DeepSeek: A video captioning tool for making videos searchable

Kratarth Goel & Juhi Naik
Department of Computer Science
Stanford University
{kratarth, juhinaik}@cs.stanford.edu

Abstract

Inspired by recent advances in the field of deep learning and success that it has gained on various problems like image captioning [8] and [15], machine translation [5], word2vec, skipthoughts [9], etc, we create a end-to-end system model for making video content searchable, such that the search is not limited to meta data but the actual content of the video. We transcribe video clips into text based on its contents and then construct a pipeline that can efficiently and accurately search for this content by an extension of the state-of-the-art skipthough vectors. This makes the task of sifting through hours and hours of video to find what one is looking for much faster. We present Deep-Seek: a deep learning model to get search through video content and show that it performs well on the task defined.

1 Introduction

YouTube has 400 hours of video uploaded every min. One of the most daunting tasks that users face on such sites is to find the interesting/relevant videos from the search results without opening and going through each one. While video summarization seems like a nice solution to the problem, a better one would be to make videos searchable for their content via Video Captioning, so that one might search for the part of the video that they are interested in rather than spending time watching on all of it.

When we summarize a video by reducing the duration of the video, we loose a lot of information. This maybe specially bad for content where sound is as informative as the video frames. This is because during video summarization, we skip frames making intermittent breaks in the audio playback. However, if we make videos searchable by content, we might just be able to skip through all the unnecessary details and watch the part that is important to us in all of its detail. In this project we present a tool to generate a text summary of parts of videos that can then be searched for.

2 Background/Related Work

Most prior work on natural-language description of visual data has focused on static images [8, 15, 14]. The existing work on describing videos with sentences [12, 10] deals with constrained domains, the first using context-free grammar to generate sentences and the second generating subject-verb-object triples. T.S. Motwani in [11] explores description of activities in videos by first classifying it into a verb class and then identifying the subjects within the frames. Our approach, instead, attempts to understand the entire context of the video clip and generate a description of it's content. It also differs from all previous work since we aim to make videos retrievable based on it's content instead of just generating captions of the video as well as seek the exact point of the video where the scene described by the query occurs. To the best of our knowledge, this has never been explored before.

3 Technical approach and models : DeepSeek

We divided the problem into two tasks:- (1) The task of captioning the videos and (2) the task of content based video retrieval. We describe both of this in this section.

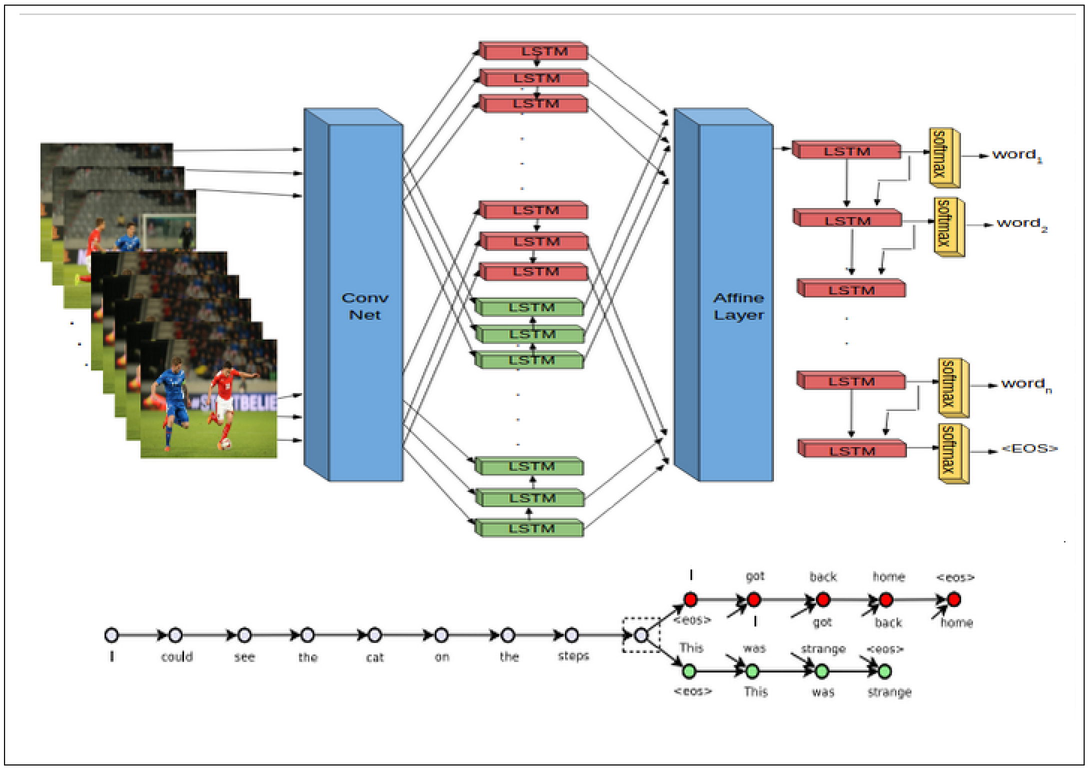

Figure 1: A sequence of image constituting the video is fed into a convnet architecture. The temporal sequence of penultimate layer feature of these images, are fed into a bidirectional LSTM. The hidden state of these are then projected into a feature space using an affine (ReLU) layer. This feature vector is then fed into the language model, to generate the caption for the video. (B) In the second part of our pipeline, we feed the generated caption into a skip thoughts model which works as a sent2vec architecture. Given a tuple $(s_{i1}, s_i\ , s_{i+1})$ of contiguous sentences, with s_i the $i-th$ sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i1} and next sentence s_{i+1}.

The Deep-Seek model consists of the following components:-

1. A state-of-the-art convolutional neural network is used to extract semantic information from the video frames to construct a temporal sequence of features that can be collected and fed into a language model. We feed into this network a video, frame by frame and thus get 10 features (penultimate layer) of a trained conv-net model corresponding to the 10 frames per video in our current dataset. For the purposes of these experiments we used a pre trained VGG Net aand ResNet. These temporal sequence of features is then fed into a bidirectional LSTM.

$$\hat{x_j^i} = f(x_j^i)$$

Where $j \in \{1, \ldots N\}$ is one video in the dataset of N videos and i is one of the 10 frames of video j and f is a convnet(VGG here).

2. The output of the conv net is fed into the bi directional LSTM, that takes the temporal sequence of frame features extracted and tried to model the time domain dependency in the

video frame features.

$$\overrightarrow{l_j^t} = g(\hat{x}_j^t)$$
$$\overleftarrow{l_j^t} = g'(\hat{x}_j^{T-t})$$

where g and g' are the LSTM function. Also $\overset{\rightharpoonup}{l_j^t}$ defines the forward context of the frame t (where $t \in \{1, \dots 10\}$) and similarly, $\overrightarrow{l_j^t}$ defines the forward context of the frame t (where $t \in \{1, \dots 10\}$) and $T = 10$.

3. The output of the bi directional LSTM is fed into a Affine layer that converts this into a single feature vector that serves as an entry point to the Language Model.

$$\alpha_j^t = h(\overrightarrow{\hat{l}^t}_j, \overleftarrow{\hat{l}^t}_j)$$

where h is a function of the following form

$$h(x_1, x_2) = ReLU(W_1 x_1 + W_2 x_2 + b)$$

4. The Language model is a LSTM, that takes as input the concatenation of all the α_j^t for all $t \in \{1, \dots, T\}$. It then generates a softmax output at each time step predicting a word, untill a end of sequence symbol is not predicted.
5. The generated caption is then fed into a pre-trained skip thoughts model. Given a tuple (s_{i-1}, s_i , s_{i+1}) of contiguous sentences, with s_i the $i - th$ sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1}. We just run the encoder part of this model, to convert our generated captions into vectors in a semantic space. These vectors are then used for retrieval.
6. The retrieval pipeline is itself very simple. The vectors are returned by increasing order of magnitude of euclidean distance. This can be illustrated as below:-

$$k = \arg\min_i (v_q - v_i)^2$$

here, v_q is the vector corresponding to the query, v_i is the vector corresponding to the i-th clip in the dataset and k is the top retrieved clip from the dataset. In general we can retrieve top-n videos, based on increasing order of the euclidean distance calculated from the equation above.

4 Experiments

4.1 Data

The dataset we are using, which was collected in VideoSET [16], consists of ground-truth summaries for two publicly available egocentric video datasets, and four TV episodes. There are 11 main classes of videos,

1. 4 egocentric ones of 3-5 hours each recording people doing daily activities such as eating, shopping, and cooking
2. 3 egocentric ones of a person during a day at Disneyworld Park
3. 4 TV episodes of 45 minutes each. The episodes consist of 1 from Castle, 1 from The Mentalist, and 2 from Numb3rs.

For the purposes of our research we create 5 second clips of each of these videos with their corresponding annotations. The dataset in total consists of 2,840 video clips of 5 seconds each. The frame rate of each of these videos is around 30fps. Thus each video will consist of 150 (= 30*5) frames per video. Instead of handling all of these images, we subsample the frames to have only 10 equally spaced frames per second. Thus our data gets downsampled to contain 284,000 frames(images). We divide the dataset to have 40000 frames i.e. 4000 videos in the test set and the same number in the validation set leaving the rest for the train set.

4.2 Experiments

The DeepSeek project has two stages - generating captions for video clips and making the video searchable using these captions. For the first part of the pipeline, we tested on different combinations of state-of-the-art convolutional networks for feature extraction, with different temporal layers used to merge the features of consecutive frames.

1. VGG16 - Affine - LM
   The conv-net used for feature extraction was a pre-trained VGG-16 model [8] which was slightly fine-tuned while training. The features extracted for the whole batch were combined using an Affine (ReLU) layer and fed into the Language model.
2. VGG16 - LSTM - LM
   This model was similar to the previous one except the Affine layer was replaced with a single forward LSTM layer, to capture the forward temporal dependency on previous frames.
3. VGG16 - BiLSTM - LM In this, model, the forward LSTM was replaced with a bi-directional LSTM to capture symmetric dependency on previous as well as following frames.
4. ResNet200 - Affine - LM
   To potentially improve the feature extraction from the frames, the VGG16 model was replaced with a pre-trained ResNet network with 200 layers [2]. The rest of the model was the same as described in 1.
5. ResNet200 - LSTM - LM
   Same as described in 2, except that VGG16 is replaced by the ResNet200 architecture.
6. ResNet200 - BiLSTM - LM
   Same as described in 3, except that VGG16 is replaced by the ResNet200 architecture.

The second stage of the pipeline used initialized using pre-trained skipthoughts model [4]. This was used to convert the generated captions into vectors. The motivation behind using skipthoughts was, that the skipthoughts vector in their original paper have been shown to be capable enoough to capture the ”meaning” of the sentences they encode. When used, without any finetuning, on tasks like semantic analysis, paraphrase detection, etc, they work very well. So we believe that the skipthought vectors would be able to cluster captions that are similar to each other in their meaning. This fact can also be verfified by the cluster that we observe by visualizing the vectors as a tsne embedding in Figure 2.

The user query, too, was converted into a vector using the same model. The similarity measurement used to rank the results for the query was Euclidean distance.

4.3 Evaluation metrics

We evaluted the quality of the generated captions by using two separate metrics, the cross entropy loss from the softmax layer of the language model and the BLEU score for the generated captions.

The model is also detailed in the figure 1. The entire model is trained using back propagation on the cross entropy on the softmax output of the language model. We measure performance in the terms of this cross entropy defined as

$$\mathbf{L}_j = -\sum_k y_i \log P(\hat{y_i})$$

where symbols have their usual meaning. L_j is defined as the cross entropy loss for the j-th word in the caption generated. The cross entropy loss is defined over the length of the caption as the average of the cross entropy loss for each generated word. Thus

$$\mathbf{CE}(S) = \frac{1}{N}\sum_{j=1}^{N} \mathbf{L}_j$$

where $S = \{w_1, \dots w_N\}$ is the sequence of words in the caption.

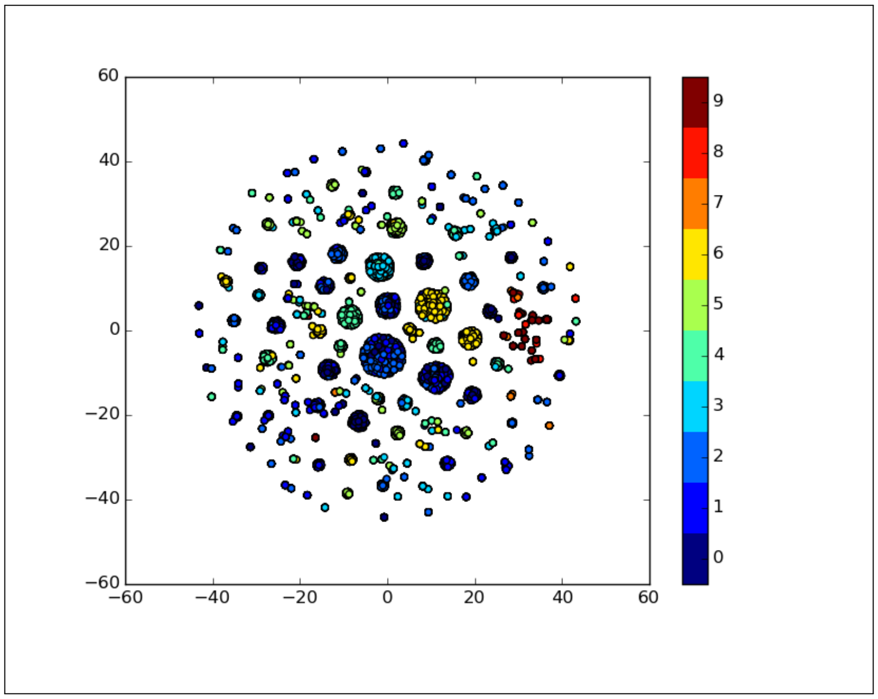

Figure 2: The t-SNE [13] embedding of the vectors generated by the skipthough model on the captions generated for the dataset. The different colors correspond to the classes of videos described in the Data section

Model	Cross-Entropy	BLEU-1	BLEU-2	BLEU-3	BLEU-4
VGG16 - Affine - LM	1.5520	60.20	41.77	24.63	16.57
ResNet200 - Affine - LM	1.3655	61.79	42.71	26.19	17.57
VGG16 - LSTM - LM	0.9822	68.79	46.26	32.81	26.7
ResNet200 - LSTM - LM	0.8932	69.65	47.60	35.56	28.70
VGG16 - BiLSTM -LM	0.7222	69.39	48.61	36.43	30.04
ResNet200 - BiLSTM - LM	**0.5932**	**71.82**	**50.40**	**38.70**	**31.77**

Table 1. The results in terms of of both the cross entropy loss and the BLEU scores evaluated on the test set for the various models trained as part of the project.

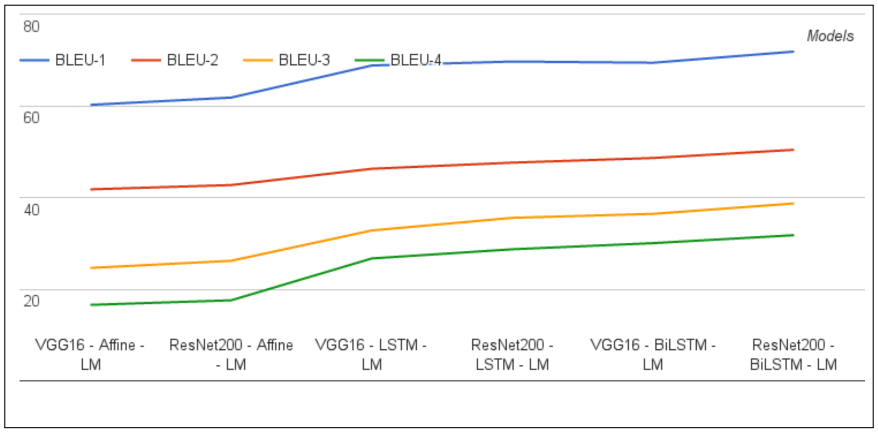

Figure 3: Comparison of BLEU-1, BLEU-2, BLEU-3 and BLEU-4 scores on test data for all the experiments

The table above shows the results of all the experiments done. Notably, changing the convnet from VGG16 to ResNet200 had a relatively small effect, as compared to changing the Affine projection layer to LSTM anf BiLSTM. We suspect the reason for that is, the convolutional networks are both very good, but the Affine model lacks the temporal dependencies that are part of the video whose caption is being generated. The BiLISTM does the best job in capturing such dependencies, hence outperforming both the affine layer models and the LSTM models. The results can also be visualized as seen in Figure 3.

To evaluate the content based video retrieval task, which would test our complete pipeline ,i.e. the caption generation and the video retrieval tasks quantitatively, we use three standard evaluation measures widely used in CBIR tasks [14], including the mean average precision (mAP), the precision at particular ranks (P@K), and the recall at particular ranks (R@K).

Since we do not have a dataset ranked for retrieval, in order to calculate the ”ideal” ranking list, we do the following. For each clip in the test set, we query on the original captions of the videos in our dataset. The rank in which the videos are retrieved form the ground truth ranking list. Then we repeat the experiment with the generated captions and compare with the ground truth list of captions that we just created, and evaluate the metrics described above.

Model	**mAP**	**P@K**=1	**p@K**=10	**R@K**=1	**R@K**=10
VGG16 - Affine - LM	0.492	0.483	0.410	0.0301	0.0289
ResNet200 - Affine - LM	0.506	0.489	0.433	0.0329	0.0296
VGG16 - LSTM - LM	0.621	0.616	0.521	0.0441	0.0428
ResNet200 - LSTM - LM	0.634	0.621	0.549	0.0468	0.0436
VGG16 - BiLSTM -LM	0.691	0.674	0.583	0.0524	0.0507
ResNet200 - BiLSTM - LM	**0.725**	**0.699**	**0.614**	**0.0561**	**0.0552**

Table 2. Evaluation of entire pipeline in terms of Precision, Recall and Mean Average Precision

Suitable $L2$ regularization was used where ever needed. Adam optimizer was used for back propagating the gradients with $\alpha = 0.8, \beta = 0.999, \epsilon = 1e - 8$. The weights of the convolutional net were fixed for the first 100 iterations of the model. After which the weights of the CNN were fine tuned using a small learning rate equal to 1e-5 where the learning rate of the rest of the model was set to 1e-3. We used a single layered LSTM with 512 units in all its components for the language model. The output of the VGG16 net was reduced to 512 dimensions, while the penultimate layer feature for ResNet200 was 2048 units. Gradient was clipped for all LSTM units at 5. The dropout probability for all the LSTM structures was 0.1 to prevent overfitting. The LSTM or biLSTM to learn the temporal dependencies also had 512 units in all their components along with the output dimension. Similarly, the affine layer used as input to the langauge model had an output dimension of 512 as well.

Most of the code is written as a combination of Theano [6] and Torch [7, 3] and the best perfoming model was trained for over 3 days on a NVidia TITAN X GPU. The initial weights to initialize the convnet model for VGGNet -16 were obtained from BVLC Caffe Zoo [1] and ResNet200 from facebook AI’s github resources [2].

5 Qualitative Results and Performance

To give an example of the qualitative performance of our pipeline, we ran the query ”Will says Becket likes Castle”. As described in the model, this query was converted into a vector representation using the skip-though model and the set of vectors for the generated captions were ranked according to their euclidean distance from the query vector. The top 5 video clips corresponding to the closest vectors, consisted of 4 clips from the Castle episode. The 2nd among this had the original caption ”Will asks Becket if she likes Castle”. The remaining had one or more of the 3 characters in the scene talking to each other.

Similarly good results were observed when queries like ”I and my friends were walking in the park”, ”I cooked in the kitchen” etc. were tried.

One interesting aspect observed from our experience with the qualitative results was, that even without the use of the audio as an input to our model, the model was still able to associate the

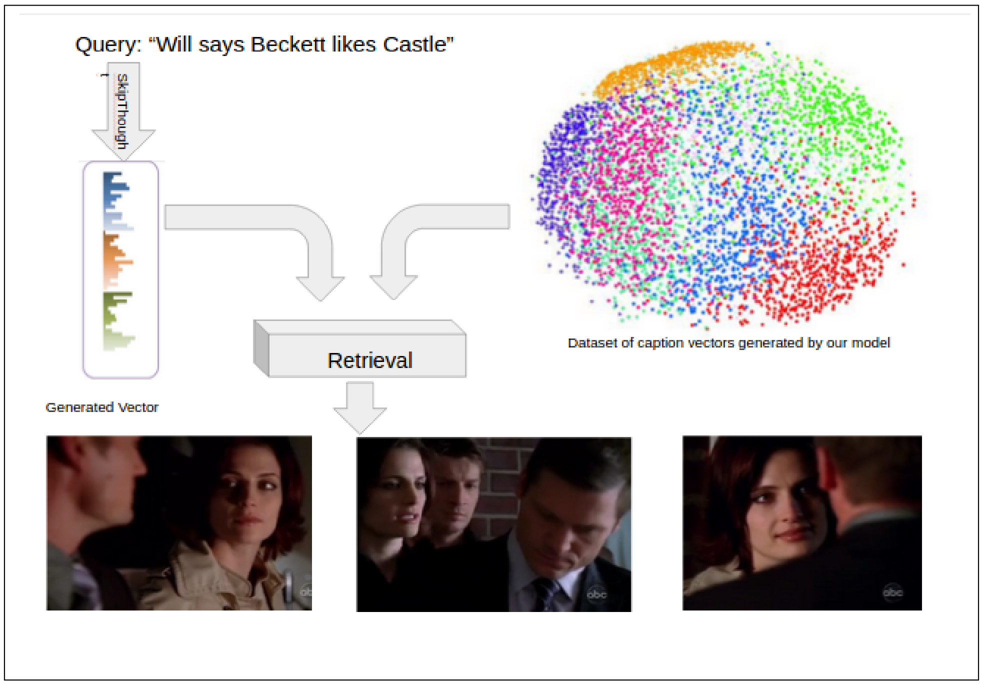
Query: "Will says Beckett likes Castle"
SkipThough
Generated Vector
Retrieval
Dataset of caption vectors generated by our model

Figure 4: An example run for our end to end pipeline. The part of the figure depicting the set of vectors for generated captions is for representational purposes only.

entities with their respect videos. For instance, "Castle" was associated with the face of "Nathan Fillion", the actor who plays the character on the tv series, etc. We assume this happened because our model was powerful enough that after repeatedly showing captions that contained the word "Castle" for video clips that had Nathan in them, the model was able to associate the image frames with the words.

As far as the performance in terms of time is concerned, the VGG-16 architecture takes about 425 msec to forward pass a mini-batch of 10 images (one video clip) on our NVIDIA TITAN X chip and the ResNet200 took 687msec for the same. However, this operation is offline. For the online retrieval operation, we just need to run the skip thoughts part of the model on the caption that is provided for the search query, and then do a euclidean distance based ranking on our dataset (of nearly 28000 videos). This takes about 0.133 seconds on average on the GPU an 1.5 seconds on the CPU. Which is not bad at all. We will try improving these times in the future.

References

[1] https://github.com/bvlc/caffe/.

[2] https://github.com/facebook/fb.resnet.torch.

[3] https://github.com/karpathy/neuraltalk2/.

[4] https://github.com/ryankiros/skip-thoughts.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[6] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[7] R. Collobert, S. Bengio, and J. Marithoz. Torch: A modular machine learning software library, 2002.

[8] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[9] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[10] N. Krishnamoorthy, G. Malkarnenkar, R. Mooney, K. Saenko, and S. Guadarrama. Generating natural-language video descriptions using text-mined knowledge. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 541–547. AAAI Press, 2013.

[11] T. S. Motwani and R. J. Mooney. Improving video activity recognition using object recognition and text mining. In *ECAI*, pages 600–605, 2012.

[12] M. Usman, G. Khan, and Y. Gotoh. Describing video contents in natural language.

[13] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[14] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the ACM International Conference on Multimedia*, pages 157–166. ACM, 2014.

[15] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[16] S. Yeung, A. Fathi, and L. Fei-Fei. Videoset: Video summary evaluation through text. *CoRR*, abs/1406.5824, 2014.