PRETRAINED AUDIO, SENTENCE, AND TAG EMBEDDINGS FOR DCASE CHALLENGE 2022 TASK 6B

Technical Report

Thomas Pellegrini

IRIT (UMR 5505), Université Paul Sabatier, CNRS, Toulouse, France

ABSTRACT

Language-based audio retrieval aims to retrieve audio recordings based on a query formulated as a free-form sentence written in natural language. This technical report presents an automated system for language-based audio retrieval submitted to the DCASE 2022 challenge. The system is based on a pretrained sentence transformer that provides an embedding for the textual query, and an audio transformer pretrained on AudioSet, used to encode the audio recordings as scene embeddings. A linear layer is used to project the audio embeddings onto a subspace supposed to be close to the sentence embeddings' one, thanks to the contrastive loss optimization learning process. Layer-norm and ℓ_2-normalization are used in both the linguistic and acoustic parts of the model. This system was improved by adding to the projected audio embeddings the sentence embeddings of the AudioSet predicted tag names, weighted by their probabilities. This summation was done in the form of a convex combination, with a 0.8 weight on the audio embeddings. Further improvement was obtained by pre-training the system on AudioCaps and fine-tuning on the Clotho development subset. It achieved a mAP@10 score of 0.2402 on the Clotho development-evaluation split, slightly improved to 0.2426 by averaging the distances obtained with two systems[1].

***Index Terms*—** Language-based audio retrieval, Pre-trained representations, PaSST, MPnet, Transformers, AudioSet tags

1. INTRODUCTION

Language-based audio retrieval allows to retrieve audio recordings whose acoustic content is supposed to be close to a textual description given as a query. This task is a cross-modal task by nature, that is rather new compared to its image retrieval task counterpart.

Text-based audio retrieval systems are usually comprised of two parts: a text encoder, responsible for encoding a concise representation of a textual query, and an audio encoder to encode the audio content of the candidate recordings [1, 2]. Both encoders are expected to provide representations embedded in the same subspace, where the textual and audio samples can be compared in terms of a distance.

In this technical report, we present our system that follows this two-part architecture. Experiments are conducted on Clotho v2, in the framework of the DCASE 2022 challenge Task 6b. Details on the dataset and on the task can be found in the challenge baseline report [2].

[1]PyTorch code available: `https://github.com/topel/my-audio-retrieval-dcase2022`

Since Clotho is a rather small dataset, the proposed approach relies on open-source pre-trained large models to extract embeddings, for both text and audio. The learnable model itself is small, comprised of a single linear layer to project the audio embeddings onto the textual embedding subspace. We tried to improve this approach by incorporating information from predicted tags on the audio recordings. Moreover, pre-training a model on a larger dataset, namely AudioCaps, was expected to also help.

2. SYSTEM DESCRIPTION

Our system follows the architecture of the baseline system provided by the organizers, comprised of two parts: a text encoder to encode a caption textual query, and an audio encoder to encode the audio signals.

Our system uses two pre-trained transformers: a sentence transformer to encode the caption query as a single embedding, and an audio transformer to encode the audio recordings. Both transformers were "frozen", the embeddings extracted only once off-line, stored on disk, and used as is as input to our proposed system.

The proposed audio encoder is comprised of a single linear layer (together with layer normalization and ℓ_2-normalization) that projects the audio embeddings onto the subspace of the textual query one. We chose to project the audio embeddings, rather than the textual embeddings, since in a preliminary experiment, it seemed to give better results.

As we will describe in details here-after, the learnable parameters of the proposed system correspond to the audio encoder ones: in the linear layer and in the element-wise adaptive biases and gains of the layer normalization, with a total of 406 558 parameters. If we take into account the transformers, this number of course explodes, reaching 196M parameters, but as already said, the transformers were frozen in our experiments (used offline once).

2.1. Text encoder

The textual queries are encoded as sentence embeddings, obtained by averaging the word embeddings outputted by a sentence transformer. The embeddings, as is common practice, are ℓ_2-normalized. The all-mpnet-base-v2 model [3], further referred to as MPnet, was used to extract these embeddings. MPnet is a transformer of more than 109 M parameters, trained on over a billion pairs of sentences. The embeddings are 768 dimensional dense vectors. Layer normalization [4], followed by ℓ_2-normalization is applied to the embeddings. Layer normalization is used to normalize each sample on its own, with an adaptive bias and a gain, just as in Batchnorm. We found that using the adaptive bias and gain led to overfitting, thus,

we chose the option to not learn them during training. Therefore, our text encoder does not have any learnable parameter.

Another sentence transformer was tested, namely all-MiniLM-L6-v2, but MPnet outperformed significantly MiniLM. MPnet was chosen since it has been reported as the best model for sentence embedding against a selected list of other models [5].

The same text encoder was used in the four submissions to the challenge.

2.2. Audio encoder

Variations on the audio encoder are described here-after.

2.2.1. PaSST-MPnet

In the proposed system, the audio recordings are encoded as scene embeddings, using the Patchout transformer named PaSST [6], pre-trained on AudioSet. We used the "logits" embedding outputted by PaSST, which is a 527-dimensional dense vector, comprised of the logits predicted for the 527 AudioSet event tag classes. Layer-norm is then used, in this case with adaptive bias and gain, followed by a linear layer and a final ℓ_2-normalization. In the result section, we refer to this system as PaSST-MPnet.

2.2.2. PaSST-MPnet-tags

PaSST-MPnet was then improved in a second system by incorporating information from the AudioSet tags: PaSST-MPnet-tags. In this system, the logits embedding is passed through a sigmoid function to obtain probabilities p_c, with $c \in [0, 526]$, the tag AudioSet class index.

The 527-d probability vector is used to weight a new embedding, obtained using MPnet applied on the tag names of the 527 event categories of AudioSet, as defined in AudioSet's ontology:

$$\text{tag\_embed} = \frac{\sum_c p_c \vec{e}_c}{\sum_c p_c} \quad (1)$$

where p_c and $\vec{e}_c$ are the probability and sentence embedding for tag category c, respectively. Examples of AudioSet categories may be a single word or a sequence of words, for instance: "Water", "Rain", "Child speech, kid speaking". The same script as the one used to encode the captions in the text encoder was used to obtain the AudioSet tag embeddings.

The tag embedding is then ℓ_2-normalized and combined to the PaSST audio embedding with a convex linear summation:

$$\text{final\_audio\_embed} = \lambda\, \text{audio\_embed} + (1 - \lambda)\, \text{tag\_embed} \quad (2)$$

where audio_embed is the normalized projected PaSST embedding. This final audio embedding is ℓ_2-normalized before being compared to the caption query embedding.

2.2.3. PaSST-MPnet-tags-AC

In this variant, PaSST-MPnet-tags was first trained on the training subset of AudioCaps [7] for a hundred epochs. After each epoch, the model was evaluated on the Clotho development evaluation subset and the best one were retained, in this case after 78 epochs. As a side note, this model, without fine-tuning on Clotho, reached a 0.1850 mAP@10 score on Clotho. This checkpoint was then fine-tuned on the Clotho development subset.

System	R@1	R@5	R@10	mAP@10
PaSST-MPnet	0.1345	0.3550	0.4823	0.2292
PaSST-MPnet-tags	0.1382	0.3640	0.4854	0.2342
PaSST-MPnet-tags-AC	0.1453	0.3652	0.5001	0.2402
Ensemble (2 models)	0.1481	0.3686	0.4978	0.2426

Table 1: Results on the development-evaluation Clotho subset of the four systems used in our four challenge submissions.

2.3. Experimental setup

All the models were trained with the same triplet loss objective as the baseline system provided by the challenge organizers, except that the margin value is different. We used a margin of 0.4 instead of 1.0, since it seems to make the optimization process easier, and slightly better results were obtained with this value. This value was somehow chosen by looking at the histograms of the anchor dot scores and the impostor scores. The margin is used twice in the objective function: once for the audio impostors, once for the query impostors. The difference in average between the anchor and impostor scores is around 0.8 after training. Other margin values were tested but 0.4 seemed to be the best one.

The models were trained on 30 epochs with minibatches of size 128 audio/query samples, and a learning rate scheduler that divided by two the learning rate if no improvement on the validation loss was observed for five consecutive epochs. After each epoch, the models were evaluated on the development evaluation subset and the best ones were retained. The same seed was used for the three models. For PaSST-MPnet-tags, grid search was performed to choose $\lambda = 0.8$ in Eq.2.

3. RESULTS

The results on the clotho development-evaluation subset of our four submissions to the challenge are reported in Table 1, with the metrics used in the challenge. The fourth one is an ensemble of two PaSST-MPnet-tags-AC models trained with different seeds and without the learning rate scheduler. By ensemble, we mean that the distance scores outputted by the two models were averaged.

Using the tag information is beneficial to the system performance, with a 0.5 absolute improvement in mAP@10. Pre-training on AudioCaps brought an additional 0.6 absolute improvement, and a further slight improvement was obtained with a 2-model ensemble system. The best map@10 result was 0.2426.

4. CONCLUSION

We described briefly the system and methods for Task 6b of DCASE 2022. The proposed system is based on open-source pre-trained transformers to extract textual and audio embeddings. The audio embeddings are projected onto the textual embedding space with a simple linear layer. This system was improved by adding sentence embeddings of the predicted AudioSet tag names. Pre-training the system on AudioCaps and fine-tuning on Clotho was found to be beneficial in terms of the mAP@10 metric. This approach led to a 0.2402 score on the Clotho development-evaluation subset. Since we used the transformers only once offline to extract the embeddings, the size of the proposed model is limited to about 407 k learnable parameters.

5. ACKNOWLEDGMENT

The author would like to thank Hervé Bredin for the fruitful discussions, in particular on contrastive loss learning.

This work was partially supported by the French ANR agency within the LUDAU project (ANR-18-CE23-0005-01) and the French "Investing for the Future — PIA3" AI Interdisciplinary Institute ANITI (Grant agreement ANR-19-PI3A-0004).

We used HPC resources (GPU partition) from the OSIRIM platform, administered by IRIT and supported by CNRS, the Region Midi-Pyrénées, the French Government and ERDF (`http://osirim.irit.fr/`).

6. REFERENCES

[1] H. Xie, O. Räsänen, K. Drossos, and T. Virtanen, "Unsupervised audio-caption aligning learns correspondences between individual sound events and textual phrases," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8867–8871.

[2] H. Xie, S. Lipping, and T. Virtanen, "Dcase 2022 challenge task 6b: Language-based audio retrieval," *arXiv preprint arXiv:2206.06108*, 2022.

[3] https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

[4] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[5] https://www.sbert.net/docs/pretrained_models.html.

[6] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," *arXiv preprint arXiv:2110.05069*, 2021.

[7] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 119–132.