Fusion of Multimodal Embeddings for Ad-Hoc Video Search

Danny Francis[†], Phuong Anh Nguyen[‡], Benoit Huet[†], Chong-Wah Ngo[‡]
[†]EURECOM, Sophia-Antipolis, France [‡]City University of Hong-Kong, Kowloon, Hong Kong
francis@eurecom.fr, panguyen2-c@my.cityu.edu.hk, huet@eurecom.fr, cscwngo@cityu.edu.hk

Abstract

The challenge of Ad-Hoc Video Search (AVS) originates from free-form (i.e., no pre-defined vocabulary) and free-style (i.e., natural language) query description. Bridging the semantic gap between AVS queries and videos becomes highly difficult as evidenced from the low retrieval accuracy of AVS benchmarking in TRECVID.

In this paper, we study a new method to fuse multimodal embeddings which have been derived based on completely disjoint datasets. This method is tested on two datasets for two distinct tasks: on MSR-VTT for unique video retrieval and on V3C1 for multiple videos retrieval.

1. Introduction

Ad-Hoc Video Search (AVS) is a challenging task consisting in using natural language queries to retrieve relevant video segments from large datasets (see Figure 1). Ad-Hoc implies that the query follows no pattern, and the terms are drawn from an open vocabulary that is not restricted to any particular domain. Due to this nature, a query can be very specific (e.g., "find shots of a surfer standing on a surfboard, not in the water") or open ended (e.g., "find shots inside a moving car"). Therefore, a "good" model for AVS should be able to extract relevant high-level features from videos, parse text queries, and find a common representation of both modalities for relevance judgement. TRECVID, an evaluation campaign, is organized yearly by the NIST to evaluate state-of-the-art models for different video processing tasks, including the AVS task [3]. In this paper, the focus of study is automatic AVS using external training data.

Most recent works in image and video processing and in text processing rely on Deep Learning techniques. For image processing, commonly used feature extractors or concepts detectors are deep Convolutional Neural Networks (CNNs) which have been pretrained on ImageNet 1000 categories [8]. For video processing, I3D trained on the Kinetics dataset has shown excellent results in activity detection [5]. Regarding text processing, recent works have shown that RNNs such as LSTMs [11] or GRUs [7] could be used to build efficient language models.

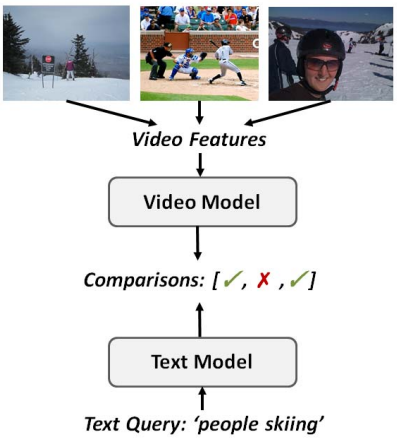

Figure 1. Principle of AVS. Video Features are derived from videos and processed by a Video Model to obtain a vector representation. At the same time, a text query is processed by a Text Model that also derives a vector representation. These two vector representations are then compared to list all relevant videos with respect to the text query.

In this paper, we propose using a fusion of three multimodal modules trained on different datasets to tackle the AVS task. Our contributions are two-fold:

- joint exploitation of object counting, activity detection and semantic concept annotation for query interpretation;
- a new fusion method that combines three modules trained on different datasets and shows competitive performance.

The remaining sections are organized as follows. Section 3 introduces the related works in AVS. Section 4 introduces the cross-modal learning employed for training three different modules, while Section 4 describes the proposed fusion method. Section 5 provides empirical insights and Section 6 concludes this paper.

2. Related Works

From AVS 2018 [2], the general approaches from the participants can be summarized as follows: linguistic analysis for query understanding combining different techniques for concept selection and fusion; or learning joint embedding space of textual queries and images; or the integration of two mentioned approaches. From the results of ten participants, we conclude that the approach of learning the embedding space is the key of success for AVS task. Following up this direction, we propose to learn three embedding spaces including objects counting, activities and semantic concepts separately, and a fusion method to incorporate these models.

3. Cross-Modal Learning

In this section we will describe the multimodal models we employed. More precisely we will first define their architecture and then how we trained them. Please note that in this section, we will consider images and videos, even though our models will be used for Ad-Hoc Video Search. The reason is that some of our models will be trained on images and applied at frame-level on videos. More informations will be given at Section 5.2.

3.1. Feature Representation

Let Q be a textual query and V an image or a video. We want to build a model so that Q and V can be compared. More precisely, we want to be able to assign a score to any (Q, V) to describe the relevance of V with respect to Q. For that purpose, we use a similar model to [9].

For processing textual queries, we represent any query Q of length L as a sequence $(w_1, ..., w_L)$ of one-hot vectors of dimension N, where N is the size of our vocabulary. These one-hot vectors are then embedded in a vector space of dimension D. More formally, we obtain a sequence of word embeddings $(x_1, ..., x_L)$ where $x_k = w_k W_e$ for each k in $\{1, ..., L\}$. The weights of the embedding matrix $W_e \in \mathbb{R}^{D \times N}$ are trainable.

The obtained sequence of word embeddings is then processed by a GRU, whose last hidden state is kept and input to a Fully-Connected layer to get a sentence embedding. Formally, a GRU is defined by the following equations:

$$u_t = \sigma(h_t W_{uh} + x_{t+1} W_{ux} + b_u) \quad (1)$$

$$r_t = \sigma(h_t W_{rh} + x_{t+1} W_{rx} + b_r) \quad (2)$$

$$\bar{h}_t = \tanh((h_t \circ r_t) W_{hh} + x_{t+1} W_{hx} + b_u) \quad (3)$$

$$h_{t+1} = (1 - u_t) \circ h_t + u_t \circ \bar{h}_t \quad (4)$$

where $W_{uh}, W_{ux}, W_{rh}, W_{rx}, W_{hh}, W_{hx}, b_u, b_r$ and b_u are trainable parameters. If the length of the input sequence is L, then the final sequence embedding v_s is defined as:

$$v_s = h_L W_s + b_s \quad (5)$$

where W_s and b_s are trainable parameters.

Regarding visual objects, the generic process we employ is to extract a vector representation $\varphi(V)$ of a visual object V where φ corresponds to any relevant concepts or features extractor. Then, we input $\varphi(V)$ to a Fully-Connected layer to obtain a visual embedding v_v:

$$v_v = \varphi(V) W_v + b_v \quad (6)$$

where W_v and b_v are trainable parameters.

Our goal is to train these models to be able to compare v_s and v_v. We will explain how these models are trained in Section 3.2.

3.2. Model Training

The objective is to learn a mapping such that the relevancy of a pair of a query and a video (Q, V) can be evaluated. As explained in Section 3.1, our model derives a query representation v_s from Q and a video representation v_v from V. Triplet loss is used as the loss function for model training. Mathematically, if we consider a query representation v_s, a positive video representation v_v (corresponding to v_s) and a negative video representation $\bar{v}_v$ (that does not correspond to v_s), the triplet loss $\mathcal{L}$ for $(v_s, v_v, \bar{v}_v)$ to minimize is defined as follows:

$$\mathcal{L}(v_s, v_v, \bar{v}_v) = \max(0, \alpha - \cos(v_s, v_v) + \cos(v_s, \bar{v}_v)) \quad (7)$$

where α is a margin hyperparameter. We chose to employ the hard-margin loss presented in [9], where $\bar{v}_v$ is chosen to be the representation of the negative video with the highest similarity with the query representation v_s among all videos in the current training mini-batch.

4. Fusion Strategy

In this section we will describe the three multimodal modules we used and how we fused them.

4.1. Multimodal Modules

Our model relies on three multimodal modules: a counting module, an activity module and a concepts module (see Figure 2). Each of them has the architecture we described in Section 3.1 and has been trained according to the optimization scheme we defined in Section 3.2.

The counting module is based on a Faster-RCNN [16] trained on the OpenImagesv4 dataset [13]. It takes images as inputs. For each input, it detects objects belonging to the 600 classes of OpenImagesv4 and counts them to obtain a vector of dimension 600, where the value at index i corresponds to the number of detected objects of class i. Embeddings are then derived from that vector.

The activity module relies on an I3D trained on Kinetics-600 and takes video inputs. Each input is processed by the

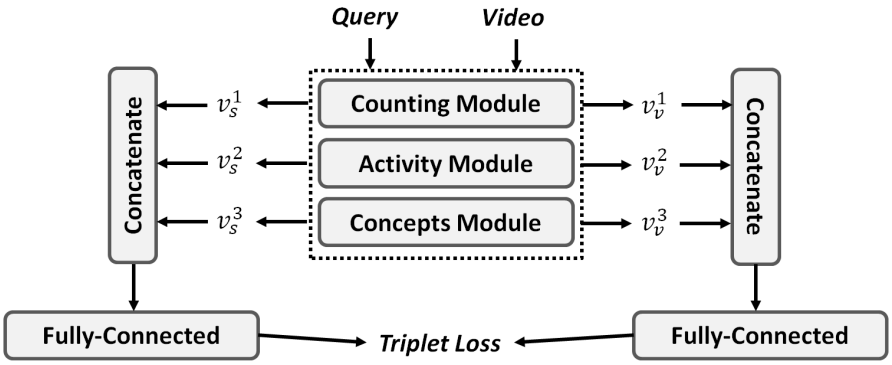

Figure 2. Proposed model. We extract embeddings from three modules: a counting module, an activity module and a concepts module. These embeddings are then concatenated and input to Fully-Connected layers to obtain new embeddings. That model is also trained using a triplet loss.

I3D, which returns a vector of 600 logits corresponding to the 600 activities of the Kinetics-600 dataset. That vector is then processed as described in Section 3.1 to obtain an embedding.

The concepts module takes as input concepts detections coming from four different concept detectors. These concept detectors are ResNet [10] models trained on ImageNet1k, Places-365 [22], TRECVID SIN [21] and HAVIC [18]. Following the same process as for other two modules, we generate embeddings from the concatenation of the concept detections coming from these four detectors.

4.2. Fusion Model

Instead of simply averaging similarity scores to compare videos and queries, we chose to train a model to draw finer similarities between them. For that purpose, we derived embeddings from our modules for videos and queries, and passed them through Fully-Connected layers to obtain new embeddings. More formally, if v_v^1, v_v^2 and v_v^3 are video embeddings respectively generated by the counting module, the activity module and the concepts module, we derived the new video embedding v_v as follows:

$$v_v = \text{concat}(v_v^1, v_v^2, v_v^3) W_v^{\text{fuse}} + b_v^{\text{fuse}} \quad (8)$$

where W_v^{fuse} and b_v^{fuse} are trainable parameters. Similarly, if v_s^1, v_s^2 and v_s^3 are query embeddings, we obtain a new query embedding as follows:

$$v_s = \text{concat}(v_s^1, v_s^2, v_s^3) W_s^{\text{fuse}} + b_s^{\text{fuse}} \quad (9)$$

where W_s^{fuse} and b_s^{fuse} are trainable parameters.

We trained our fusion models using the same triplet loss as we did for multimodal modules, as decribed in Section 3.2.

5. Experiments

In this section, we describe how we implemented and trained our models, and present our experimental results.

5.1. Datasets

We used the MSCOCO [15] dataset to train the counting module (not the Faster-RCNN itself) and the concepts module. MSCOCO is composed of about 120k images, and five captions per image. We trained modules on the whole dataset, using 1k images for validation: we did not employ the usual train/validation/test split.

Regarding the activity module, it has been trained on the TGIF [14] dataset (containing about 100k animated GIF images and 125k captions) and on the MSVD [6] dataset (containing 1970 videos and about 70k captions).

Fusion models have been trained on the MSR-VTT [20] dataset, containing 10k videos with 20 captions each. We used the usual split: 6513 videos for training, 497 for validation and 2990 for testing.

Our models have also been evaluated in terms of mean average precision based on 10,000 retrieved shots on V3C1 [4], containing 7475 videos split into 1,082,657 shots, using the provided ground-truth results for six queries.

5.2. Implementation details

We implemented our models using the Tensorflow [1] framework for Python. Each of them has been trained for 150k iterations with mini-batches of size 64. We used the RMSProp [19] algorithm, with gradients capped to values between -5 and 5 and a learning rate of 10^{-4}. Hidden dimensions of GRUs are always 1024, and embeddings output by multimodal modules and fusion models are of dimension 512. The size of vocabularies has been set to 20k. We applied dropout [17] with rate 0.3 to all outputs of Fully-Connected layers, and batch normalization [12] to the inputs of our models. In triplet losses, the α parameter has been set to 0.2.

Modules trained on images (counting and concepts modules) are used for videos during testing in two different ways. For tests on MSR-VTT, we extracted uniformly

one frame every fifteen frames, applied the extractor on each frame (Faster-RCNN for the counting module or concepts extractors for the concepts module) and averaged obtained vectors. For tests on V3C1, we processed provided keyframes instead of entire videos.

5.3. Performance of Modules

Model	R@1	R@5	R@10	medR
M_1 (Counting)	2.95%	7.16%	11.40%	264
M_2 (Activity)	2.83%	8.80%	13.59%	**167**
M_3 (Concepts)	**3.88%**	**10.69%**	**15.44%**	168

Table 1. Results on the MSR-VTT test dataset of three modules.

Model	mAP
M_1 (Counting)	2.15%
M_2 (Activity)	0.00%
M_3 (Concepts)	2.15%

Table 2. Results on the V3C1 dataset of three modules.

In this section, we report results of each module on the unique video retrieval task (evaluated on MSR-VTT) and the multiple videos retrieval task (evaluated on V3C1). Results are reported in Table 1 and Table 2.

One can notice that relative results of modules are completely different with respect to the task. On the unique video retrieval task, the counting module has the worst results, and the concepts module has the best results. On the multiple videos retrieval task, the counting module and the concepts module perform similarly, and the activity module has very bad results.

We think that these results are due to the fact that shots in the V3C1 datasets are much shorter than the videos on which the I3D activity extractor has been trained. For that reason, we will not report results of fusions involving the I3D on V3C1 in the following. Regarding the fact that the counting module performs as well as the concepts module on the multiple videos retrieval task, our hypothesis is that the multiple videos retrieval task requires less precision than the unique video retrieval task: the concepts module covers a large range of visual concepts, which is useful when looking for a specific video, but less useful when the goal is to retrieve as many videos as possible.

In the next section, we will present results of fusions

5.4. Performance of Fusions

Results of fusion models are reported in Table 3 and Table 4. Two types of fusions have been tested : $M_i + M_j$ means that we summed up similarity scores between modules M_i and M_j, and F(M_i, M_j) means that we applied the fusion scheme we described in Section 4.

In each case, the best model involves a fusion according to our fusion method. In the unique video retrieval task, the fusion alone performs better than other models whereas in the multiple videos retrieval task, the sum of similarity scores of modules and of their fusion has the best results. The reason may be that our fusion scheme makes finer representations of videos, which is less useful for multiple videos retrieval than for unique video retrieval.

Model	R@1	R@5	R@10	medR
$M_1 + M_2$	3.91%	11.31%	16.87%	133
$M_1 + M_3$	4.29%	11.56%	16.22%	149
$M_2 + M_3$	4.69%	13.31%	19.19%	105
$M_1 + M_2 + M_3$	**5.00%**	**13.70%**	**19.37%**	**104**
F(M_1, M_2)	5.20%	15.78%	23.69%	59
F(M_1, M_3)	4.80%	14.70%	22.09%	70
F(M_2, M_3)	5.90%	18.00%	26.39%	49
F(M_1, M_2, M_3)	**6.48%**	**19.27%**	**27.99%**	**42**
Sum of best	6.72%	17.80%	24.72%	67

Table 3. Results on the MSR-VTT test dataset of fusions of modules.

Model	mAP
$M_1 + M_3$	4.54%
F(M_1, M_3)	4.01%
F$(M_1, M_3) + M_1 + M_3$	**5.41%**

Table 4. Results on the V3C1 dataset of fusions of modules.

6. Conclusion

In this paper, we proposed to tackle the AVS problem using three modules: a counting module, an activity module and a concepts module. Each of these modules analyzes videos and derives embeddings in a multimodal space. We showed that jointly taking advantage of counting objects, detecting activities and detecting semantic concepts in videos allowed to deal efficiently with the complexity of the AVS task. Moreover, we proposed a method to fuse modules trained on different datasets that appeared to lead to significantly better results than simpler fusion methods.

Acknowledgments

This work was supported by ANR (the French National Research Agency) via the ANTRACT project, the European H2020 research and innovation programme via the project MeMAD (Reference Np.: GA780069), a grant from the Research Grants Council of the Hong Kong SAR, China (Reference No.: CityU 11250716), and a grant from the PROCORE-France/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the Consulate General of France in Hong Kong (Reference No.: F-CityU104/17).

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] G. Awad, A. A Gov, B. , K. Curtis, Y. Lee, y. Gov, J. Fiscus, D. Joy, A. Delgado, A. Smeaton, Y. Graham, W. Kraaij, G. Qunot, J. Magalhes, and S. Blasi. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. 04 2019.

[3] G. Awad, A. Butt, J. Fiscus, D. Joy, A. Delgado, W. Mcclinton, M. Michel, A. Smeaton, Y. Graham, W. Kraaij, et al. Trecvid 2017: Evaluating ad-hoc and instance video search, events detection, video captioning, and hyperlinking. 2017.

[4] F. Berns, L. Rossetto, K. Schoeffmann, C. Beecks, and G. Awad. V3c1 dataset: An evaluation of content characteristics. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, ICMR '19, pages 334–338, New York, NY, USA, 2019. ACM.

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, 2017.

[6] D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics, 2011.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. Vse++: Improving visual-semantic embeddings with hard negatives.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[13] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018.

[14] 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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4641–4650. IEEE, 2016.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[18] S. M. Strassel, A. Morris, J. G. Fiscus, C. Caruso, H. Lee, P. Over, J. Fiumara, B. Shaw, B. Antonishek, and M. Michel. Creating havic: Heterogeneous audio visual internet collection. Citeseer.

[19] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[20] J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

[21] W. Zhang, H. Zhang, T. Yao, Y. Lu, J. Chen, and C. Ngo. Vireo@ trecvid 2014: instance search and semantic indexing. In *NIST TRECVID Workshop*, 2014.

[22] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.