CS230

DeepRemix: An Automated Remix Video Generator

Rey Barceló
Department of Computer Science
Stanford University
rbarcelo@stanford.edu

Abstract

DeepRemix is a deep learning pipeline which maps arbitrary videos to "remix" videos which match them in entities contained and actions performed. This is achieved in two steps: caption generation (accomplished with a CNN/LSTM) and caption matching (accomplished with NLP). Results are compared across a variety of caption matching schemes.

1 Introduction

A **remix video** is a type of video in which widely-recognized media (such as music videos and movie scenes) are recreated using stock footage. Traditionally, remix videos are assembled by hand, with massive amounts of labor required to pair each **source clip** with a **remix clip** (aka stock footage) that contains similar entities performing similar actions. In this paper, I propose DeepRemix: a method to generate remix videos automatically. DeepRemix's fundamental task is as follows: given a source clip of 3 seconds, it outputs a remix clip that matches the source clip's entities and actions.

DeepRemix performs this clip-to-clip matching in 2 stages: **caption generation** and **caption matching**. In caption generation, DeepRemix generates a caption for the clip using a CNN/LSTM encoder-decoder network called **im2txt** that will be described below; in caption matching, it matches the source clip's caption with the most appropriate caption of a remix clip. Once clip-to-clip matching is implemented, DeepRemix can also perform video-to-video matching for videos of arbitrary length.

Note: this project is being submitted for both CS 229A and CS 230. The caption generation component is my submission for CS 230 and the caption matching component is my submission for CS 229A.

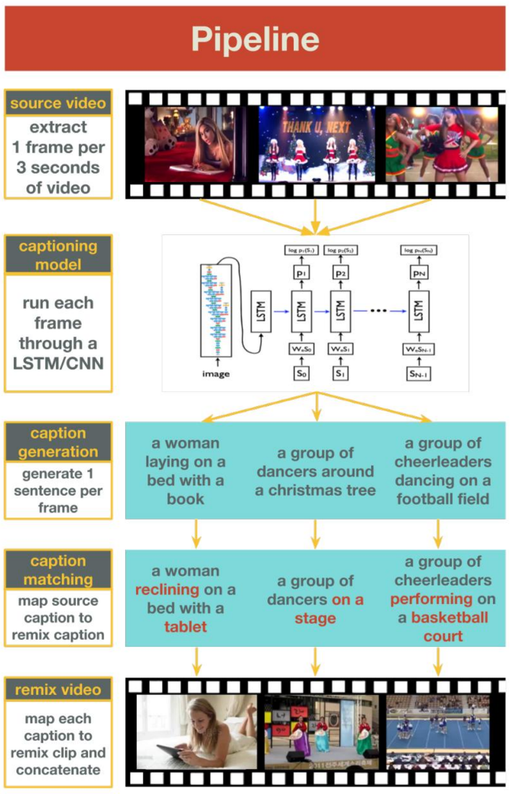

Pipeline
source video
extract 1 frame per 3 seconds of video
captioning model
run each frame through a LSTM/CNN
image
LSTM
caption generation
generate 1 sentence per frame
a woman laying on a bed with a book
a group of dancers around a christmas tree
a group of cheerleaders dancing on a football field
caption matching
map source caption to remix caption
a woman reclining on a bed with a tablet
a group of dancers on a stage
a group of cheerleaders performing on a basketball court
remix video
map each caption to remix clip and concatenate

Figure 1: The DeepRemix pipeline, from source video to remix video. There are 2 main stages: caption generation and caption matching.

2 Datasets

DeepRemix uses two datasets: a set of images labeled with captions used to train the image captioning model, and a set of 3-second videos of common actions used as the bank of remix videos.

The image dataset is the **Microsoft COCO** dataset: a set of over 300,000 images annotated with 80 different object categories and with bounding boxes for each object. Object categories include people, various kinds of animals, about a dozen foods, and a handful of household appliances.[1] The COCO dataset was used as the input to the im2txt model

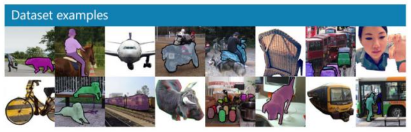


Figure 2: The COCO dataset.

The video dataset is the **Moments in Time (MIT)** dataset: a set of 1 million 3-second videos, each labeled with the action that occurs in them.[2] Actions include singing, dancing, and calling. Note that DeepRemix does not actually use the MIT labels; instead, it extracts frames from each clip and uses im2txt to caption them, as described below.

3 Methods

3.1 Caption Generation and the im2txt Model

DeepRemix's first step is to map a video to a sequence of captions that describe it. For this part, I used im2txt, an image caption generator made by Google Research and implemented in TensorFlow.[3] im2txt's architecture is shown at right.

Im2txt is an "encoder-decoder" network: it first uses a CNN to encode an image into a high-dimensional vector, then uses a LSTM to decode that vector into natural language representation. More specifically, im2txt is given an input image I and is trained to maximize the likelihood of producing a target sequence of words $S = S_1, S_2, ...$ that accurately describe the image.

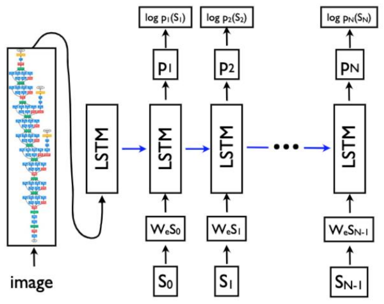


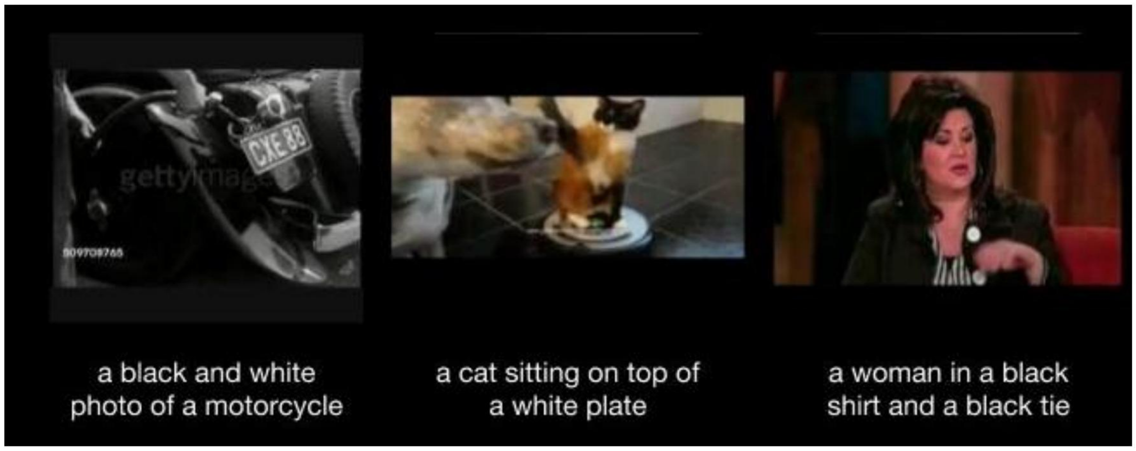
Figure 3: im2txt, the model used for image captioning. It uses a CNN to encode the image into a vector embedding, then a LSTM to decode the vector embedding into a sentence caption.

The CNN/LSTM combination is well-suited to image captioning because the problem bridges the fields of computer vision and natural language processing. CNNs are optimized to process arrays of pixel values, while LSTMs are designed to maximize sequences of words. Since training im2txt takes on the order of weeks, I used a pretrained model trained for 2 million iterations and downloaded from GitHub.[4]

Caption generation occurs twice: First, during preprocessing we caption all the frames of clips in the remix bank. Then, at runtime we caption all the frames of the source video. That way, we can match source clips and remix clips by the similarity of their captions. (This matching process will be described in the next section.)

For the remix video preprocessing, I used OpenCV to extract the middle frame from each 3-second MIT video. I ran im2txt on each frame to generate a caption, then made a CSV file of {caption, clip filename} pairs.

At runtime, the video is split into 3-second clips using a tool called moviepy.[5] As in preprocessing, OpenCV extracts the middle frame from each clip and feeds it into im2txt, generating a caption. Below are some real image/caption pairs.



CXE 83
gettyimages
909708765
a black and white photo of a motorcycle
a cat sitting on top of a white plate
a woman in a black shirt and a black tie

Figure 4: Real image/caption pairs. Notice that entity captioning is relatively expressive ("cat", "woman", "shirt" are correct but "motorcycle" is not). Action captioning is much less expressive ("sitting" is the only action in all 3 captions).

3.2 Caption Matching

Now that we have a caption for our source frame and a set of captioned images, it's time to match them. I explored 3 different approaches to caption matching: **entity matching**, **action matching**, and **bleu matching**.

For entity matching, I used NLTK to extract all entities from a source caption.[6] For example, a caption like "people standing next to a spinning top" would return the entities ("people", "top"). To match from these source entities to a remix caption, I iteratively searched for remix captions with the same captions, in order of rarity. If, for example, there were 10,000 remix captions with the word "people" but only 2,000 with the word "top", I would first search for captions with the word "top". Then, I'd look for a caption containing both words; if not, I'd use one with "top". I learned experimentally that prioritizing rare entities made the results more expressive; you wouldn't be that impressed if I matched a picture of people to another picture of people, but you might be more impressed if I matched a picture of a ring-tailed lemur to a picture of a ring-tailed lemur.

The action matching process was nearly identical, except I used NLTK to extract the verbs from each caption instead.

For the BLEU matching process, I calculated the BLEU score between the source caption and every caption in the remix bank. The BLEU score is an algorithm used to assess the similarity of machine translations; I used NLTK for this as well.

4 Experiments/Results/Discussion

To get quantitative feedback on my remix videos, I created a Google Form and sent it out to a wide group of people. Respondents were shown 4 pairs of clips. Each pair of clips contained a source clip and a remix clip generated by one of the following caption **matching schemes**: entity, action, bleu, and human. (The human scheme was the clip I felt best matched the entity and objects of the original clip.) Respondents were asked how well the clips matched, assessed against three **matching metrics**: how well the entities in each clip matched, how well the actions in each clip matched, and how well the clips matched overall. (The third question, how well the clips matched overall, was deliberately left vague so I could assess what kinds of videos people intuitively thought "matched" and whether the entity/action definition was accurate to this intuition.) Responses ranged from 1 (the clips did not match at all) to 5 (the clips matched extremely well). They were also asked how well the caption matched the source clip.

Below are the performances of the 4 matching schemes against the 3 different matching metrics, averaged across the 3 clips respondents watched. (In case you're curious, the clips were taken from the "Despacito" music video, the "Gangnam Style" music video, and the "Thank u, Next" music video. I chose music videos because they tend to contain a wide array of objects not found in a typical dialogue-heavy video, therefore increasing the expressivity of the captions and hopefully of the matching).

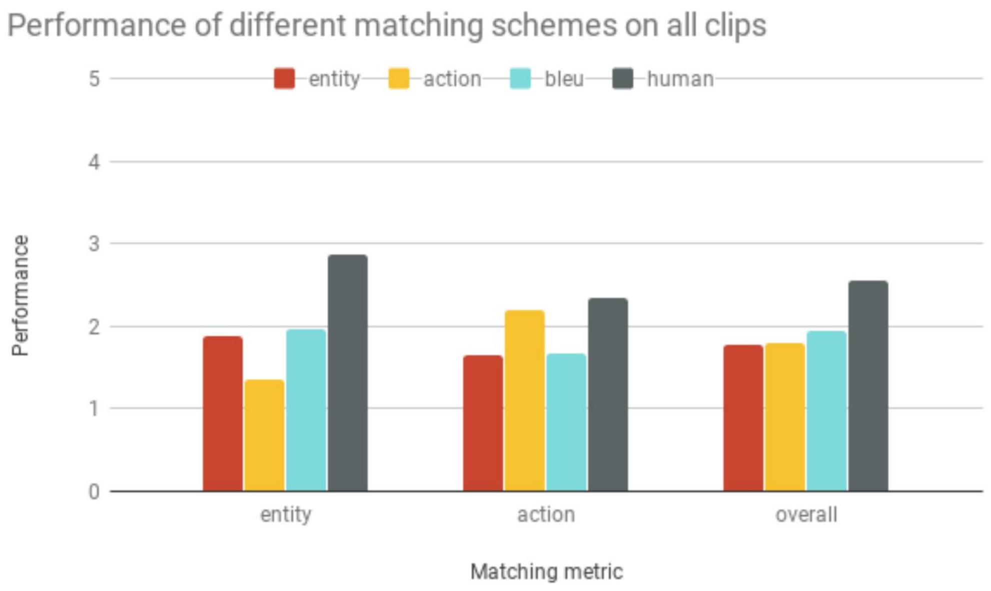


Figure 5: Performance of the matching schemes {entity, action, bleu, human} against the matching metrics {entity, action, overall}. Performance of 1 means the source and remix clip matched "not at all" against the given metric, while performance of 5 means the source and remix clip matched "extremely well".

Here are some sample source video / remix video matchings, as generated using the bleu matching scheme.

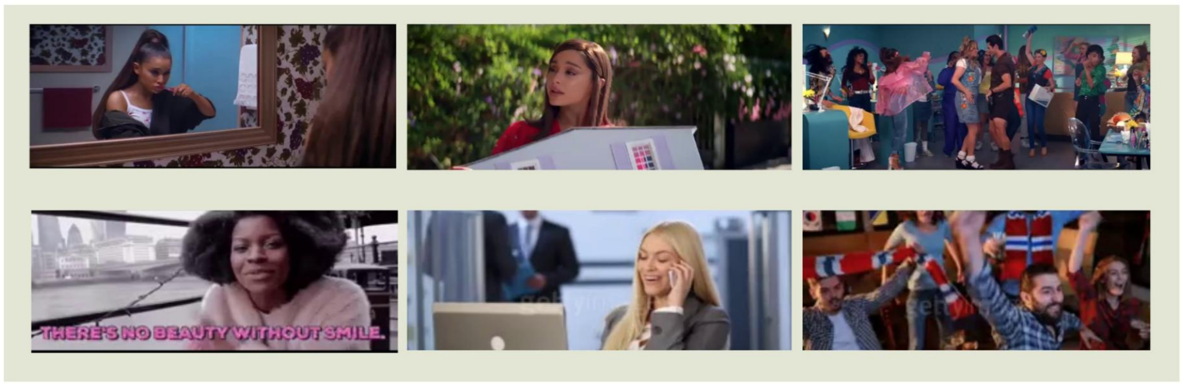


Figure 6: Source clips (top) and their corresponding remix clips (bottom).

As can be seen, the human caption matching scheme outperformed the automated schemes on all metrics, which means there's still improvement to be made! However, it's worth noting that the highest possible performance, which is by a human, still does not exceed "moderately" good matching. This shows that remix video matching is a hard and inherently subjective problem; responses had a wide variance which indicates that people don't tend to agree on which clips match.

It's worth noting that the entity matching scheme significantly outperforms the action matching scheme on the entity matching metric, while the action matching scheme also significantly outperforms the entity matching scheme on the action matching metric. To boil that down, the fact that schemes which encode a certain type of information (eg. entity scheme) do better at retaining that information (eg. on the entity metric) than schemes which do not encode that type of information (eg. action scheme) indicates that the schemes actually *are* encoding that information to some degree. The entity scheme actually encodes some information about the entities in the image, and the action scheme actually encodes some information about the actions in the image.

5 Conclusion

We've seen that by using a combination of computer vision and natural language processing, it is possible (but by no means a solved problem!) to create mappings from videos to other videos with similar entities and actions. My hope is that this system will be useful to documentarians or other filmmakers who need quick access to specific stock footage or who want to circumvent copyright restrictions by using open source footage which matches their desired footage.

In the future, I would like to perform video classification instead of image captioning in order to match clips. I think this will better capture actions and enable the remixing to be more expressive. I'd also like to conduct some user studies to determine exactly what factors cause people to consider whether two clips "match"; perhaps my hypothesis that similar actions and entities make for a good matching was incorrect.

References

[1] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." *European conference on computer vision*. Springer, Cham, 2014.

[2] Monfort, Mathew, et al. "Moments in time dataset: one million videos for event understanding." *IEEE transactions on pattern analysis and machine intelligence* (2019).

[3] Vinyals, Oriol, et al. "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge." *IEEE transactions on pattern analysis and machine intelligence* 39.4 (2017): 652-663.

[4] `https://github.com/KranthiGV/Pretrained-Show-and-Tell-model`

[5] `https://zulko.github.io/moviepy/`

[6] `https://www.nltk.org/`

[7] Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.