0.6453	-	-	-	-	-	-
ESC50	-	0.8870	0.9195	0.6910	-	-	-
FSD50k	-	0.4676	0.4504	0.4572	-	-	-
US8k	-	0.7185	0.6585	0.5731	-	-	-
DCASE17	-	0.3150	0.3143	0.3506	-	-	-
AudioSet	-	0.1216	0.1230	0.1635	-	-	-
TUT 2017	-	0.2562	0.3525	0.2216	0.1716	-	-
GTZAN Genres	-	0.3230	0.3420	0.3180	-	-	-
GTZAN MS	-	0.9440	0.9606	0.9922	-	-	-
Opera	-	0.2373	0.6229	0.4449	-	-	-
NSynth Instrument	-	-	-	-	-	0.5007	-
NSynth Pitch	-	-	-	-	-	0.8676	-
NSynth Velocity	-	-	-	-	-	0.3728	-
NSynth Qualities	-	-	-	-	-	0.3860	-
RAVDESS	-	-	-	-	-	-	0.1846
CREMAD	-	-	-	-	-	-	0.2032
Vocal Sounds	-	0.5778	0.6035	0.5688	-	-	-
SESA	-	0.5162	0.5402	0.5350	-	-	-
ESC50 Actions	-	0.5277	0.5111	0.4846	-	-	-
Clotho Ret. (T2A)	-	0.0938	-	-	-	-	-
AudioCaps Ret. (T2A)	-	0.1771	0.1407	-	-	-	-
Clotho Ret. (A2T)	-	0.1148	-	-	-	-	-
AudioCaps Ret. (A2T)	-	0.1819	0.1771	-	-	-	-

Table 11: We use different text prompts and observe the performance on downstream tasks. '-' indicates the prompt is not used. The metrics used for each downstream tasks are same as Table 3.

Model	ESC50
Wav2CLIP	0.414
AudioCLIP	0.694
CLAP	0.826
LAION	0.91
CLAP (ours)	0.89

Table 12: CLAP zero-shot performance on ESC50

E Frozen audio encoder

The audio encoder a_ϕ transforms the raw audio input into an audio embedding. We used the audio transformer backbone from CLAP trained in Section D as our audio encoder in our experiments. In Computer Vision, Visual Language Models [38, 2, 35] use an image encoder from CLIP [42] which is frozen throughout experiments. However, there is a magnitude order difference in data collection of image-text vs audio-text pairs. Therefore, for Pengi we train the audio encoder as well. Nonetheless, we report numbers on Pengi's performance if the audio encoder is kept frozen. The results are shown in Table 13. Frozen Pengi underperforms Pengi across all downstream tasks.

Model	Audio Captioning ↑ AudioCaps	Clotho	Audio Q&A ↑ ClothoAQA	Sound Event Classification ↑ ESC50	FSD50K	US8K	DCASE17 Task 4
Frozen Pengi	0.4535	0.2577	0.6395	0.8950	0.4117	0.6319	0.3225
Pengi	**0.4667**	**0.2709**	**0.6453**	**0.9195**	**0.4676**	**0.7185**	**0.338**

Model	Acoustic Scene Classification↑ TUT2017	Music ↑ Music Speech	Music Genres	Instrument Classification ↑ Beijing Opera	Instrument family	Music Note Analysis↑ NS. Pitch	NS. Velocity	NS. Qualities
Frozen Pengi	0.3449	0.9219	0.2550	0.4814	0.2949	0.7131	0.3330	0.3830
Pengi	**0.3525**	0.9688	**0.3525**	**0.6229**	**0.5007**	**0.8676**	**0.3728**	**0.3860**

Model	Emotion Recognition↑ CRE MA-D	RAV DESS	Vocal Sound Classification↑ Vocal Sound	Action Recog.↑ ESC50 Actions	Survei llance.↑ SESA
Frozen Pengi	0.1816	0.1312	0.5371	0.5196	0.5316
Pengi	**0.1846**	**0.2032**	**0.6035**	**0.5277**	0.**5402**

Table 13: The model 'Frozen Pengi' indicates Pengi with audio encoder frozen. The '-' symbol indicates numbers were not available while '✗' indicates that the model cannot support the task. Higher is better for all numbers. The evaluation metric is mAP for FSD50k, AudioSet, and NSynth sonic; F1 score for DCASE17; and SPIDEr for AudioCaps and Clotho captioning. All other downstream tasks use Accuracy.

F Different type of Pengi errors

There are three types of errors that lead to a drop in Pengi's performance. We categorize them into audio concept errors, hierarchy errors, and text-matching errors.

Audio concept errors. These types of errors are when the model gets the base audio concepts wrong. For example, while generating an audio caption, the model predicts it as "a sound of a dog barking in a neighboring field" instead of "a sound of door knocks with cars moving nearby". This indicates the model fails to detect the sound event of a door knock and confuses it with dog barking. These are Pengi model errors stemming from the audio encoder.

Heirarchy errors. The hierarchy error comes from a mismatch between Pengi's model prediction and the target domain classification. For example, in classifying sound events, Pengi predicts the sound as "domestic sounds", however for ESC50, the target classification requires a more fine-grained classification within domestic sounds like Vaccum cleaner, Toilet flush, brushing teeth, etc. If text matching is used for classification, then the model will not be able to categorize "domestic sounds" into any of the fine-grained classes. To solve this error and get a more fine-grained response, we can use improved text prompts or switch to the log-likelihood method.

Text-matching errors. The text-matching errors are the errors that result from the text embeddings or the text-matching method used. This means depending on the text embedding and similarity method used, the performance of Pengi on close-ended tasks will change.