REAL-TIME VISUAL QUESTION ANSWERING

May 14, 2020

Sofiya Semenova
University at Buffalo
CSE Ph.D. Student
sofiya@sofiya.io

1 Introduction

Visual Question Answering (VQA) offers a unique, difficult challenge at the intersection of natural language processing and computer vision. Within the past 5 years, the VQA problem has exploded in popularity among the computer vision, natural language processing, and machine learning communities. Even more recently, these communities have looked at tackling **Video** QA. However, despite the newfound popularity of visual and video QA, all currently proposed approaches to video QA still solely answer questions about static video segments. While there is still room for improvement in the model accuracy in the state of the art, a system that cannot run in real-time bars its own adoption in cases where timely answer generation on video streams is required.

Real-time video QA poses an additional challenge to regular video QA. For a system to be real-time, it must provide some timeliness guarantee. A user should be able to ask questions about something they "saw" a few seconds ago, and they should receive an answer within a reasonable amount of time. The problem, then, becomes one of processing an incoming video stream, saving video data, and searching across old video data quickly. On the video QA side, the system must perform feature extraction on a video, encode the question, and run a neural net to generate an answer, which frequently comes with an attention component across the entire video segment.

My project is a video QA system that processes incoming video streams and answers questions about them, where the answer is one word chosen from a set.

1.1 Related Work

Currently, there are no papers for real-time video question answering. [6], published in 2010, does achieve a real-time video question answering system by leveraging paid workers on Amazon's Mechanical Turk, but their system obviously does not use computer vision techniques.

Video question answering, sans the real-time, is itself a new research area. [3] introduces the model that I used for this project. Their approach generates appearance and motion-based features for the input video segment, and gradually refines attention over the dual features to generate an answer. Their model uses only questions as input text, and generates a one-word answer to the question. [7] introduces an encoder-decoder model using GRUs to learn the temporal context of an input video. Their model answers "fill in the blank" questions with multiple choice answers. [8] and [9] both use external data in their models (video descriptions and attributes, respectively) to inform the answer generation process. On the other hand, [10] is the first to propose a model to generate *open-ended* answers. Their model is a combination of a GRU network that performs adaptive video segmentation, a GRU network that performs hierarchical attention across the video segments and input question, and a reinforced decoder network that generates answers.

2 System Design

My real-time video QA system contains two parts: a *question answering component*, and a *caching component*. The question answering component is responsible for ingesting a video and a question and running a neural network to generate an answer. The caching component turns the incoming video stream into usable chunks of data that can be fed into the question answering component.

As a video is played, the caching component saves video frame data into *video chunks*. When enough frames have played, the caching component generates features for the video chunk, deletes the frames, and caches the data. Then, when a question is asked, the video question answering component uses the generated features in the cache to answer the question.

The system was written in Python, using Tensorflow for the video QA model.

2.1 Question Answering Component

For the Video QA component, I used the model described in [3], which introduced a gradually-refined attention component over both motion and appearance features.

Their model takes as input VGG15 and C3D features for a given video clip, as well as a word embedding for a question. They use VGG16 features to capture appearance in the video, and C3D features to capture motion. To generate an answer for a video-question pair, the model turns the question into a word embedding, which is then sequentally fed into an LSTM and an attention memory unit (AMU). This way, their model gradually refines the overall attention across the entire question and over both appearance and motion features. Then, after all the attentions are computed, they fuse together the attentions and features and plug the result into a classifier based on a pre-defined answer set.

This model is largely incorporate into my system as-is. As video frames are processed and a question is asked, the caching component sends the cache and the question to the video QA component. The video QA component, in turn, runs the model on the most recent chunk in the cache. If neither the apperance nor motion weight in the result is high, the video QA component runs the model again on the earlier video chunk in the cache.

2.2 Caching Component

To adapt the system for real-time performance, I create a caching component which is responsible for automatically creating and caching *video chunks* from the incoming video stream. As the video stream runs, the caching component saves the raw pixel values of each frame it sees and keeps track of how many frames it has seen. Every 50 frames, the caching component sends the current chunk to the cache and creates a new chunk. At this step, C3D and VGG16 features are extracted for the saved video frames, the saved video frames are deleted and old chunks are evicted from the cache to free up memory. Chunks are evicted on a FIFO basis.

When a question is asked, the caching component sends the cache and the question to the video question answering component, which returns the generated answer. When receiving a question, the caching component will most likely be in the process of saving frames for a video chunk. Thus, it has the option to either create a "short" chunk with the frame data it currently has (potentially creating a chunk that is far too short and doesn't have meaningful visual information), wait until the chunk is available (potentially needlessly increasing the time to get an answer), or proceed to question answering with a stale cache (potentially leading to the wrong answer if the cache is too stale). I adopt the following policy in these cases:

1. If the amount of saved frames is between than 50% and 75% of the total frames of a "normal" chunk, manually truncate the chunk and add it to the cache.
2. If the amount of saved frames is fewer than 50%, use the stale cache.
3. If the amount of saved frames is larger than 75%, wait until the chunk available.

2.2.1 Configurable Variables

There are a few hard-coded variables that I arrived upon by manually optimizing the system on my experimental setup, that could be configured for different results:

- **Cache size** The amount of video chunks to save in the cache. I chose 6, but the "correct" choice depends on the amount of memory on the device and the size of the chunks.
- **Chunk size** The amount of frames in a video chunk. I chose 50, but the "correct" choice depends on the memory on the device, the amount of chunks in the cache, and how often the cache is evicted.
- **Cache eviction** How often to evict the cache. The cache has an opportunity for eviction whenever a question is asked. However, eviction takes a non-trivial amount of time, so the relationship between speed and memory should be taken in consideration. I evict the cache when the amount of chunks in the cache exceeds 2 more than the cache size.
- **Cache on RAM vs disk** I implemented a cache both in RAM and on disk, in a sqlite database. I use RAM for my experiments because it is a lot faster than the database, but had to compromise on the size and quality of my cache to run comfortably within the amount of RAM on my experimental setup.
- **C3D feature extraction** C3D features are extracted by evenly splitting a video into equally-spaced clips of some length. The amount of clips to create and the length of each clip are up to the user. Obviously, the more clips are created and the longer each clip is, the more data must be saved in memory. I chose to create a clip every 5 frames with a length of 5 frames each, for a video chunk that is 50 frames

long. Thus, half the frames are not represented in the C3D feature data. This has ramifications for model performance, but stays within the available memory on my computer.

3 Experiments

3.1 Dataset

For my experiments, I used the MSVD-QA dataset generated in the Gradually Refined Attention paper [3]. MSVD-QA uses clips from the MSVD dataset [1], which provides clips and annotations from YouTube. To create question-answer pairs for each video, they use the tool proposed in [2] to automatically generate question-answer pairs from video annotations. The resulting dataset has 1970 video clips, and 50,505 question-answer pairs. The training, validation, and test splits comprise 61%, 13%, and 25% of the dataset, respectively.

3.2 Experimental Setup

Because my system is meant to run on real-time video streams and sufficiently answer questions in a timely manner, creating an adequate experimental setup is a little more complicated than sequentally inputting question-answer pairs and videos into a neural network. To capture the "real-time"-ness of the system, I first create an interleaving of randomly chosen videos from the dataset, where each video is played after the other with no delay. Then, the video interleaving is "played" frame-by-frame and the system is allowed to ask a random question every 3 seconds. The random question is chosen from the sets of questions for the currently playing video and the two videos that played before. Because the results of a run are not conclusive, I run the system 10 times for each question type for 30 minutes and average the results of all the runs.

The experiments are run on a AWS t2.large ec2 instance which has 8 GB of RAM, 2 vCPUs, and no GPU.

3.3 Results

Table 1 summarizes the accuracy of my system compared to those reported in [3]. E-VQA, E-SA, and E-MN are baseline Video QA models introduced in [3]. GRA-O is the gradually refined attention model in [3] that my system extends. The authors did not release their trained model, so I had to re-train it myself. Thus, GRA-R represents my trained model on the same dataset and code as GRA-O. RT VQA is the real-time adapted system for this project.

For E-VQA, E-SA, E-MN, GRA-O, and GRA-R, the accuracy is computed by running each model on every question-answer pair in the MSVD-QA dataset. For my system, RT VQA, the accuracy is computed using the experimental setup discussed in Section 3.2. For all, the accuracy of each is the average of questions that were answered correctly across all the questions tested. An answer is marked correct if it is the same as the

Accuracy						
Methods	what	who	how	when	where	ALL
E-VQA	.097	.422	.838	.724	.536	.233
E-SA	.150	.451	.838	.655	.322	.276
E-MN	.129	.465	.803	.707	.500	.267
GRA-O	.206	.475	.835	.724	.536	.320
GRA-R	.199	.436	.706	.724	.464	.298
RT VQA	.214	.356	.698	.714	.571	.269

Table 1: Results of training the model on the MSVD-QA dataset. Above the horizontal split are the results reported from [3] for their model (GRA-O) and three baseline models. Below the horizontal split are my results.

answer in the dataset – there is no notion of being "close, but not quite" in the evaluation. The reason the *ALL* accuracy does not equal the average of each of accuracies for each question type is because there is not an equal distribution of each question type in the dataset.

There is a slight discrepenacy between GRA-O and GRA-R for most of the question types, where GRA-R has a resultant total accuracy that is significantly lower than GRA-O, but still better than the baselines. My system (RT VQA) performs slightly worse than GRA-R, but still better than the baselines.

The average answer generation time in my system takes 0.383 seconds.

3.3.1 Performance v. Memory

As I mentioned earlier, there is an interplay between model performance and system memory, particularly for motion-based questions that require C3D features. For C3D feature extraction, I used a chunk size of 50 frames per chunk, 5 evenly distributed C3D clips across each chunk, and 5 frames per clip. This necessarily loses 50% of the frame data for each chunk. However, raising the amount of clips or frames per clip ends in the experimental setup running out of memory. Moving the cache to disk alleviates this problem, but increases answer generation time.

If real-time requirements are loosened by a few seconds, moving the cache to disk is a reasonable solution. Otherwise, a lowered model performance is unfortunately a necessary requirement for speedy answer generation.

3.3.2 Moment Localization

My initial project idea included a *moment localization* component, which would have been used to find the most relevant video chunk for a given question. The code that I used for this component was that of an activity recognition-based moment localization neural net from [4], pre-trained on the Charades-STA dataset.

Integral to the moment localization neural network is capturing spatio-temporal features from a video segment. While my system already captures these kinds of features by using the C3D net, [4] was trained on *I3D features*. Unfortunately, after incorporating their model into my system, I found that I3D feature extraction was prohibitively slow, at around 15 seconds per video segment.

They mention in their paper that technically, training the model with C3D features should be possible and will glean results similar to training on I3D features, but I didn't have time to re-train their model so I chose to turn this part of the system off. Further, given the tradeoff between accuracy and RAM/disk space, it may not actually be advantageous to run the moment localization neural net if the user's device contains enough memory.

3.3.3 Dataset Choice and Evaluation Metrics

My experimental results may be more inconclusive than they initially appear due to logical errors in the MSVD-QA dataset and in misleading evaluation metrics.

While observing my experiments, I noticed that the MSVD-QA dataset I used might not be ideal because it contains some logical inconsistencies, all of which I believe are the result of generating question-answer pairs instead of using human-annotated question-answer pairs. In particular, the following problems occur quite often:

1. Similar logical terms are distinctly different answers, and answering with the wrong one does not count as correct. E.g. "man" vs. "guy", "lady" vs. "woman".

2. Some vague terms are preferred over a more specific term. E.g., "someone", "somebody", "thing".

3. At least one case of a nonsensical word – "slouse".

4. The difference between "who" vs "what" questions are not as cut and dry as may appear, so comparing the accuracy of models on these types of questions is misleading.

5. Because questions are automatically generated, some questions don't make logical sense. E.g., Q: "When did people get?" A: "Stage", Q: "When do a horse and the person riding it stand up?" A: "While".

These dataset issues are likely creating misleading results, and I think it would be more conclusive to test the system on a dataset that is at least modified to remove these logical inconsistencies, and ideally is created with the work of human annotators.

Further, the reported accuracy of each model is misleading because the result reports how many answers were correct as a percent of total answers, with no measure of correctness other than a binary true or false. For example, if the correct answer is "sidewalk" but the model guesses "street", the model is penalized equally to if it had guessed something completely wrong. Thus, perhaps the accuracy of the all models across the board are misrepresented, if it is easy for them to near-miss frequently.

3.3.4 Inconsistent Results

Another potentially surprising outcome in my experiments was that the accuracy is wildly fluctuating across different runs. Slightly different results for different runs would make sense because the experiments are created by randomly interleaving videos and randomly asking questions about them, but the overall results fluctuate between as low as .05 to .8 depending on the interleaving.

These highly fluctuating results are probably due to a combination of the random video/question interleaving and the dataset itself. Because a core assumption of my system is that real-time questions will already be somewhat temporally localized, so the system only searches recent video chunks, it's likely that some random interleavings will generate too many "out of bounds" questions. The set of viable questions is created from the current and previous two videos, chunks are created with fixed lengths, and the cache evicts chunks very liberally ... yet a long video that played hundreds of frames in the past could have many questions about it and unfairly skew the randomly chosen questions. This is likely one of the causes of the extraordinarily bad runs.

The dataset and random interleavings can also falsely raise the accuracy because the dataset contains many easy questions that can be gamed – for example, "man" and "two" tend to be pretty safe answer for "who" and "how many" questions, respectively. Thus, if the random question interleaving contains many of these questions, then the systems performs unusually well. This probably explains the few times the system accurately answered 80% of questions.

Another specific problem I've noticed is that the experiment often causes the system to ask a highly temporally localized question and occasionally the system has not yet "played" that section of the video yet, so the system answers incorrectly. This probably could be alleviated with a more robust experimental setup.

4 Discussion

My project for this course is a real-time video question answering system that performs only slightly worse than the vanilla video question answering component that it builds upon. This shows to me that a real-time video question answering system can be possible, but that the parameters for the system should be optimized, especially given the interplay between memory and model performance. In the future, I plan on testing my system with more datasets (particularly, the ActivityNet QA dataset [5], which specifically was created as an improvement upon the MSVD-QA and MSRVD-QA datasets proposed in [3]).

Having never taken any machine learning course before, I appreciated the more in-depth look at particular problems that arise in neural networks and common solutions to them, especially when discussed in a computer vision context. I also appreciated learning about the current state of the art for several computer vision problems, since I came into the class only knowing the "classic", non-neural net solutions to those problems. Particularly important to me was understanding the rationale behind the choices an author could

make for their model, especially since I previously looked at machine learning models as impenetrable black boxes. Further, having the opportunity to work on a project and complete a survey was very helpful for my research, as I believe this is a viable research area to continue pursuing and plan on expanding my work to be a full paper.

References

[1] Chen, David L., and William B. Dolan. "Collecting highly parallel data for paraphrase evaluation." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.

[2] Heilman, Michael, and Noah A. Smith. Question generation via overgenerating transformations and ranking. No. CMU-LTI-09-013. CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST, 2009.

[3] Xu, Dejing, et al. "Video question answering via gradually refined attention over appearance and motion." Proceedings of the 25th ACM international conference on Multimedia. 2017.

[4] Opazo, Cristian Rodriguez, et al. "Proposal-free Temporal Moment Localization of a Natural-Language Query in Video using Guided Attention." arXiv preprint arXiv:1908.07236 (2019).

[5] Yu, Zhou, et al. "ActivityNet-QA: A dataset for understanding complex Web videos via question answering." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.

[6] Bigham, Jeffrey P., et al. "VizWiz: nearly real-time answers to visual questions." Proceedings of the 23nd annual ACM symposium on User interface software and technology. 2010.

[7] Zhu, Linchao, et al. "Uncovering the temporal context for video question answering." International Journal of Computer Vision 124.3 (2017): 409-421.

[8] Zeng, Kuo-Hao, et al. "Leveraging video descriptions to learn video question answering." Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[9] Ye, Yunan, et al. "Video question answering via attribute-augmented attention network learning." Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 2017.

[10] Zhao, Zhou, et al. "Open-Ended Video Question Answering via Multi-Modal Conditional Adversarial Networks." IEEE Transactions on Image Processing 29 (2020): 3859-3870.