Recipe Generation from Videos using Pretrained Models

Hemanth Kumar Jayakumar [*1] **Avi Singhal** [*2] **Ishita Kundaliya** [*2]

Abstract

We attempt to tackle the challenging task of recipe generation from videos using only pre-trained models. We divided the process of recipe generation into various modules which include event generation, frame extraction, featurizing frames, removing frame redundancy, frame enhancement, frame captioning, and summarization using LLM. We used various pre-trained models to perform different tasks required to achieve desired results at each stage of our recipe generation pipeline. We used the temporal nature of videos, and the power of image embeddings, and harnessed the power of LLMs to extract meaningful content and generate recipes in an efficient manner. We have demonstrated the quality of the recipe generated using various metrics which highlight the impact of our work.

Keywords: Dense Video Captioning, PDVC, Cosine Similarity, BLIP, BERTScore, YOLO, RESNET, BARD, LLM
Code Links: Custom PDVC GitHub — Pipeline GitHub Presentation

1. Introduction

There are a lot of cooking videos which are available online and the duration of such videos varies. People with some experience in cooking do not need visual aspects in order to make the food, written guidelines are enough for such people. When recipes for such food items are not available, people are left with no choice but to watch the complete video making it time-consuming and frustrating. We intend to try to solve this problem by capturing the key information in the video and presenting it in a recipe format. The task we are trying to achieve is similar to video summarization, so a user can directly input the video and get the recipe without having to spend time watching the video.

[1]Department of Computer Science, Rice University, Houston, Texas, United States [2]Credits to Dr. Vicente Ordóñez .

2. Related Works

There are several works on recipe generation from videos but all of them involve extensive training of large models to generate results. (Nishimura et al., 2022) uses an end-to-end transformer-based joint training for event selection and sentence generation for recipes, the main idea the authors incorporate is recipe story awareness i.e. the model should have a memory about past events when generating recipes for the next events. (Wu et al.) introduces an ingredient recognition module that uses the predicted ingredient information for sentence generation. They also use the previous step's visual information for the next step in addition to the ingredient information, thereby achieving better results than the prior work. (Fujii et al.) treat this task as an image captioning task, they use a sentence encoder and an image encoder, and they combine the embedding given by the encoders, and feed it to a decoder which is a LSTM. This complete module is trained in an end-to-end fashion.

3. Methodology

We shaped our project in a way that will require almost no training while maximizing the use of pre-trained models. We divided our recipe generation into multiple modules such that each module is almost independent of the other and we can easily change any module without affecting the other. This gives us the flexibility to try out different approaches for some parts of the recipe generation and to easily adapt this pipeline to future state-of-the-art models. The different modules of the pipeline are explained below:

Event generation: A video is just a sequence of images that are displayed rapidly in succession to create the illusion of motion. So there is a high correlation between neighboring images, this information is redundant and can be skipped. Overall in a video, there are only a few key events, and extracting these is done in this module. We employed Parallel Dense Video Captioning(PDVC) (Wang et al., 2021) model for generating events. The model outputs time stamps of key events in a video, it also generates the captions for those events, but these are irrelevant to our pipeline and therefore discarded. The advantage of using PDVC is that it does not rely on handcrafted methods for event selection, the model is trained in an end-to-end fashion to identify the key events in a video, the reason it is named "Parallel" is

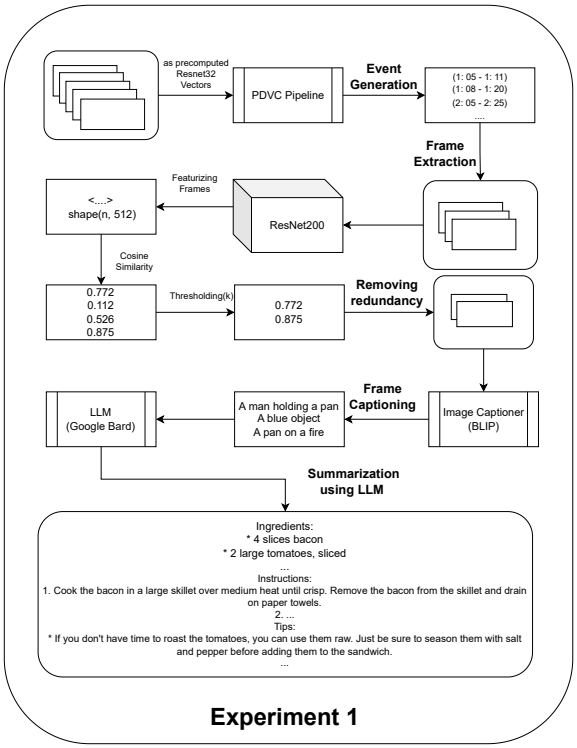

Figure 1. Experiment 1 pipeline showing the flow of data to each process

because the intermediate information is fed in parallel to a localization head(which does event selection) and the captioning head(for sentence generation), so both of these heads are trained in parallel instead of being trained serially. This makes the two tasks deeply interrelated and mutually promoted through optimization. The reason that we are not using the captions generated by PDVC is that the pre-trained model that is released has been trained on the Activity-Net dataset(Caba Heilbron et al., 2015) which is a dataset consisting of human activities. Since our dataset is comparatively much smaller than Activity-Net it was feasible and also a good approach because cooking involves a lot of human activities (hand, dish movement, etc), transferring those movements to cooking actions. The captions which are generated by PDVC after finetuning were very poor, where most of the tokens are very different from the ground truth. We fine-tuned the model for 30 epochs, upon which there was no significant decrease in the training and test loss. For using PDVC, we used (Zhou et al., 2018b) to acquire temporal sensitive network features (TSN) of the videos, comprising of a Resnet-200 for extracting the appearance features and a BN-inception network for optical flow features for each frame. Our PDVC model gives out 10 proposal events in the output for each video. PDVC was deployed to reduce redundancy in events proposals. However, as there is still a chance of redundancy which we removed in the next module, the event generation module will take the videos as input and produce the time stamps (start and end times) of important events in each video. This output is in the form of a JSON file and will be used in subsequent modules. One of the hyperparameters in this module which can affect performance is the number of event proposals, we do not have control over this as we use a pre-trained model.

Frame Extraction: Using the output from PDVC, we sample the frames in each of the time stamps given by PDVC. Since there can be repetition in the frames, after sampling we have sorted the frames as per the time at which they were sampled so that frames closer to each other occur together. The input to this module is the time stamps given by PDVC, the output is the sampled frames.

Featurizing Frames: Since there can be redundant frames due to the temporal nature of videos, we will remove the frames which are very similar. For measuring similarity, we use the cosine similarity metric. For using cosine similarity, we will need to vectorize the images. We have used Resnet32 for this purpose. The input to this model is the sampled frames, for the output, each frame has been converted to a vector. In this module, the threshold used for similarity is a hyperparameter and can influence the performance.

Removing Redundancy: We use cosine similarity over the adjacent frames generated by the previous module. Once the similarity falls within a certain threshold, we consider that frame as a new frame. If the similarity is high, we do not consider it as a new frame. The output of this module is a reduced number of frames based on the threshold of similarity.

Frame Enhancement: We employed YOLO for object detection of the frames from the previous module. YOLO returns the objects as well as the coordinates for the bounding boxes of the objects. We added the object labels as well as the bounding boxes to the frames. This was done as an experiment to enhance the frames for captioning purposes.

Frame Captioning: We use BLIP model for captioning each of the frames generated by the previous module. For each frame, we append the caption generated by BLIP. In the end, we get captions for each of the frames.

Summarization using LLM: We used BARD model for summarizing the output of the previous module. We performed prompt engineering to get the best possible recipe as output.

Metric Evaluation: Since we did not perform any training, we do not think it is meaningful to use METEOR score as the metric, METEOR score is the most widely used metric for comparing generated text. However, since we are using different techniques for generating the recipe, it is not nec-

essary to get the exact words in the output. Since METEOR tries to check for exact word match and positioning, it does not make sense for our project. For our project, the goal is that the generated recipe should convey the same meaning as the ground truth. We use the BERTScore metric for evaluation, as it converts the text to embeddings and checks for pairwise similarity between reference and generated text. This metric is more meaningful as words that are similar will have similar embeddings.

We used the Youcook2 dataset (Zhou et al., 2018a), which is one of the largest cooking video-and-language datasets. It is a labeled dataset i.e. for each video in the dataset, there are instructions for the recipe in the video. We use these instructions as the ground truth for our evaluation.

4. Experiments

We have performed 4 experiments for recipe generation.

In the first experiment, we used the pipeline as described in the methodology and shown in Figure 1. We used the event proposals given by PDVC and extracted the important frames from those proposals using cosine similarity. Then for each of these frames, we obtained the captions using BLIP and passed these captions to BARD model.

Experiment 2 was very similar to experiment 1 but we also used YOLO to enhance the frames. An example of this is shown in 2.We sent these enhanced frames to the BLIP model for captioning. The main idea behind using YOLO was that object labeling and the bounding boxes would provide more information to BLIP and improve the captioning. Our intuition is that the bounding boxes would in a way tell BLIP that these objects are of significance in the frame.

In Experiment 3, we did not completely use the pipeline described in the methodology. Instead, we used some portions of it. We only used the transcripts of the videos as the caption for a particular video. We then sent these captions to the ”summarization using LLM” module. The main idea behind this experiment was to understand the significance of the captions in the recipe generation. We wanted to identify if the captions were hindering our results as we suspected BLIP was not giving us the desired captions.

In Experiment 4, we obtained the transcripts for the videos. Since the BLIP model was giving out generic captions for each frame, the captions did not give much information about food items, ingredients, or steps being performed in that frame. The captions given by BLIP were also very repetitive, providing syntactically and semantically incorrect text, which would not provide meaningful information about the recipe. Instead of using the BLIP model for captioning each frame, we found the time stamp for the particular frame under consideration and concatenated the transcript of the time just before and after that timestamp. We believe this could provide more meaningful captions for the particular frame.

The figures demonstrating our experiments 2,3 and 4 have been included in the presentation. Also the guidelines for running the experiment has been added in the presentation, we have added videos for ease.

Experiments	Precision	Recall	F1	METEOR
Experiment 1	0.776	0.814	0.795	0.15
Experiment 2	0.77	0.794	0.785	0.147
Experiment 3	0.819	0.881	0.848	0.35
Experiment 4	0.805	0.815	0.81	0.11

Figure 2. Image after YOLOS-tiny portrayed with a bounding box drawn

5. Results

We have taken the average for each metric over all the videos on which we ran the experiments. It can be seen that all the metrics obtain similar values for experiments 1 and 2, and it confirms that YOLO is not beneficial for enhancing the frames. To prove this claim, we show the output of BLIP with and without using YOLO in the attached presentation in the Appendix section. The captions produced are identical, which confirms the ineffectiveness of YOLO for this task. Experiment 3 gave the best scores in each metric, this highlights the importance of captions of each frame. If the captions are more relevant, we can obtain a better recipe. These results made us recognize the power of the transcripts and we incorporated these transcripts in our pipeline. The results of experiment 4 are much better than experiment 1,2 and are very close to experiment 3. All the related works that we have referenced to do not consider BERTScore as a metric for recipe evaluation. Most of them use METEOR score. Since the METEOR score checks for exact word matches, it is unsuitable for our project. The related works perform extensive training and hence it is suitable to use the METEOR metric. Since we have not performed any kind of training, BERTScore is the most relevant. In Figure 3, we present the recipes generated by experiment 1, experiment 3, the concatenation of the captions generated by PDVC and the ground truth recipe. The PDVC outputs are not usefull in any way and hence were not used by us.

1. Heat the olive oil in a large skillet over medium heat. 2. Add the chicken and cook until browned on all sides. 3. Add the onion and garlic and cook until softened, about 5 minutes. 4. Stir in the curry powder, cumin, turmeric, cayenne pepper, salt, and black pepper. Cook for 1 minute more. 5. Add the crushed tomatoes, chicken broth, and yogurt. Bring to a boil, then reduce heat and simmer for 20 minutes, or until the chicken is cooked through. 6. Stir in the cilantro and serve.	1. In a large bowl, combine the chicken, flour, cornstarch, egg yolk, olive oil, sesame oil, black pepper, cumin, and salt. Mix well to coat the chicken. 2. Heat a large skillet or wok over medium-high heat. Add the chicken and cook, stirring occasionally, until golden brown and cooked through, about 5-7 minutes. Remove the chicken from the skillet and set aside. 3. To the same skillet, add the lemon juice, rice vinegar, chicken stock, ginger garlic paste, barbecue sauce, sriracha chili sauce, soy sauce, fish sauce, oyster sauce, sesame oil, and olive oil. Bring to a boil, then reduce heat and simmer for 5 minutes, or until the sauce has thickened slightly. 4. Return the chicken to the skillet and stir to coat in the sauce. Add the celery and onion and cook for 2-3 minutes, or until the vegetables are softened. 5. Serve immediately over rice or noodles.
add UNK UNK and UNK to the pan. add UNK and UNK to the pan. add UNK and UNK to the pan. add UNK UNK and UNK to the pan. add the UNK to the pan. add the UNK to the pan. add UNK to the pan. add UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK. add UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK. add UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK.	add sesame oil egg yolk salt black pepper paprika all purpose flour and corn starch to a bowl add pieces of chicken to the bowl and mix add olive oil to a heated pan add the chicken to the pan and stir pour the chicken into a bowl pour chicken stock oyster sauce sesame oil and olive oil to a pan add soy sauce fish sauce hoisin sauce ginger-garlic paste sriracha chili sauce and barbecue sauce to the pan add rice vinegar and lemon juice to the pan and stir add the chicken to the pan and mix add chopped celery and half an onion into the pan and mix

Figure 3. (Top Left) a. Output from experiment 1, (Top right) b. Output from experiment 3, (Bottom Left) c. PDVC Output (Bottom Right) d. Ground truth recipe [Human annotation]

6. Conclusion and Future Work

We tried generating recipes from videos using only pre-trained models and performed 4 different experiments to achieve good results. Using only the transcripts for recipe generation gave the best performance compared to other experiments. This highlighted the importance of the captions assigned to each frame. Combining the transcripts as well as our pipeline led to a similar performance as experiment 3. Transcripts are not available for all videos, hence in those cases our pipeline can be of great use and give decent recipes.

As a next step, we will try to train a model in an end-to-end fashion so that we do not have to worry about manually selecting the hyperparameters. This will also ensure training a model specific to a task which will perform better compared to using pre-trained models.

We also plan to add the step of matching each recipe instruction to an image/time frame between which it captions, allowing for better presentability and accountability of the output of the pipeline

References

Caba Heilbron, F., Escorcia, V., Ghanem, B., and Carlos Niebles, J. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding, 2015. URL https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Heilbron_ActivityNet_A_Large-Scale_2015_CVPR_paper.html. [Online; accessed 27. Apr. 2023].

Fujii, T., Orihara, R., Sei, Y., Tahara, Y., and Ohsuga, A. Generating cooking recipes from cooking videos using deep learning considering previous process with video encoding. In *Proceedings of the 3rd International Conference on Applications of Intelligent Systems*, APPIS 2020: 3rd International Conference on Applications of Intelligent Systems. Association for Computing Machinery. ISBN 1-4503-7630-4.

Nishimura, T., Hashimoto, A., Ushiku, Y., Kameko, H., and Mori, S. Recipe generation from unsegmented cooking videos, 2022.

Wang, T., Zhang, R., Lu, Z., Zheng, F., Cheng, R., and Luo, P. End-to-end dense video captioning with parallel decoding, 2021.

Wu, J., Pan, L., Chen, J., and Jiang, Y.-G. Ingredient-enriched recipe generation from cooking videos. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, ICMR '22: International Conference on Multimedia Retrieval. Association for Computing Machinery. ISBN 1-4503-9238-5.

Zhou, L., Xu, C., and Corso, J. J. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pp. 7590–7598, 2018a. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17344.

Zhou, L., Zhou, Y., Corso, J. J., Socher, R., and Xiong, C. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8739–8748, 2018b.