Automatic Generation For Natural Language Queries in Ego4D Dataset

Simone Scalora, Lorenzo Lombardi, Matteo Gravile

{s329444, s330654, s322944}@studenti.polito.it

https://github.com/poliSimoneScalora/AMLproject2

Anonymous CVPR submission

Paper ID *****

Abstract

Egocentric Vision captures human interactions from a privileged viewpoint, via cameras mounted directly on the head of the user performing the actions, using large-scale datasets like Ego4D. The Natural Language Queries (NLQ) benchmark within Ego4D introduces the challenge of localizing temporal segments in videos based on natural language inputs, defining it as a temporal segment prediction task where, given a video clip and a query, the model must predict the segment where the answer is visible or deducible. In this work, we explore the NLQ task by exploring and comparing two architectures, VSLBase and VSLNet, using pre-extracted features from Omnivore and EgoVLP models and we propose an extension by integrating the Gemma-2b-it model to generate new queries from narrations in videos.

1. Introduction

Egocentric Vision, which captures human activities from a first-person view, has become an important research area with the release of large datasets like Ego4D. These datasets offer new ways to study everyday activities through wearable cameras, allowing tasks like recognizing actions, predicting future activities, and identifying parts of videos that match specific events.

One of the most challenging tasks in egocentric vision is the Natural Language Queries (NLQ) benchmark introduced in Ego4D. The NLQ benchmark defines a problem where models must predict the start and end times of video segments that answer specific natural language questions. These questions cover a wide range of topics, such as object identification, human interactions and actions being performed. This variety and the need for precise alignment between video and text make NLQ a challenging but very important benchmark for improving video understanding.

In this work, we address the NLQ task by using two architectures, VSLBase and VSLNet, which are specifically designed for video language understanding. These models are evaluated using pre-extracted features from Omnivore and EgoVLP. In addition to comparing their performance, we introduce an extension to the NLQ dataset using the GEMMA language model. GEMMA is a generative language model by Google trained for tasks requiring contextual understanding and natural language generation. Leveraging GEMMA, we generate synthetic queries from text narratives associated with Ego4D videos. This process allows us to expand the NLQ dataset, providing additional pre-training opportunities.

Our main contributions can be summarized as follows:

- A comparison of VSLBase and VSLNet on the NLQ benchmark using different features sets.
- Implement one variation of the VSLNet architecture and compare the performance with the baseline. Specifically, use different (non shared) encoders for the two modalities (video and text).
- Implementing GEMMA to generate queries to increase the NLQ dataset.

2. Related Work

2.1. Using NLQ

Analyzing egocentric videos using natural language queries (NLQ) is a growing research area with applications like understanding everyday interactions and building models to extract information from complex content [2] [1].

The NLQ task focuses on finding specific video segments that match a natural language query. This task is especially interesting in egocentric vision,

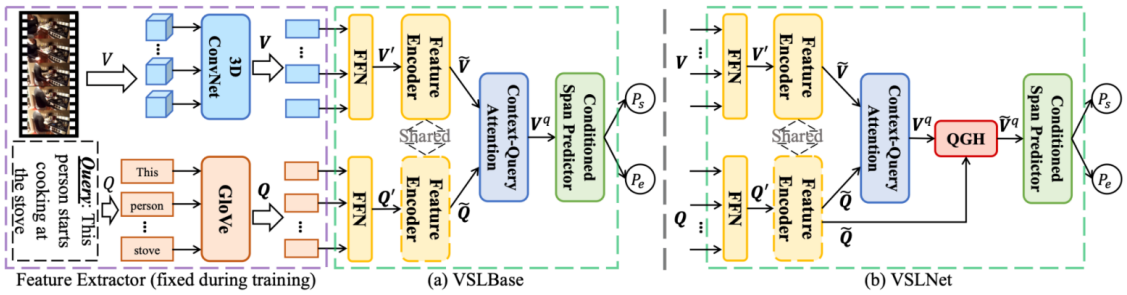

Figure 1. Architecture of the VSLBase and VSLNet models.

where the goal is to locate key moments in long sequences of daily activities by combining visual and textual information effectively. The main challenges include the length and complexity of videos, often filled with irrelevant details, and the need to create models that map textual queries and visual content into a shared space.

2.2. Datasets used

In this context, large-scale datasets are essential for training and evaluating models to address these challenges. Ego4D is a key dataset for egocentric vision research. It captures everyday interactions through first-person videos, providing over 3,600 hours of real-world content across various activities and scenarios. These range from interacting with objects to observing environments and people.

The Natural Language Queries benchmark in Ego4D is crucial for the NLQ task, offering around 19,000 annotated queries over 227 hours of video. These queries range from object-focused questions like "Where is X?" to activity-based questions like "What is the user doing?" Annotations include precise temporal segments, forming a basis for evaluating models on semantic understanding and accurate localization [4].

Although this project does not directly use Ego4D videos for training, it leverages the NLQ benchmark annotations and templates to evaluate performance. For training and visual data representation, we used pre-extracted features from two pre-trained models: Omnivore and EgoVLP, which adopt complementary approaches in computer vision.

2.3. Pre-trained Models

Omnivore is a multi-modal pre-trained model designed to analyze various visual data types, including images, videos, and point clouds. Unlike specialized models, Omnivore generalizes across diverse visual inputs, making it suitable for broad applications. Its architecture combines information from multiple sources to understand video content comprehensively. The version of Omnivore that we are using extracts high-level visual features using 16-frame temporal windows with a stride of 16 frames. This method reduces computational load by avoiding the need to train a visual model from scratch [3]. Omnivore's generalist approach captures universal visual patterns applicable across domains, including egocentric videos. It provides a baseline for comparing generalist feature-based approaches to domain-specific models.

EgoVLP is a pre-trained model specifically designed for egocentric videos, using a contrastive video-language approach. It is trained on a subset of Ego4D, leveraging weakly supervised text annotations describing activities in videos. EgoVLP captures rich, semantic representations that combine visual and textual information, enhancing its understanding of temporal and contextual relationships in videos. Its pre-extracted features include detailed temporal sequences, crucial for the NLQ task, where precise temporal localization is key. EgoVLP focuses on understanding human interactions, movements, and actions in daily life, offering advantages over general models like Omnivore. This specialization highlights the benefits of pretraining on egocentric data, providing better performance in tasks requiring natural language queries in videos [5].

By combining these features, we analyzed NLQ model performance using different representations, comparing a generalist model like Omnivore with a domain-specific model like EgoVLP. This approach offers a comprehensive view of how models handle egocentric vision challenges.

2.4. VSLBase and VSLNet

In the context of our work, in addition to the datasets mentioned, two architectures have been used for the Natural Language Queries task: VSLBase and VSLNet, both designed to address complex vision-and-language tasks with a focus on egocentric video sequences.

VSLBase is a neural network designed to tackle the problem of Natural Language Video Localization (NLVL) [6] using a span-based Question Answering (QA) framework. In this approach, the NLVL task is transformed into a series of SQuAD-style triplets (Context; Question; Answer), where the context is represented by the visual features extracted from the

video, the question is the linguistic query, and the answer is the video segment corresponding to the question.
The network consists of various modules: a feature encoder that projects the video's visual features and the linguistic query embeddings into a shared space; a convolutional block followed by a multi-head attention layer to produce the final representations; and a context-query attention module, which captures interactions between visual and textual features. Subsequently, a conditional span predictor, based on a unidirectional LSTM model, predicts the start and end boundaries of the video segment corresponding to the query answer. Start and end scores are calculated for each frame of the video, and the final answer is determined by selecting the boundary pair that maximizes the joint probability.

VSLNet, on the other hand, is an extension of VSLBase that introduces a Query-Guided Highlighting (QGH) strategy [6] to address the key differences between the NLVL task and traditional span-based QA tasks. In VSLNet, the target moment, i.e., the video segment corresponding to the query answer, is treated as the "foreground," while the rest of the video is considered "background." The system extends the foreground boundaries to include preceding and succeeding content, enhancing prediction quality and providing additional useful context.

This QGH process involves extending the start and end boundaries of the target moment, controlled by a hyperparameter α. Additionally, a binary classification module is built to predict whether a visual feature belongs to the foreground or background. The visual features are then weighted based on this classification to determine the relative importance of each part of the video. During training, the network is optimized using two loss functions: one for span prediction (as in VSLBase) and another for foreground and background classification, with the total loss minimized.

2.5. VSLNet with separate encoders

A variation of the VSLNet model is to use two separate encoders, where the visual and linguistic components are handled by two distinct components. The main innovation lies in separating the processing of visual information from that of language, allowing each encoder to specialize in its own domain. The visual encoder is responsible for extracting visual features from video frames, while the linguistic encoder focuses on understanding the textual query. The outputs of both encoders are then fused, where an attention module computes the relevance of visual features in relation to the query, thereby improving the localization of relevant video segments.

3. Methodology

3.1. Narrations

The work that we have done is for an automatically generated extension for the dataset. Until now we have only trained using the NLQ benchmark dataset. While this is comprehensive of around 227 hours of annotated videos, the real Ego4D dataset stores more than 3500 hours of video. In order to have a larger dataset to work with we can expand what we already have in an autonomous way.

We can use an LLM to create Natural Language Queries in order to augment the dataset. We will base everything around the file narrations.json. Narrations are basically the description of every single thing happening in the video, referring to either the Cameraman (c for short in the narrations) or to the people or things surrounding him. These all range from actions that the cameraman is doing to how people interact with the environment. By studying the narration json formatting, we can extrapolate all the data that we need. First, we are going to choose some videos to begin with present in the Omnivore dataset. We want to use videos that are not already present in the test and val splits. In our case we chose to go firstly with some randomly selected videos that had narrations, by checking for a large number of narrations, while some other videos have been chosen manually (mainly for debugging reasons when we were troublueshooting).

By using the Ego4D Visualizer tool we were able to easily identify a set of videos that had little to no Natural Language Queries annotations, while still having a good amount of narrations. Once the videos are chosen they are stored in an array. Keep in mind that we are considering Omnivore video features so all videos must be present in the Omnivore dataset, hence why we are randomly accessing videos from there. We are then loading the narrations from narrations.json, and then we are going to process the file. What we are doing is transforming the file into a python dictionary and then checking the keys linked to first the video uid itself, then narration and pass_1

This allows us to access the data of the single narration. In order to craft an NLQ query we are going to need the timestamp second (the second that said narration happens), the timestamp frames, the narration text (which we are going to "feed" the LLM with) and the annotation uid. We are selecting M windows of N length for each video. Everything is then stored in dedicated arrays. Before the LLM we are going to process the narration texts a little bit more, making them a little bit more readable for someone out of context (for example the C is always replaced with cameraman).

3.2. Usage of Gemma

We are using the model Gemma-2b-it for query generation. HuggingFace-cli (hence the need of a token for running) is used for accessing Gemma. After accessing the model we are executing a standard code in order to setup Gemma (Setting up gemma in the code). By asking a specific question and concatenating it with each group we get 5 questions each that are coherent with the narrations. The prompt that we are using for each iteration is: "Given the following narrations describing the actions of a person, generate a set of simple queries (one per line, five in total, only give the five queries for the answer) that could be answered by looking at the video segments corresponding to these narrations:" and then we will append the five narrations that we want to have a question generated from.

Please note that we are using the gpu for this since it is quite expensive performance wise and this is the best way to make it work for generating a large scale of data like we are trying to do in our case. We have also increased the max new token limit to mitigate the cut-out responses. Even if the prompt has always the same format and always asks the same thing, Gemma might generate different formats of output, so the final dataset might not be perfect. After many Gemma runs and some trial and error, we managed to find a string pattern that should always extract the five queries.

Once the model outputs everything we need we are going to do more post-processing in order to extract the newly generated queries from the output text. Here we'll also check for errors, since Gemma is not perfect and sometimes won't generate a set of 5 questions, or doesn't generate anything at all. This only happens once every few hundred outputs so it's irrelevant in our case, and the code is written so that in that case the output is skipped. A list is printed that signals every output that failed to produce a response in the format that we expected.

3.3. Creating the annotations

We finally have the final json part where we use the previously obtained data from narrations.json (that we have stored in dedicated arrays) and the newly generated and processed queries in order to generate the NLQ annotations. For the time we used the next question's timestamp as the end time for each query, and we are also adding a second in case of mistakes in the annotation of timestamps. If the query is the last of its window, it's going to estimate a 2 second window of time (start has -1 second, end has +1 second).

The annotations will just be composed of the start and end times, start and end frames and the query that we have generated. The rest of the data (template, tags ecc.) likely needs human effort to be correct, so for automation's sake we'll just leave them as null. Everything is generated in the same order that it has been processed so the file will have the consecutive narrations of each window one after the other in order. In this phase we are also keeping placeholders for the video metadata (start and end times for the whole video, and clip uid). Every set of queries is then annotated with the video uid that they belong to and also they all are marked as "train" in the split field.

We are then finally extracting the data from Ego4D.json. This file has metadata of all the videos present in Ego4D, so we can extract the full video length from here. We also use this file to extract clips for each video uid, so we're matching the video uids between ego4d and our file to update clip uid in our dataset (previously it had a placeholder)

We also have implemented a small snippet of code so that we can merge results. Since all of this was pretty heavy to run (and mostly also because of colab's GPU timeouts) we split the work of generating the dataset and then we used this tool to get the final dataset.

To manually check for errors on this part we have exploited the Ego4D Visualizer Tool. By picking a random video and selecting a random narration in that video, we use the visualizer to see if the question produced by Gemma is reasonable, and that the timestamps are correct.

4. Experiments

4.1. Parameters

For experiment we will be training the VslNet model with the generated queries and then with the real ones, and we're going to compare the results with the others variations that we've done. In the case of the generated dataset we have used a window length of 5 (N parameter), so that all queries will be in packs of 5 that are consecutive, and we are picking 20 random windows per video (M parameter).

The number of videos in the dataset produced during a normal run is set to 20, but since we fused together various smaller datasets the number will be higher in the dataset found in the repo. Since some of the videos will be cut out because we are selecting only videos which have 100 or more narrations. We can also select videos with fewer narrations, as long as we decrease accordingly the M parameter to mitigate data redundancy (with fewer narrations there is more risk of a duplicate).

Baseline	IoU=0.3 (%)		IoU=0.5 (%)	
	r@1	r@5	r@1	r@5
VSLNet with slowfast features	5.45	10.74	3.12	6.63
2D-TAN with slowfast features	5.04	12.89	2.02	5.88
VSLNet with Omnivore	6.42	12.95	3.63	8.18
VSLNet with EgoVLP	6.40	13.83	4.02	9.21
VSLBase with Omnivore	7.07	13.11	4.05	7.97
VSLBase with EgoVLP	**7.15**	**15.56**	**4.64**	**9.86**
Non shared encoders with Omnivore	6.19	12.20	3.66	8.05
VSLNet with augumented dataset (using our extension)	5.58	12.49	2.89	7.82
VSLNet with pre-train (using our extension) and fine tuning (using real annotations)	1.54	4.59	0.90	2.45

Figure 2. Results.

4.2. Metrics

The performance is evaluated using four key evaluation metrics that combine temporal localization accuracy with ranking effectiveness: Val/Rank@1 mIoU@0.3, Val/Rank@1 mIoU@0.5, Val/Rank@5 mIoU@0.3 and Val/Rank@5 mIoU@0.5. These metrics evaluate the model's ability to accurately predict the temporal boundaries of video segments based on natural language queries.

- **Rank@k** The Rank@k metric measures whether the correct segment is within the top k ranked predictions for a given query.
- **Intersection-over-Union (IoU)** IoU measures the overlap between the predicted temporal segment and the ground truth segment. We consider a prediction correct if its IoU exceeds a given threshold.
- **Combined Metrics** By combining Rank@k and IoU thresholds, we computed the following metrics:
  - **Val/Rank@1 mIoU@0.3** The percentage of queries where the top-ranked prediction has IoU ≥ 0.3.
  - **Val/Rank@1 mIoU@0.5** The percentage of queries where the top-ranked prediction has IoU ≥ 0.5.
  - **Val/Rank@5 mIoU@0.3** The percentage of queries where at least one of the top five predictions has IoU ≥ 0.3.
  - **Val/Rank@5 mIoU@0.5** The percentage of queries where at least one of the top five predictions has IoU ≥ 0.5.

5. Results and Discussion

In this section, we present our results and discuss their implications. Figure 2 summarizes the performance of different models under varying IoU thresholds and Rank@k metrics.

The results indicate that using the **VSLBase with EgoVLP** achieves the best performance in every configuration, particularly in the Rank@5 metric, showing its effectiveness in this task. This suggests that EgoVLP works better for visual-linguistic retrieval tasks because it can combine visual and language features in a way that focuses on the viewer's perspective.

VSLBase with Omnivore and VSLNet with EgoVLP show similar performance. The differences between these models are relatively small, indicating that both approaches are effective.

VSLNet and 2D-TAN with slowfast features features show lower performance than the other models.

The augmented dataset and pretraining with the gemma dataset with fine-tuning with real annotations don't perform well. This means these strategies need more improvement to work better, as they have the lowest recall values overall (except for the slowfast VSLNet which is beaten by the augmented dataset)

5.1. Comparison

EgoVLP is made to combine vision and language from the viewer's perspective, which helps with tasks that need a deep understanding of both. This makes it

especially good for complex visual-language retrieval tasks, where the connection between images and text is important.

On the other hand, Omnivore is a good visual-language encoder but might not capture the same level of detail or context as EgoVLP, especially in tasks that need more specific information.

The augmented dataset and pre-training/fine-tuning did not significantly improve performance. This shows that these strategies need to be better optimized or that different augmentation methods should be tried. The lack of progress also suggests that the current pre-training and optimization approaches may not be the best fit for this task or dataset.

We also tried to train 2D-TAN with Omnivore features, but to no avail. The code seems to run fine but it simply takes too long to execute and Colab kept timing us out. We have annotated the modification that needed to be done in the GitHub Repo.

6. Conclusion

In conclusion, the results obtained from the Natural Language Queries (NLQ) task on egocentric videos indicate that the dataset generated using the Gemma-2b-it model, although an extension of the existing datasets such as Omnivore and EgoVLP, achieved performance that is comparable to the other two.

The analysis showed that the values obtained from the Gemma-2b-it dataset were very similar to those from Omnivore and EgoVLP, suggesting that the use of synthetic query-answer pairs for expanding the NLQ dataset does not negatively affect the model's ability to predict accurately. When evaluating the models, including VSLBase, VSLNet, and VSLNet with separate encoders, all demonstrated strong performance in localizing the relevant video segments in response to the given natural language queries, regardless of the dataset used.

This indicates that the approach of generating a custom dataset with Gemma-2b-it offers a valid alternative to traditional datasets, providing additional training opportunities without compromising the overall effectiveness of the models. The results support the potential for GEMMA-generated data to be utilized in future research and applications in egocentric vision tasks.

References

[1] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6202–6211, 2019. 1

[2] Jiyang Gao et al. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, 2017. 1

[3] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens Van Der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A single model for many visual modalities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16102–16112, 2022. 2

[4] Kristen Grauman et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[5] Kevin Qinghong Lin et al. Egocentric video-language pretraining. *Advances in Neural Information Processing Systems*, 35:7575–7586, 2022. 2

[6] Hao Zhang et al. Span-based localizing network for natural language video localization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. 2, 3