THE NTT DCASE2020 CHALLENGE TASK 6 SYSTEM: AUTOMATED AUDIO CAPTIONING WITH KEYWORDS AND SENTENCE LENGTH ESTIMATION

Technical Report

Yuma Koizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino

NTT Corporation, Japan

ABSTRACT

This technical report describes the system participating to the Detection and Classification of Acoustic Scenes and Events (DCASE) 2020 Challenge, Task 6: automated audio captioning. Our submission focuses on solving two indeterminacy problems in automated audio captioning: word selection indeterminacy and sentence length indeterminacy. We simultaneously solve the main caption generation and sub indeterminacy problems by estimating keywords and sentence length through multi-task learning. We tested a simplified model of our submission using the development-testing dataset. Our model achieved 20.7 SPIDEr score where that of the baseline system was 5.4.

Index Terms— Audio captioning, sequence-to-sequence model, keyword estimation, acoustic event/scene estimation.

1. INTRODUCTION

This technical report describes the system participating to the Detection and Classification of Acoustic Scenes and Events (DCASE) 2020 Challenge, Task 6: automated audio captioning [1]. Automated audio captioning (AAC) is an intermodal translation task when translating an input audio into its description using natural language [2–6]. In contrast to automatic speech recognition (ASR), which converts a speech to a text, AAC converts environmental sounds to a text. This task potentially raises the level of automatic understanding of sound environment from merely tagging events [7,8] (e.g. alarm), scenes [9] (e.g. kitchen) and condition [10] (e.g. normal/anomaly) to higher contextual information, for example, "*a digital alarm in the kitchen has gone off three times.*"

Our submission focuses on solving the indeterminacy problems in AAC which were tackled in our previous studies [3, 11]. This indeterminacy can be broadly divided into the indeterminacy in (i) word selection [11] and (ii) sentence length [3]. The first problem is caused by that one acoustic event/scene can be described with several words, such as {*car, automobile, vehicle, wheels*} and {*road, roadway, intersection, street*} [11]. The second one is caused by that a sound can be explained in either short or long sentences, such as "noisy car sounds," or "a lot of cars are driving on the roadway and there are very loud engine noises" [11]. Such indeterminacy leads to a combinatorial explosion of possible answers, making it almost impossible to estimate the ground-truth and difficulty in training an AAC system.

Our strategy for solving these problems is to simultaneously estimate keywords and sentence length through multi-task learning framework. Figure 1 shows the overview of our system. The pre-processing stage involves rule-based keywords and sentence length extraction from the caption and metadata. The captioning DNN has
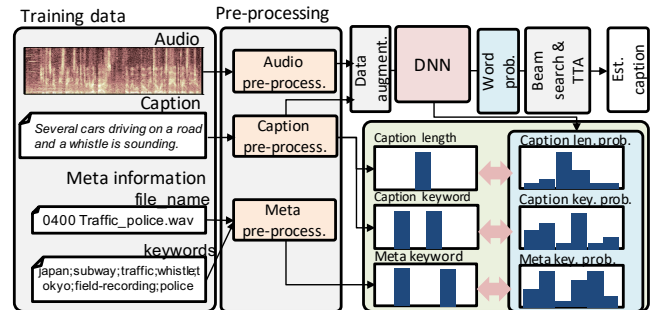

Figure 1: System overview.

keyword estimation and a sentence length estimation branches, and estimates the ground-truth caption by integrating these results.

2. SYSTEM DESCRIPTION

This section describes the detail of our system. Since this paper is a technical report, we focus on describing the detailed implementation of the system. Effectiveness of each modules will be discussed in the workshop paper through ablation studies.

2.1. Pre-processing

Audio pre-processing: As acoustic feature, we used three log-mel-spectrograms calculated from the time-domain input audio x. The first one was the log-mel-spectrogram of the input audio $S \in \mathbb{R}^{F \times T_s}$, where F and T_s are the number of mel-filterbanks and time-frames. The second and third ones were that of the harmonic-percussive source separation (HPSS) outputs, $H \in \mathbb{R}^{F \times T_s}$ and $P \in \mathbb{R}^{F \times T_s}$. These three spectrograms were concatenated on the channel dimension $X \in \mathbb{R}^{3 \times F \times T_s}$.

The hyper-parameters of the audio pre-processing are as follows. All audio samples were down-sampled at 22.05 kHz. The window- and hop-size of short-time Fourier transform (STFT) were 4096 and 2048 points, respectively. The number of mel-filterbank was $F = 64$. The hyper-parameters of the HPSS were default one of `librosa.decompose.hpss` [12].

Caption pre-processing: All captions were tokenized using the word tokenizer of the natural language toolkit (NLTK) [13] while removing punctuation. All tokens in the development dataset were then counted, and words that appeared more than five times were appended in the word vocabulary. The vocabulary size was $C^{\mathrm{cap}} = 2144$, which includes BOS, EOS, PAD, and UNK tokens. In addition, the sentence length L of each caption was counted. Also,

caption keywords $\boldsymbol{k}^{\text{cap}} = \{k_i^{\text{cap}}\}_{i=1}^{K_c}$ was extracted using the keyword vocabulary which is discussed below.

Meta pre-processing: Meta keywords were extracted from the `file_name` and `keyword` provided in the metadata csv file, using a keyword vocabulary which was manually created beforehand. The procedure of creating the keyword vocabulary is as follows. First, `file_name` and `keyword` were split at places at space and punctuation. Next, words that seem to be nouns, verbs, adjectives, and adverbs were converted to its lemma. Finally, all lemmas were counted, and lemmas that appeared more than ten times were appended in the keyword vocabulary, which is a hash table that maps the original word to its lemma. The vocabulary size was $C^{\text{key}} = 421$. The keyword vocabulary was used to extract meta keyword $\boldsymbol{m} = \{m_k\}_{k=1}^{K_m}$ and caption keyword $\boldsymbol{c}$. Note that the procedure for creating the keyword vocabulary can be automated by using the part-of-speech (POS)–tagger and the WordNet Lemmatizer of the NLTK, however, we did this manually because their use is prohibited in this task.

2.2. Data augmentation

TF-IDF-based sample selection and data augmentation: Since the target metrics of this challenge is SPIDEr, we need to accurately predict captions which include low frequent words and topics. To deal with word and topic bias in the training dataset, we adopted two tricks for training sample selection based on inverse document frequency (IDF), and one trick for data augmentation based on term frequency (TF)–IDF [15].

The first trick is for selecting an audio sample $\boldsymbol{x}$ from the training dataset. First, we concatenated the five ground-truth captions corresponding to each $\boldsymbol{x}$ in the training dataset, and used as a "sentence". Then, we calculated IDFs for all words in all sentences, and calculated the average IDF of each sentence. Finally, each average IDF was normalized by the sum of the average IDF. We regarded the normalized IDF as the parameter of the Categorical distribution, and selected $\boldsymbol{x}$ based on this probability.

The second trick is for selecting a ground-truth caption $\boldsymbol{w}$ from five captions corresponding to the selected $\boldsymbol{x}$. The basic strategy was the same as the first trick. First, we calculated IDFs of all words in the five captions. Here, note that the document was the five captions in contrast to the first trick. Then, we calculated the normalized IDF and used as the parameter of Categorical distribution, and selected the target caption $\boldsymbol{w}$ based on this probability.

Finally, we adopted the third trick which is the TF-IDF based word replacement [15] to augment text data.

Random data cropping: To train our captioning DNN using minibatches, we adjusted the input length of audio sequence and text sequence using random cropping and padding. We set the input length of audio to 20 seconds ($T = 216$), and the number of words is $N = 20$. Thus, the inputs of the captioning DNN were $\boldsymbol{X} \in \mathbb{R}^{3 \times F \times T}$ and $\boldsymbol{w} = (w_1, ..., w_N)$. For $\boldsymbol{X}$ whose T_s was greater than T, a random crop was performed so that the time-length was T, and shorter ones were applied zero-padding. Similarly, if the sentence length was greater than N, words after the N-th word were cropped, and PAD tokens were added for shorter ones.

Mix-up: After adjusting input length T and N, we used the mix-up data augmentation. First, we drew a mixing parameter β from a beta distribution as $\beta \sim \text{Beta}(0.4, 0.4)$ where $\sim$ is sample drawing from the right-hand distribution. Then, two audio samples were mixed by multiplying β and $(1 - \beta)$, respectively. Since text inputs
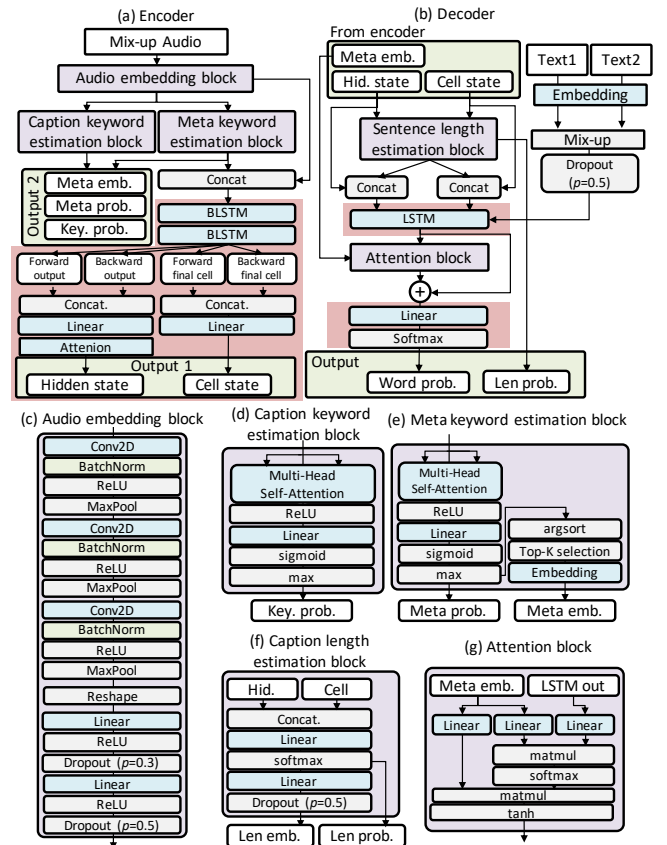

Figure 2: Network architecture of captioning DNN.

a set of class labels, direct mixing of $\boldsymbol{w}$ is not suitable. Thus, we mixed the embedded word tokens by multiplying the mixing parameters.

2.3. Model description

Figure 2 shows the network architecture of the captioning DNN. The pink area in Fig. 2 is a basic sequence-to-sequence (Sec2Sec)–based captioning model [14] using bidirectional long short-term memory (BLSTM)–LSTM. The encoder BLSTMs outputted the initial hidden state $\boldsymbol{h} \in \mathbb{R}^D$ and cell states $\boldsymbol{c} \in \mathbb{R}^D$ of the decoder LSTM, where $D = 120$ was the hidden dimension of the whole network. Then, the decoder LSTM estimated posterior probability of n-th word given the audio signal $\boldsymbol{x}$ and 1st to $(n - 1)$-th words $p(w_n | \boldsymbol{x}, w_{1,...,n-1})$ by using embedded word tokens. In our submission, to solve the indeterminacy problems in AAC, we additionally used sub-blocks for keyword and sentence length estimation. The following describes these sub-locks in detail.

Audio embedding block $\mathcal{A}$: The input audio $\boldsymbol{X}$ was first passed to this block. This block embedded $\boldsymbol{X}$ into a feature space as $\boldsymbol{A} = \mathcal{A}(\boldsymbol{X}) \in \mathbb{R}^{D \times T_a}$. As shown in Fig. 2 (c), this block consisted of three convolutional neural network (CNN)–blocks and two fully connected (FC)–blocks. The kernel size, stride, padding, and number of output channels of CNN were 3, 1, 1, and 64 for all CNN layers, respectively. The kernel size and stride of the 2D maxpooling were 2 and 2, respectively. Then, the output of CNN-blocks $64 \times F_a \times T_a$ was reshaped in to $64 F_a \times T_a$, where $F_a = \frac{F}{2^3} = 8$

and $T_a = \frac{T}{2^3} = 27$, respectively. The reshaped output was passed to the first FC layer which converts $\mathbb{R}^{64F_a \times T_a}$ to $\mathbb{R}^{D \times T_a}$. Finally, the second FC-layer outputted $\boldsymbol{A} \in \mathbb{R}^{D \times Ta}$.

Caption keyword estimation block $\mathcal{C}$: This block estimated caption keyword probabilities of each keyword $\boldsymbol{p}^{\text{cap}} \in [0, 1]^{C^{\text{key}}}$ from $\boldsymbol{A}$ as $\boldsymbol{p}^{\text{cap}} = \mathcal{C}(\boldsymbol{A})$. We expected that this block guides the audio embedding block so that its output includes information of the keywords of the ground-truth caption. As shown in Fig. 2 (d), this block consisted of a muti-head self-attention (MHSA) layer and a FC layer. The number of heads of MHSA was 4. The output shape of MHSA and FC layer were $D \times T_a$ and $C^{\text{key}} \times T_a$, respectively. Since the caption keyword has no time-labels, we aggregated the output by taking a maximum value in the time direction and outputted caption keyword probabilities of each keyword $\boldsymbol{p}^{\text{cap}}$.

Meta keyword estimation block $\mathcal{M}$: This block estimated meta keyword probabilities $\boldsymbol{p}^{\text{meta}} \in [0, 1]^{C^{\text{key}}}$ and its embedding $\boldsymbol{M} \in \mathbb{R}^{D \times K_m}$ from $\boldsymbol{A}$ as $\{\boldsymbol{p}^{\text{meta}}, \boldsymbol{M}\} = \mathcal{M}(\boldsymbol{A})$. As shown in Fig. 2 (e), the base architecture was the same as the caption keyword estimation block, which consisted of MHSA and FC layers. The base architecture outputted meta keyword probabilities of each keyword $\boldsymbol{p}^{\text{meta}}$. To embed the estimated meta keywords into the feature space, first, we used the `argsort` function which returns the index on which $\boldsymbol{p}^{\text{meta}}$ sorts in descending order. Then the top $K_m = 15$ indices were selected as the estimated meta keyword $\{\hat{m}_k \in \mathbb{N}\}_{k=1}^{K_m}$. Finally, these indexes were passed to the embedding layer to obtain the estimated meta keyword embedding $\boldsymbol{M}$. After this block, $\boldsymbol{A}$ and $\boldsymbol{M}$ were concatenated as $(\boldsymbol{M}, \boldsymbol{A}, \boldsymbol{M})$, and it was passed to BLSTMs.

Sentence length estimation block $\mathcal{L}$: This block estimated the sentence length probability $\boldsymbol{p}^{\text{len}} \in [0, 1]^{L^{\max}}$ and its embedding $\boldsymbol{l} \in \mathbb{R}^{D_l}$ as $\{\boldsymbol{p}^{\text{len}}, \boldsymbol{l}\} = \mathcal{L}(\boldsymbol{h}, \boldsymbol{c})$, where $L^{\max} = 20$ is the maximum sentence length that we assumed. First, $\boldsymbol{h}$ and $\boldsymbol{c}$ were concatenated, and the first FC layer estemited $\boldsymbol{p}^{\text{len}}$ from the concatenated feature. Finally, $\boldsymbol{p}^{\text{len}}$ was passed to the second FC layer, and outputted $\boldsymbol{l}$. After this block, $\boldsymbol{l}$ was concatenated to $\boldsymbol{h}$ and $\boldsymbol{c}$, and used as the initial hidden and cell state of the decoder LSTM.

Attention block: Before calculating $p(w_n | \boldsymbol{x}, w_{1,\dots,n-1})$ using the final FC layer, this block integrated the output of the LSTM $\boldsymbol{H} \in \mathbb{R}^{(D+D_l) \times N}$ and $\boldsymbol{M}$ and outputs $\boldsymbol{M}' \in \mathbb{R}^{(D+D_l) \times N}$ by using three FC-layers, like an MHSA with a single head. Then the tanh activation was applied to $\boldsymbol{M}'$. Finally, it is added to $\boldsymbol{H}$ as $\boldsymbol{H} + \tanh(\boldsymbol{M}')$ and passed to the final FC-layer to estimate $p(w_n | \boldsymbol{x}, w_{1,\dots,n-1})$.

2.4. Loss functions

In order to train encoder/decoder and sub-blocks simultaneously, we designed loss function as a sum of multiple losses functions. In addition, since the input audio and text were augmented by the mix-up, the cost function was also calculated using the mix-up; each loss was calculated for each of the two original label data and mixed using mixing-parameters β and $(1 - \beta)$. The following describes these loss functions in detail.

Word estimation loss: For word prediction, we used the cross-entropy loss between w_n and $p(w_n | \boldsymbol{x}, w_{1,\dots,n-1})$. To avoid over-fitting, we used label smoothing where smoothing factor was 0.1.

Caption/meta keyword estimation loss: The weighted binary-cross entropy was used as the loss function for both caption/meta keyword estimation block as

$$-\frac{1}{C^{\text{key}}} \sum_{i=1}^{C^{\text{key}}} \lambda_i z_i \ln p_i + \gamma_i (1 - z_i) \ln(1 - p_i). \tag{1}$$

Note that for all variables, we omitted the superscripts $^{\text{cap}}$ and $^{\text{meta}}$ which indicate whether the variable belongs to caption keyword or meta keyword. Here, the meanings of each variable are followings: z_i is 1 when ground-truth keyword set includes i-th word and 0 otherwise, p_i is the i-th value of the estimated posterior vector $\boldsymbol{p}$, and λ_i and γ_i are the weight for i-th keyword as $\lambda_i = (p(z_i))^{-1}$ and $\gamma_i = (1 - p(z_i))^{-1}$, respectively, where $p(z_i)$ is the prior probability of the i-th keyword calculated by

$$p(z_i) = \frac{\text{\# of } c\text{-th keyword in training samples}}{\text{\# of training samples}}. \tag{2}$$

To balance this loss and other losses, we multiplied a weight $(1 - 10^{-4})^s$ to this loss, where s is the number of training steps.

Sentence length estimation loss: We used the softmax cross entropy between L and $\boldsymbol{p}^{\text{len}}$ as the loss for the sentence length estimation block. To balance this loss and other losses, we multiplied a weight 10^{-2} to this loss.

Keyword co-occurrence loss: In order to prevent the decoder outputs the words which are obviously not related to the meta keywords, we used the keyword co-occurrence loss between words in a caption and its meta keywords. For example, when meta keywords are {*car, sing, bird*}, words not related to the keywords such as {*people, children, talking, talk, speak*} may not be included in the correct caption. To prevent the decoder outputs such words, we adopted a penalty based on the decoder outputs $p(w_n | \boldsymbol{x}, w_{1,\dots,n-1})$.

Before training, we created a hash-table of the co-occurrence lists; the keys of the hash-table are all keywords in the keyword vocabulary, and the element of each key is a list of the words that have co-occurred with the keyword in the training dataset. For example, in the case of meta keywords are {*car, sing, bird*} and ground-truth captions are {*Cars are driving and birds are singing, A car passes by while birds are chirping and singing*}, {*cars, are, driving, and, birds, singing, a, passes, by, while, chirping*} are added to the co-occurrence lists of *car, sing,* and *bird*. Then, in the training step, we added penalties of the decoder outputs to the whole loss value as

$$\frac{1}{C^{\text{cap}}} \sum_{n=1}^{N} \sum_{i=1}^{C^{\text{cap}}} |b_i \cdot p(w_n = i | \boldsymbol{x}, w_{1,\dots,n-1})|, \tag{3}$$

where $b_i \in \{0, 1\}$ a binary mask where $b_i = 1$ when none of all co-occurrence lists of the ground-truth meta keywords includes the i-th word, and otherwise $b_i = 0$.

2.5. Beam search and test time augmentation

We used the beam search decoding for the word decision process from $p(w_n | \boldsymbol{x}, w_{1,\dots,n-1})$. The beam size was 5, and n-gram blocking size was 2, i.e. a hypothesis in a beam was discarded if there was a bi-gram that appeared more than once within it. In addition, we used test time augmentation (TTA) for audio input. This is because the audio input was randomly cropped for limiting the time-length as $T = 216$ in training phase. If the length of audio input is changed in testing phase, it may have a bad influence on the batch normalization layers. Therefore, in testing-phase, we also randomly cropped

Table 1: Experimental results on development-testing dataset.

Model	B-1	B-2	B-3	B-4	CIDEr	METEOR	ROUGE-L	SPICE	SPIDEr
Baseline	38.9	13.6	5.5	1.5	7.4	8.4	26.2	3.3	5.4
Model1	52.6	33.5	22.4	14.6	30.1	14.7	34.7	9.0	19.5
Model2	51.2	32.1	21.3	14.1	29.7	14.5	33.9	9.1	19.4
Model3	51.7	33.0	22.0	14.5	30.0	14.7	34.3	8.6	19.3
Model4	53.0	33.7	22.4	14.5	30.2	14.8	35.2	9.1	19.6
Ensemble	**53.7**	**34.8**	**23.5**	**15.6**	**31.9**	**15.2**	**35.9**	**9.4**	**20.7**

and zero-padded the audio input so as to $T = 216$. We generated five input audios by this process, and took the average of five outputs of the decoder.

2.6. Training hyper-parameters

We used the AdamW [17] optimier with a constant learning rate 10^{-4}. The minibatch-size was 48. We randomly splitted $2893 + 1045$ samples in the development dataset into 3842 training samples and 96 validation samples. We used a DNN whose validation score was the best while 300 epochs training.

2.7. Submitted systems

We used a model ensemble to output the final results; each model in the ensemble outputted $\ln p(w_n|\boldsymbol{x}, w_{1,...,n-1})$, and we took the average of all log-probabilities in the beamsearch phase. The four submitted results were four types of different combinations of following models.

Model1 The base model described in Sec. 2.3.

Model2 Modified model of Model1. A FC layer was used instead of the MHSA layer in the caption keyword estimation block.

Model3 Modified model of Model2. The meta keyword estimation block in the encoder and the attention block in the decoder were removed.

Model4 Modified model of Model1. Mix-up augmentation for text was removed.

Model5 Modified model of Model1. The audio embedding block consists of one CNN block, the reshape block, one FC layer for changing the hidden dimension to $D = 120$, and one shared Transformer encoder block [16] with time-direction sub-sampling operation. The Transformer encoder block and sub-sampling operation were used twice, with the subsampling operation thinning out the one time-frame every two time-frames.

Model6 Modified model of Model1. The encoder has only one BLSTM layer, and $D = 160$.

The details of four submitted systems are followings:

Submission 1 Ensemble of 20 models. This model consists of two Model1, two Model1single, two Model1param2, two Model2, two Model3, two Model3param2, two Model4, two Model4single, and four Model4param2. The number of trainable parameters was 33.0M.

Submission 2 Ensemble of 50 models. This model consists of five Model1, five Model1single, five Model1param2, five Model2, five Model3, five Model3param2, five Model4, five

Model4single, and ten Model4param2. The number of trainable parameters was 82.5M.

Submission 3 Ensemble of 12 models. This model consists of two Model1, two Model3, four Model4, two Model5single, and two Model6single. The number of trainable parameters was 20.7M.

Submission 4 Ensemble of 30 models. This model consists of five Model1, five Model3, ten Model4, five Model5single, and five Model6single. The number of trainable parameters was 51.7M.

where single means we did not used the HPSS (i.e. $\boldsymbol{X} = \boldsymbol{S}$), and param2 means two additional modification: (i) before adding the meta keyword estimation loss, we multiplied 0.8 to it as a loss weight. (ii) we did not used the second trick in minibatch sample selection, i.e. the target caption was selected with equal probability from the five ground-truth caption of an audio.

3. EVALUATION ON DEV-TEST DATASET

To give a sense of the accuracy of the submitted system, we tested a simplified **Submission 1** on the development-test dataset of the Challenge. First, we conducted three unit tests for Model1, Model2, Model3, and Model4, and then evaluated the ensemble model as Ensemble. Although Ensemble is simpler than actual our challenge submissions, it should be useful for testing the performance of each model and the effectiveness of the ensemble.

Table 1 shows the evaluation results. All models significantly outperformed the baseline system, and with these ensembles model achieved the SPIDEr score 20.7. Our model consists of a complex combination of various sub-blocks and cost functions. As a future work, we will conduct ablation studies to determine how each blocks/cost functions has affected.

4. CONCLUSIONS

This technical report described the system participating to the DCASE 2020 Challenge Task 6 [1]. Our submission focused on solving the indeterminacy problems in word selection and sentence length. We simultaneously solved the main caption generation and sub indeterminacy problems by estimating keywords and sentence length through multi-task learning. The SPIDEr score of our submission on the development-testing dataset was 20.7. Since our model consisted of a complex combination of various sub-blocks and cost functions, as a future work, we will conduct ablation studies for these modules.

5. REFERENCES

[1] http://dcase.community/challenge2020/task-automatic-audio-captioning

[2] K. Drossos, S. Adavanne, and T. Virtanen, "Automated Audio Captioning with Recurrent Neural Networks," in *Proc. of IEEE Workshop on Application of Signal Process. to Audio and Acoust. (WASPAA)*, 2017.

[3] S. Ikawa and K. Kashino, "Neural Audio Captioning based on Conditional Sequence-to-Sequence Model," in *Proc. of the Detection and Classification of Acoust. Scenes and Events Workshop (DCASE)*, 2019.

[4] M. Wu, H. Dinkel, and K. Yu, "Audio Caption: Listen and Tell," in *Proc. of Int'l Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, 2019.

[5] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating Captions for Audios in The Wild," in *Proc. of the North American Chapter of the Association for Computational Linguistics: Human Lang. Tech. (NAACL-HLT)*, 2019.

[6] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An Audio Captioning Dataset," in *Proc. of Int'l Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, 2020.

[7] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic Event Detection in Real Life Recordings," in *Proc. of Euro. Signal Process. Conf. (EUSIPCO)*, 2010.

[8] K. Imoto, N. Tonami, Y. Koizumi, M. Yasuda, R. Yamanishi, and Y. Yamashita, "Sound Event Detection By Multitask Learning of Sound Events and Scenes with Soft Scene Labels," in *Proc. of Int'l Conf. on Acoust., Speech, and Signal Process. (ICASSP)*, 2020.

[9] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic Scene Classification: Classifying Environments from the Sounds they Produce," *IEEE Signal Processing Magazine*, 2015.

[10] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised Detection of Anomalous Sound based on Deep Learning and the Neyman-Pearson Lemma," *IEEE/ACM Tran. on Audio, Speech, and Lang. Process.*, 2019.

[11] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A Transformer-based Audio Captioning Model with Keyword Estimation," submitted to *Interspeech*, 2020.

[12] https://librosa.github.io/librosa/generated/librosa.decompose.hpss.html

[13] S. Bird, E. Loper and E. Klein, "Natural Language Processing with Python," *O'Reilly Media Inc.*, 2009.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Proc. of Advances in Neural Information Process. Systems (NIPS)*, 2014.

[15] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised Data Augmentation for Consistency Training," *arXiv preprint, arXiv:1904.12848*, 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proc. of Neural Information Processing Systems (NIPS)*, 2017.

[17] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, 2019.