An Automatic Caption Generation for video clip with reducing frames in order to shorten processing time.

Riiya Kondo Takashi Yukawa

Nagaoka University of Technology, Nagaoka, Japan 940-2188, s141034@stn.nagaokaut.ac.jp

1. Introduction

The conventional method used entire frames which sampled at regular intervals from video clip to generate text descriptions [1]. Almost generating description methods are using deep learning techniques, such as encoder-decoder framework [2] [3]. Therefore, a learning and text generation take a long time if video has a lot of frames. In other words, processing time is increasing with the length of the video clip. There is a method using keyframe of video as conventional researches in order to reduce processing frames to generate explanation texts. But this method does not consider time step variation of video clips. Video is a set of consecutive images, but it is clearly different from simple image because video has information of time step. We assume that it is important for shortening process time that is not only reducing processing frames but also considering time steps in a video clip.

Humans can create explanation texts only using some of frames not include whole of frames. In particular, we can recognize the content using only few frames if movie is short like which is uploaded on Vine and Twitter. For this reason, we propose a new method for reducing process frame with sustaining semantics and compared proposed method and conventional ones on score and processing time with doing evaluation experiment.

2. Approach

Our approach focuses on reducing process time using fewer frames. The system processes consecutive frames only first and last of video clip instead of all sampled frames as the previous studies because LSTM learns difference in each time steps [4]. More specifically, we take only consecutive 5 each of frames from first and last of video clip to create explanation texts. Each frame is converted to 2048-dimensional feature vector through Inception-V3 Network [5], and these vectors are representative as inputs. Encoder-decoder framework is constructed by two LSTM networks and it's training on end to end. Our encoder-decoder framework has bidirectional LSTM encoder [6]. In general, 2 LSTM networks in bidirectional LSTM process same input data, but processing direction is difference. However, in our proposed method, forward and backward encoders process different inputs. This idea is based on a thing that frames are separated temporally. Forward encoder processes only first consecutive frames, and opposite encoder processes last ones. The outputs what two encoders processed are concatenated with hidden axis to compute attention vector [7]. An attention vector expresses which time step should decoder see. Moreover, two encoder's final states are summed up each other and used for decoder's initial states. LSTM decoder processes sentences words by words. In training, decoder uses ground truth as inputs, but in inference, input on time step t is output of time step t-1 because the model doesn't know future information at inference. Figure 1 shows outline of our proposal system using the mentioned method.

The model uses cross entropy as loss function and implements Adam optimizer to optimize model. A learning rate in the optimizer is 0.001. Our model trained on mini batch training with 256 batch size.

We named our proposed system "s2s" and each run file which is created by using our proposed system includes that name, and in order to compare to our system, we also named conventional method "bitr" as baseline.

3. Experimental evaluation

3.1 Datasets and Setup

We used TGIF dataset as training dataset [8]. TGIF dataset contains about 100k gif data and 120k sentences describing visual content of the animated gifs which are collected from Tumblr. In addition to above, our one of run file has done transferred training

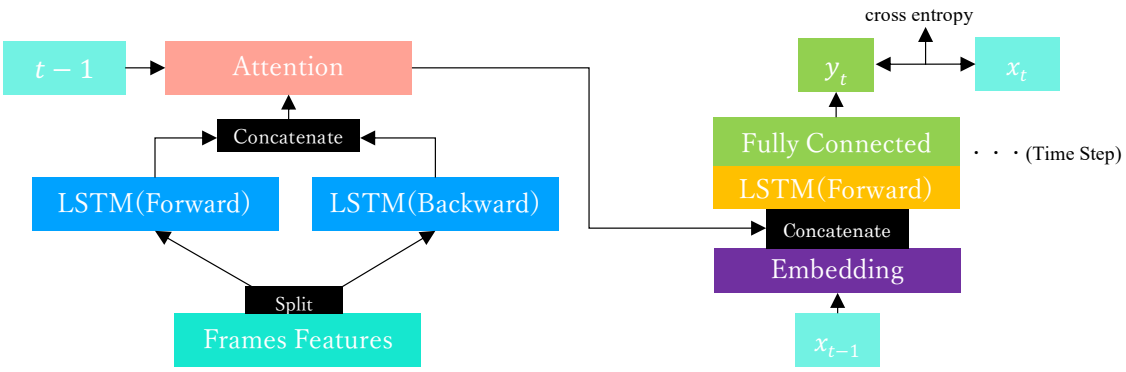

Figure 1 – Proposed system overview

using past (2016 and 2017) VTT datasets which are collected from Twitter and Vine. This run file includes “Transferred” in its file name.

Our other one of run file is created by using beam search algorithm instead of greedy algorithm. The algorithm helps the model to create sentences more correctly. But this algorithm has problem that the computation cost proportionally increases with the beam length N. Moreover, there is no meaning if beam length N is one because it is same as greedy one. Therefore, we determined beam length is 2. This run file includes “BS” in its file name. Table 1 shows dataset and algorithm what did each run file use to create explanation texts.

Table 1 – Run files with dataset and algorithm

Run file name	Dataset	Algorithm
generation.s2s.txt.primary	TGIF only	Greedy
generation.s2sBS.txt	TGIF only	Beam search
generation.s2sTransfered.txt	TGIF & VTT	Greedy
generation.bitr.txt	TGIF only	Greedy

When we use entire frames, we changed the model to a simple bi-directional encoder-decoder model to consider out of memory (OOM).

We also measured processing time how long does each run take training not only scores. And Table 2 shows an environment of computer which carries out our evaluation experiment.

Table 2 – Process environment

Operating System	Ubuntu 18.04.3 LTS
Memory	32 GB
CPU	Intel® Core™ i7-8700 CPU
GPU	NVIDIA GeForce GTX 1080 Ti
Framework	Tensorflow 2.0.0-alpha0 (python3)

3.2 Validation

We created video description texts using past VTT (2018) dataset for validation. The dataset is also collected from Twitter and Vine as well as VTT2016 and VTT2017 dataset. After that we evaluated the scores which is calculated by using METEOR [9] and ROUGE-L [10] metrics. These metrics are known for automatic evaluate guidelines and used on various tasks of natural language processing like machine translation and conversation systems. Table 3 indicates validation scores using VTT2018 data.

Table 3 – Validation scores using VTT2018 data

Run file name	METEOR	ROUGE-L
generation.s2s.txt.primary	0.194	0.2605
generation.s2sBS.txt	0.210	0.2646
generation.s2sTransfered.txt	0.196	0.2488
generation.bitr.txt	0.199	0.2489

Table 3 indicates that beam search algorithm shows highest score. However, as we said, there is tradeoff between processing time and score. A bottom one in Table 3, our baseline, indicates almost same score as our primary run.

3.3 Processing Time

Table 4 shows how long did the models take training per batch in the cases using a part of frames (proposed method) or entire frames (conventional method).

Table 4 – Processing time per batch

	Processing Time(sec/batch)
The proposed method	730.787 sec/batch
The conventional method	4262.087 sec/batch

Table 4 indicates training time of our method is shorter overwhelmingly than the case of using whole of frames. The proposed method succeeded to shorten the time one-fifth than the conventional one with sustaining its scores.

3.4 Evaluation Result using VTT2019 dataset

We tested our models using VTT2019 dataset and calculated scores as well as validation. Table 5 shows the test scores of each run in METEOR and ROUGE-L.

Table 5 – Test scores using VTT2019 data

Run file name	METEOR	ROUGE-L
generation.s2s.txt.primary	0.204	0.2783
generation.s2sBS.txt	0.221	0.2754
generation.s2sTransfered.txt	0.211	0.2821
generation.bitr.txt	0.197	0.2621

A run of beam search one is highest like in validation. Whereas beam search algorithm taken about 1.5 hours to generate 2054 sentences for VTT 2019 videos, but the other ones finished generating within just 25 minutes. A run file which is created by transferred learning shows top score in ROUGE-L metrics.
Finally, Figure 2 shows a scatterplot of relevance between meteor score and number of frames on video clips. According to Fig.2, there was no relationship between run and video length.

4. Discussion

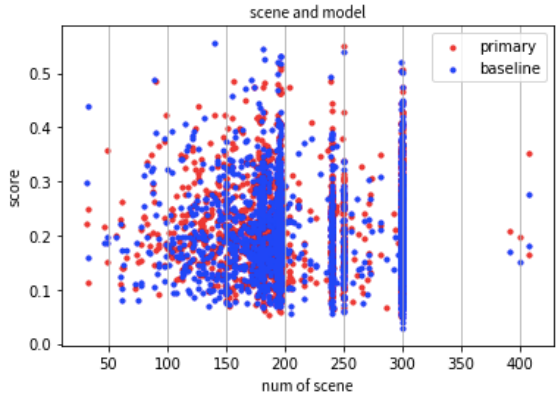

Figure 2 - Relevance in length of video and meteor score

According to Table 5, we found that our proposed method achieved almost the same score as conventional one. In addition, we also found that some efforts like a beam-search algorithm and

transfer learning is effective for our method as well as previous research. But as we said in previous section, we need to consider tradeoff between computation cost and score although these efforts help model to improve certainly. A score difference in primary run and baseline run is slightly different, so we can regard it is within the margin of error. Table 3 indicates a meteor score of the baseline run is higher than primary one at validation. It means there is no difference between both.

We thought our proposed method is not appropriate for long length video because the method doesn't use middle in frames. Whereas the thought, we found from Fig 2 that our model created sentence correctly in spite that video has much of frames. In other words, easy rule base video trimming like our method is useful for generating description in case target video length is from 6 sec to 10 sec, which is like uploaded ones in Twitter, Vine and Flickr. Many of these videos doesn't include scene cut, that is, majority of frames are redundant. In particular, that is clearly when video has consistent frames. In addition to this, an encoder in our proposed model processes different input data bidirectionally. It is assumed that this feature makes the model to learn start and end of video efficiently to generate proper caption. However, understanding video content is depending on video construction rather than video length. Even human cannot understand content or create proper description from limited frames if videos were complex which is constructed by many scenes cut, only abstract scenes and so on. So, we need to categorize videos before train the model. More specifically, we should tell the model what type is video. A "type" doesn't mean video's genre, it means that how video construct is. Hopefully, it is the best way that model learns which frames are useful and selects frames automatically to generate caption. But if its training has high computation cost, it is misplaced our priorities. Therefore, we have to find a new way how to obtain appropriate frames for video captioning with low computation cost. Our preprocess method, just take frame from first and last (except converting to vector), is too simple for generating video description. We need to change a part which we use depending on video type.

In the future, our system will be improved by ensuring of video preprocessing method as such as scene cut detection using color space. Description generation from part of frames is enable but it is importance which part of frames use. Our proposed system used only consecutive 5 frames at each of first and last in a video, however depending on what a type of video is, the system might need other part like "only first", "first and middle" and so on.

5. Conclusion

We found it is effective that use some of first and last frames in video clip in order to reduce processing frames. In particular, it's important each part should be input separately to the model when using frames are separated temporally. To get further improvements scores, we need to investigate what feature is an effective for a model, and how many frames are appropriate to use. In addition, as we already described, there was no big relevance between video length and score according to Fig.2. For this reason, we also should investigate what type of video has high or low score in our system. In summary, this study needs further improvement in the categorization video type. Therefore, we will clarify those points and we also validate our dataset, data preprocessing method and our model to improve the system.

6. References

[1] G. Awad, A. Butt, K. Curtis, Y. Lee, J. Fiscus, A. Godil, A. Delgado, A. F. Smeaton, Y. Graham, W. Kraaij and G. Quénot, "TRECVID 2019: An evaluation campaign to benchmark Video Activity Detection, Video Captioning and Matching, and Video Search & retrieval," in *Proceedings of TRECVID 2019*, 2019.

[2] A.-A. Liu, Y. Qiu, Y. Wong, N. Xu, Y. Su and M. S. Kankanhalli, "Tianjin University and National University of Singapore at TRECVID 2017: Video to Text Description," 2017.

[3] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014, pp. 3104-3112.

[4] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," vol. 12, 2000, pp. 2451-2471.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818-2826.

[6] A. Graves, N. Jaitly and A.-r. Mohamed, "Hybrid speech recognition with Deep Bidirectional LSTM," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.

[7] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv:1409.0473 [cs.CL], 2014.

[8] Y. Li, Y. Song, L. Cao, J. Tetreault, L. Goldberg, A. Jaimes and J. Luo, "TGIF: A New Dataset and Benchmark on Animated GIF Description," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[9] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65-72.

[10] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Proceedings of the Workshop on Text Summarization Branches Out*, 2004, pp. 74-81.