Natural Language Queries for Egocentric Vision

Elisa Sanmartino s348046
Giorgia Lanciotti s329435
Andrea Rinaudo s345885
Lorenzo Spanio s327504

Abstract

The egocentric view captured by head-mounted cameras offers a first person perspective of human activities presenting unique challenges for large-scale annotation. This work addresses the Natural Language Queries (NLQ) benchmark on the large-scale Ego4D corpus, where models must locate temporal segments of long video recordings that answer textual questions in natural language. We evaluate two architectures: VSLBase and its improved variant VSLNet using frozen feature representations from Omnivore, a unified visual transformer and EgoVLP, a video-language contrastive model. An in-depth exploratory analysis characterizes the distributions of query templates, segment durations, and scenario contexts. We also implement an extension for Automatic Query Generation: using a large open source language model (gemma-1.1-2b-it) to synthesize additional question-answer pairs from natural language video narratives, in order to pretrain our model before fine-tuning on real NLQ benchmark data.

1. Introduction

The popularity of wearable cameras has propelled egocentric vision into one of the most interesting fields of computer vision, as they allow for the continuous capture of first-person footage of everyday activities. Such data are essential for applications ranging from assistive technologies to activity monitoring and human-robot interaction. All these factors have prompted the development of large-scale datasets such as Ego4D, with the aim of facilitating research into how machines can understand and reason about human behaviour. In general, the data available for egovision are typically long, uncut videos with few annotations, mainly done manually. This lack of data limits our ability to understand the context and events in these videos. In fact, ideally we would like to ask a free natural language question about a past event or context of the video and receive as an answer the interval that corresponds to the answer of the question. In this way, the system would be able to understand all the events that occur during the entire video. Natural language queries (NLQ) are exactly the field of computer vision that attempts to solve this problem by using a textual query, e.g. "What utensils did the user handle?" or "When was the wallet last viewed?" and the system must directly retrieve the relevant video intervals. The Ego4D NLQ benchmark formalises this as time segment prediction: given a multi-minute video clip and an arbitrary natural language query, the model must predict the start and end times that comprise the answer. Evaluation based on the recall at multiple time intersections (tIoU) measures both boundary accuracy and ranking effectiveness and is used to measure performance at different thresholds.

As mentioned above, the main challenge in training robust NLQ models is the significant cost and effort required to create high-quality annotated datasets. Although the official Ego4D NLQ benchmark is a valuable resource, its approximately 227 hours of annotated videos represent only a small fraction of the approximately 3,600 hours available in the full Ego4D dataset. It contains time-stamped textual narrations where there is a description of their actions. For instance, a narration might state "# C takes puts the pan on the shelf" at timestamp 149.91. These are descriptive statements, not questions and they form a direct link between language and specific moments in time.

This lack of data limits the models ability to learn from a diverse range of scenarios, limiting their ability to generalise to unseen tasks.

In this study, we systematically analysed two leading NLQ architectures: VSLBase, which employs dual-mode encoders and attention to contextual questions and VSLNet, which is an extension of VSLBase and adds a question-driven highlighting mechanism for finer grained alignment. To address data scarcity and improve performance, we implemented an automatic question generation extension: using an open source large language model to generate synthetic question-answer pairs from narrated video segments.

2. Related Work

The analysis of egocentric videos has become increasingly important with the introduction of the Ego4D benchmark, which has made long videos annotated with time-stamped

narratives available and defined the NLQ task aimed at locating time segments in response to questions expressed in natural language. However, manual annotations cover only a fraction (approximately 227 hours out of a total of 3,600), limiting supervised training [1].

To overcome this limitation, the NaQ approach was introduced to automatically convert narrations into synthetic question-answer pairs [2], demonstrating that the use of these pseudo-annotations significantly improves performance on NLQ tasks. Subsequent work has instead experimented with hybrid pipelines where video captions are generated, LLMs are used to identify candidate time windows and responses are refined using specialized models such as VSLNet or ReLER [3].

The NLQ task falls within the broader area of Natural Language Video Localization (NLVL), where a text query is used to identify a relevant time span within a video. In this context, the VSLBase framework was proposed as a span-based QA framework [4], later extended into VSLNet, which introduces the Query-Guided Highlighting (QGH) strategy to focus the model's attention on relevant regions of the video [5].

These architectures have been successfully adapted to Ego4D NLQ, showing excellent results even in long video contexts. For example, GroundNLQ [6] leverages a multi-scale encoder pre-trained on Ego4D narratives, while GazeNLQ [7] integrates gaze information through contrastive pre-training.

In the field of vision-language pretraining, EgoVLP [8] showed how contrastive learning between video and text blocks can improve semantic alignment, boosting performance on tasks such as NLQ and VQ.

In addition to these supervised or semi-supervised approaches, recent work has explored the use of LLMs for the automatic generation of NLQ data. For example, LifelongMemory uses multimodal LLMs to create narrative descriptions from long videos, from which NLQ answers are then extracted in zero-shot, outperforming baselines such as NaQ++ [9].

3. Datasets and backbones

The Ego4D NLQ benchmark comprises of about 19.000 annotated queries spanning 227 hours of egocentric video across diverse scenarios (e.g., cooking, commuting, household tasks). Each sample couples:

- A natural language query, drawn from one of 13 high-level templates (e.g., quantity, spatial relation, activity description);
- temporal annotations at both the clip level *(clip_start_sec, clip_end_sec)* and the full-video level *(video_start_sec, video_end_sec)*, enabling precise evaluation of model localization.

Here's an example of what a time-stamp description is like:"#C C puts the salt shaker back on the shelf.", with time stamp 154.22. And the queries are like this:"How many frying pans can i see on the shelf?" with time interval [start, end].

This Ego4D NLQ benchmark is the "gold standard" in egovision because it allows the NLQ models to understand the actions and link them with the queries and their interval. However, it is just a subset of this dataset: Ego4D, it is significantly larger, 3,600 hrs. This bigger dataset has the same type of videos and the data is structured in the same way as Ego4D NLQ benchmark. It has for all the videos a description of the events with the same time-stamp seen above but it lacks the queries and their intervals. This is a troublesome problem since the formulation of these queries requires watching the entirety of the video, understanding its content and formulating a query in a specific format and assign a precise time interval. Such kind of problems are the reason for which in egovision having sufficient amount of high quality data is paramount and not a negligible problem to deal with. In this paper Ego4D NLQ benchmark will be used for the training part in both VSLBase and VSLNet and also for the extension.

To get more familiar with Ego4D NLQ benchmark here are now presented some plots to understand this dataset. Firstly we analyze the relationship between the length of the queries and the length of the videos. Figure 1 and 2 represents the relative size of the queries, given by the ratio between the size of the queries and the size of the clips, normalized between 0 and 1. We can observe that the majority of the queries have a small size, the distribution is concentrated near 0, this allows us to conclude that the queries have tight intervals compared to the length of the clips, therefore the questions tend to be quite specific and they can be answered in a relatively small interval. There are some cases where some queries have null size, probably due to a mistake during manual annotation (these ones were removed during the training phase).

In figure 3 can be seen how most clips are 480 seconds long, while the minimum and the maximum length are respectively 207 and 1200 seconds.

To enable efficient model training, we employ frozen features extracted by two backbones:

1. Omnivore: A Swin-based vision transformer capturing multi-scale spatial and short-term temporal cues.

2. EgoVLP: A video-language contrastive model pretrained on Ego4D narrations, providing semantically aligned embeddings.

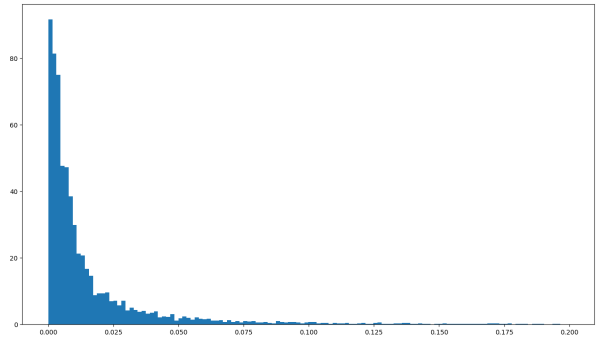


Figure 1. Distribution of the relative size of the queries (< 0.2)

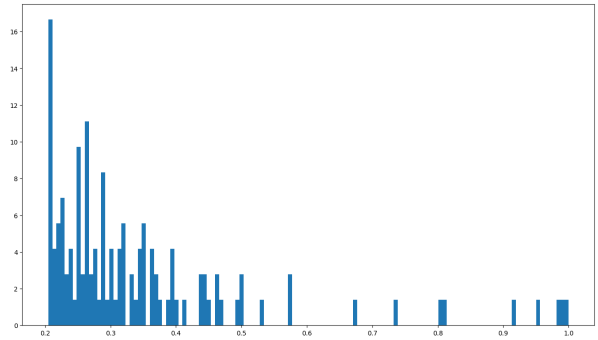


Figure 2. Distribution of the relative size of the queries (> 0.2)

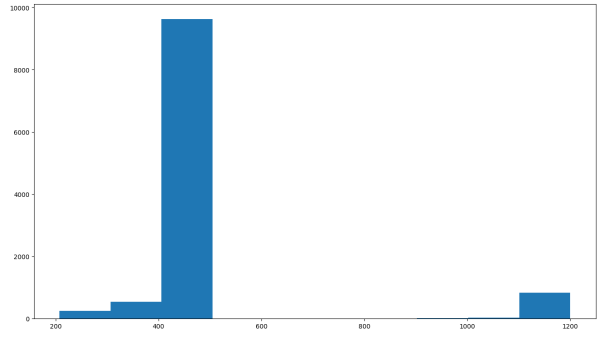


Figure 3. Size of the clips

For the extension part we deal with the annotation scarcity and diversify linguistic expressions by generating synthetic data using for our case gemma-1.1-2b-it, a open source large language model to automatically generate questions from the free-form video narrations. Indeed, given sequences of narrated actions with known timestamps, the LLM produces corresponding queries whose ground-truth answer align with the narrated intervals (see later for more details). This synthetic corpus is then used to pretrain VSLBase and VSLNet before fine-tuning on the original NLQ data.

4. Training

In this paper we propose training two models, VSLBase and VSLNet, using Omnivore and EgoVLP features in order to improve the understanding of linguistic queries in egocentric vision scenarios.
Both models were trained for 10 epochs, with a batch size of 32, a dimension of 128, and an initial learning rate of 0.0025 which decreases linearly to 0 at the end of training.

4.1. VSLBase and VSLNet with Omnivore features

The architectures implemented are based on the formulation proposed in the official GitHub repository for NLQ episodic memory with specific modifications to adapt them to our context.
We adapted and trained both models independently:

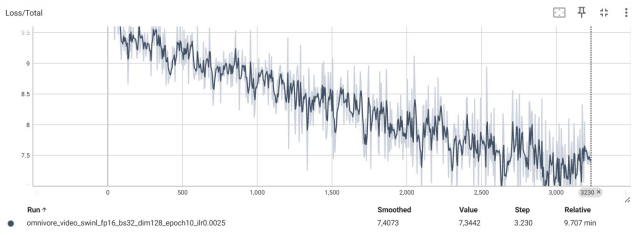
- VSLBase: a simplified version without components for estimating temporal relevance.
- VSLNet: an extended version that includes a specific module `HighlightLayer` that adds an explicit temporal attention mechanism to identify which portions of the video are most relevant with respect to the text query.

The additional components introduced in VSLNet as compared to VSLBase are the following:

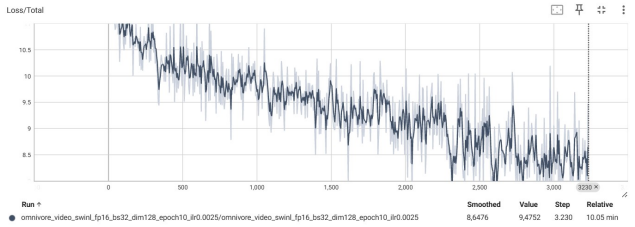
1. Estimation of temporal relevance thanks to the `HighlightLayer` module that produces a vector `h_score` $\in [0,1]^T$, where T is the temporal length of the video. Each element represents the level of importance associated with a specific clip in relation to the query.
2. During the inference phase the combined features video + query are modulated by element-wise multiplication with the `h_score` vector. This operation acts as an adaptive filter, allowing the most relevant information with respect to the query to be highlighted and reducing noise because with this strategy the model is encouraged to focus on frames that are actually relevant.
3. To facilitate the learning of temporally relevant information has been introduced an additional loss function, `compute_highlight_loss(scores, labels, mask)` which provides an explicit supervision in the distinction between relevant and non-relevant frames in the query `h_score`.

4.1.1. Results and comments

We now discuss the main training and validation results obtained with VSLBase and VSLNet. By analysing loss metrics and performance indicators we aim to highlight the strengths and limitations of both models.



(a) VSLBase



(b) VSLNet

Figure 4. Total loss during the traning of VSLBase and VSLNet.

Model	Loss/Total (smoothed)	Value	Relative Time
VSLNet	8.6476	9.4752	10.05 min
VSLBase	7.4073	7.3442	9.707 min

Table 1. Total loss comparison between VSLNet and VSLBase.

IoU Threshold	Model	R@1 (%)	R@5 (%)
0.3	VSLNet (reported)	5.45	10.74
	VSLNet (reproduced)	6.43	12.67
	VSLBase (reproduced)	5.50	12.57
0.5	VSLNet (reported)	3.12	6.63
	VSLNet (reproduced)	3.74	8.03
	VSLBase (reproduced)	3.33	7.98

Table 2. Comparison between the reproduced versions of VSLBase and VSLNet performance on the Ego4D NLQ validation set using Omnivore features, compared to the official baseline results provided in the Ego4D paper and trained on SlowFast features.

The comparison between `Loss/Total` in figure 4 and Table 1 shows that VSLBase is more effective in minimizing overall loss. This does not necessarily translate into better generalization; in fact the performance metrics shown in Table 2 reveal that VSLBase tends to overfit the training data, resulting in lower accuracy on the validation set than VSLNet.

During the training, TensorBoard produces different loss curves that reflect how the model optimizes the different components of its objective function. In VSLNet these visualizations reveal a richer supervision mechanism compared to VSLBase; in particular VSLNet introduces the following additional loss terms:

- `Loss/Loc`: penalizes discrepancies between the predicted temporal segment and the ground-truth annotated interval.
  Values: Smoothed: 7.7139, Value: 8.3654
- `Loss/highlight`: encourages the model to generate a temporal attention map that emphasizes the video segments most relevant to the query.
  Values: Smoothed: 0.1867, Value: 0.222
- `Loss/highlight(*λ)`: corresponds to the highlight loss scaled by a weighting factor λ which reflects its contribution to the total loss.
  Values: Smoothed: 0.9337, Value: 1.1098

These additional results reflect the fact that the objective of both models is to find the time interval in the video that answers the question, but VSLNet tries to figure out which parts of the video are most important in arriving at that answer.

So VSLNet is able to represent better the link between the question and the video, but because it has more parameters and the model is more complex, it is also slower to train.

One last thing to note is the learning rate. Both models use the same scheduling: the learning rate starts at an initial value of 0.0025 and decreases linearly to 0 over the course of the training. Since the scheduling and the initial value of the learning rate are identical for VSLBase and VSLNet, we can confirm that the observed differences in the final performance of the models are attributed exclusively to the architecture of the two models.

4.2. VSLNet variants

The VSLNet model was also trained using EgoVLP features instead of Omnivore and the GloVe encoder instead of BERT. Analyzing the data reported in table 2, 3 and 4, several key insights emerge regarding the impact of linguistic encoders, visual feature extractors and model architectures on performance.

IoU Threshold	R@1 (%)	R@5 (%)
0.3	8.57	16.78
0.5	5.16	11.20

Table 3. VSLNet performance with EgoVLP features

- **Linguistic Encoder:** The use of a contextual model such as BERT yields a substantial gain over static

IoU Threshold	R@1 (%)	R@5 (%)
0.3	3.15	8.03
0.5	1.70	4.59

Table 4. VSLNet performance with GloVe encoder

GloVe embeddings. Egocentric queries often contain fine-grained temporal and semantic cues, which GloVe cannot disambiguate due to its fixed word representations. BERT's context-dependent embeddings, pretrained on large text corpora, enable more precise alignment between the user's question and the video features, leading to noticeably higher grounding accuracy.

- **Visual Feature Extractor:** SlowFast, while adept at modeling motion dynamics in third-person videos, underperforms in first-person scenarios: it lacks specialization for continuous close-up views of hands and objects characteristic of egocentric footage. Omnivore introduces a moderate improvement through general-purpose multimodal training (images, video, text) but remains suboptimal for egocentric nuances. In contrast, EgoVLP,pretrained on paired egocentric video and text, delivers the most discriminative and semantically grounded features, significantly boosting the model's ability to localize the correct temporal segments.

- **Model Architecture and Overfitting Dynamics:** VSLBase achieves a lower training-set loss compared to VSLNet, indicating faster convergence on the training distribution. However, its lower IoU on the validation set suggests overfitting: VSLBase appears to memorize training-specific patterns at the expense of generalization. Conversely, VSLNet, despite slightly higher training loss, maintains stronger robustness, yielding superior IoU in validation and demonstrating a better trade-off between fitting the training data and accurately predicting unseen segments.

5. Extension: Automatic Queries Generation using LLMs

5.1. Purpose of the extension

In this extension we address the data scarcity problem by employing gemma-1.1-2b-it, a Large Language Model (LLM) as a query synthesizer, to automatically generate a large-scale, synthetic NLQ dataset using the videos from Ego4D dataset. This allows to pre-training a temporal localization model on this augmented dataset, before fine-tuning it on the official benchmark data. This allows the model to be more robust and capable of generalize and understand the relationship between language and visual events, leading to improved performance. Each of the following sections is a step necessary to create the extension

5.2. Creating the synthetic dataset

We want to identify all the unique video_uids that appear in the NLQ benchmark files. We then decides to only consider narrations from these specific videos. Because in this project we have only downloaded the pre-computed video features for these benchmark videos. It would be useless to generate a question for a video segment for which we have no features to train on. So here is what we practically do: The script iterates through nlq_train.json, nlq_val.json, and nlq_test_unannotated.json again. For every video, it collects the start and end times (video_start_sec, video_end_sec) of every single "clip" defined in the benchmark. This creates a dictionary where each video UID maps to a list of time intervals that are already in use and we can consider this a "black zone" that we want to stay away, so that we actually produce queries from narrations that are not in these zones. A single video might have multiple clips in the benchmark that are close together or even overlap. For example, a video might have a clip from [10, 50] seconds and another from [40, 90] seconds. To simplify the filtering, the script uses a merge_intervals function, it takes the list of forbidden zones for a video and merges any that overlap. However despite this attempt to link the intervals there still might be discontinuities between the intervals. This is in general not a problem because the the whole point of using LLM is to generate a synthetic data that is a more diverse and "imprecise" and must handle narrations that "jumps" from a good interval to the next one and skipping the events present in the "forbidden zone".

We actually don't need to utilize fully this just created dataset. In fact, the number of narrations present are a huge amount and for our purposes, we just a tiny fraction of it for the pretraining needed in this extension. We decided therefore to sample randomly 1300 narrations form this dataset. This hyperparameter is generally very very small for these kinds of training, a more correct value would be 10000, however due to the very strict limitations in the computing resource we had to adapt to such a small parameter.

5.3. Length of the context for the LLM

In order to generate a sensible query it is not enough to just look at the single narration, because it would lack the context to formulate a precise question and therefore not using fully the information that the query can give to the training process. On the other hand, when we select a video we don't actually use all of its narrations at once because a video, in general, can be really long and therefore the size of the narrations can exceed the context window of a LLM. As a consequence, there must be a trade-off to be considered. Here's what we practically do: in a single video we will have a single sliding window that will take on the first

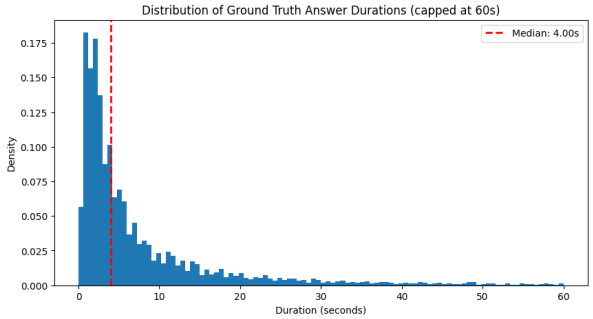


Figure 5. A histogram shows the distribution of answer durations (of the training set of NLQ benchmark). The red dashed line marks the median. The plot is capped at 60 seconds to improve readability by ignoring very long answers.

N narrations, provide for each one the queries and move on to the next N narrations (there is no overlapping between windows) and we repeat the same pattern.

Having excluded the extremes of N, now we go into the motivation for the choice of this hyperparameter. If we choose the N too big the LLM will have a very good context window and will be able to capture the themes of the video very well providing on point queries for each narrations. However, a few queries will contain information about the entire video and will make it harder to the VSLnet model to exactly pinpoint the start and end of the answer to the question. More often than not it will provide as the segment the entirety of the video, not making it very useful, due to the fact that the question can be very general and across narrations. On the other hand choosing N too small will result in a query provided by the LLM to be more specific but it will not understand the overall task. In general a smaller N is preferred due to the desire to connect the visual with the textual modalities. In the next section there is an analytical way to calculate a reasonable N.

5.4. Choosing the length of the context for the LLM using data analysis

The actual choice of N is, as discussed before, hard in practice, in general N=1-2 is too small and N over 50 is too big of a context. In order not to guess randomly we performed an analysis on the data, we want to choose N such that the total time spanned by that block of narrations is similar to the typical duration of an answer in the official NLQ benchmark training dataset. This helps ensure the model is trained on data that mirrors the characteristics of the data it will be tested on. So, for each query, we calculate the duration of the correct answer segment (video_end_sec - video_start_sec) and calculate the median and mean of this distribution, as we can see in figure 5.

We then calculate in the unused dataset the temporal time between the narrations (for all the videos). This tells us the typical time covered by a single ”step” in our narration sequences. During this loop we apply a filter, since we want to consider small, continuous time gaps, representing a fluid sequence of actions. We ignores very large gaps, which might occur if a large segment of the video was filtered out because it was part of the NLQ benchmark. So we calculate the median of the step between narrations. Now we are ready to calculate the number of narrations we will give to the LLM in a single prompt: we divide the median length of the ground truth answers by the length of the median step between narrations, we then add 1 (because so far we calculated the gaps between narrations and instead we need the number of narrations), we then round the number and in our case we obtain N = 3. This value of this hyperparameter will be used for all the pretraining part of this extension.

Having done this approach we must consider that the average ground truth interval will be bigger than the ones calculated with this method, because for each block there will be many queries and so each one will be a subset of the block interval (an approximation of the average ground truth interval). This was a deliberate choice because we are creating a dataset for the pretraining and we want the model to capture specif events-descriptions.

5.5. Choice of the LLM

We decided to use the open-source model 2 billion parameter Gemma model (gemma-1.1-2b-it). It is a rather small model but is actually enough for the task required for this specific project. Indeed, we are required to generate simple, direct questions based on short, descriptive narrations of actions with just a few narrations for each prompt (N=3 as stated before). This allows to save space and speed up the process of generation all the queries. Initially we used llama 3 8B parameters but due to the simplicity of the task required, the cost of running this heavier model didn’t outweigh the benefits so we switched to the aforementioned model. We used Hugging Face to access the Gemma model.

5.6. Prompt engineering

The choice of the prompt had to be carefully chosen to generate the synthetic queries. Before delving into the content of the prompt given to the LLM we ran a few tests and we encountered some difficulties generating the data in the corrected format, even with a very specific prompt. This is probably due to the fact that LLM are not used to receive prompts like the ones we are providing , that is, narrations that are structured like this: ”#C C opens the drawer”. The #C at the beginning causes the most trouble, so we decided to eliminate the initial part, each of the narrations now like this:”C opens the drawer”. Then these ones are fed into the LLM.

Here’s a summary of the prompt given to the LLM: we provide the overall task that it must complete. Given a se-

quence of narrations we want to generate queries for each one of the narrations (and exactly one). We also specify the structure the output has to have. We also specify that the output must not contain information other than the ones required.

What the prompt does not do is to require the queries to follow the standard template of the Ego4D NLQ benchmark, this is done on purpose since we want these synthetic data to be able to teach the VSLNet model to be more general and robust. Capable of understanding better the relationship between human language and video actions. Therefore, we want the model to learn to answer questions about videos in general, not just questions that fit a narrow set of predefined templates. This freeness of the queries also limits the hallucinations of the LLM, imposing a strict templates can lead to nonsensical queries (e.g., "How many bottles of water are there?" to a narration such as:"#C C takes a bottle of water").

5.7. Hyperparameters for LLM and GPU

Once we set up the prompt we proceeded with the actual generation of the queries. The limited resources available (we used a T4 GPU) and the LLM chosen (gemma-1.1-2b-it) allowed us to used BATCH_SIZE = 16 and temperature = 0.3 for the LLM. The choice of the batch size is standard for a single GPU, instead the choice of the temperature is based on the required task. Our primary goal is to generate queries that are factually grounded in the narration, we don't want the model to get creative and ask about things that aren't happening. We need it to follow the prompt's rules precisely. So a lower choice of the temperature seems reasonable (the "standard" temperature is 0.6). As a secondary control we put top_p=0.9, it tells the model to completely ignore the lowest-probability words. This way we maintain just enough linguistic diversity to make the pretraining dataset richer and robust while producing sound queries.

5.8. Generation of the queries with LLM

We use the prompt to generated a block of queries, since they imposed that they had to be indexed at the beginning (to link the narrations and queries with a 1 to 1 relationship), we now want to remove these numbers to have more natural queries. The output of this whole process is a list of dictionaries. But before having this new dataset we had to deal with numerous problems after the generation process.

Having the synthetic queries now we had to specify the interval corresponding for each one. But this sparked a problem: even if we consider the temporal window of a query to be the interval between two consecutive narrations, the last narration cannot access the next timestamp. A simple solution would be to assign for all the narration in the block the same interval, that starts at the beginning of the first narration and ends with the timestamp of the last one. This approach may be reasonable in our case because the number of narrations are very limited for each prompt. However, this implies that every narration in the same block has the same interval, therefore the interval of a question is surely much bigger than what it should be. This inevitably leads to worse performance of the pretraining part due to the fact that the data is not precise enough with the interval. To deal with this issue we decided to use an analytical approach: the window for each query is the distance between narrations. The interval of the query corresponding to the last narration is the average length of the past intervals. This is a somewhat acceptable approximation because the narrations are already put as a sequence of actions, so it is reasonable to assume that the last action will have a duration similar to the previous ones. A single synthetic data contains all the information regarding the video that was taken from: the start and end of the video, the synthetic query, the start and end end of this query, as well as the end time of the entire block of narrations processed together in the prompt. This is due to the fact that later the narrations processed together will be considered part of the same clip (limited by the starting timestamp of the first narration and the approximated end of the last query).

5.9. Transforming the synthetic data to the correct format

Once we generated the synthetic data what we have is a list of annotations (dictionaries). These data must be transformed in such a way that the VSLNet is able to read them correctly. This means transforming these data in the same json format (nested dictionaries) of the original NLQ benchmark ,otherwise the pretraining process can not happen. We use the "inside-out" approach for building nested structures, starting with the most deeply nested items and gradually wrap them in their parent containers until we reach the top level. So, we begin by grouping all the queries that belong to the same video, and within that video we group all queries that were generated from the same "block" of narrations. Once we obtain that we build the final correct structure by considering all the narrations of the same block as a single clip. These clips are then put inside the corresponding videos_uid. Since we are dealing with new clips we have to assign to each one of them a unique identifier, we used a progressive index starting from 0 (the very same idea must be applied for the queries inside each clip, query_uid).

5.10. Pretraining and hyperparameters

In this paper [8] the authors provided a Python script to preprocess the data. Its job is to take the raw, human-friendly data (nlq_synthetic_train.json file and the video feature files) and transform it into a highly optimized, machine-ready format that the VSLNet model can consume efficiently during training. So it performs all the slow, heavy, and repetitive

preparation work once, upfront. The goal of pre-training on this synthetic dataset is obviously not to get a perfect, final model. Instead, the goal is to teach the model the general patterns and the basic structure of the Natural Language Query task. Specifically, it should get familiar that words like ”open,” ”put,” ”take,” or ”close” correspond to specific visual patterns in the Omnivore feature vectors. As well as getting used to taking a text query and a video as input and predicting a start and end time as output. We used as a hyperparameter 5 epochs, this is in general a low value for the pre-training, however for our case we had to choose such a value because the number of generated queries are very limited and with an aggressive learning rate (see later) and numerous epochs we found that the model overfits the synthetic dataset and as result the finetuned model performs way worse, so a lower number was preferred. The number of question-video pairs the model processes at once before updating its weights is 32, it generally is a standard batch size. For the pretraining we used a relatively high learning rate 0.0025, because the model starts from a random state and we want it to learn quickly and make large adjustments to its weights to get out of that random state and into a ”sensible” region of the solution space. Before proceeding to the fine-tuning training part we save the weights just obtained and the AdamW parameters to maintain the training momentum observed so far.

5.11. Fine-tuning

The pretrained model helps understand the general relationship between language and video but might be biased by the LLM-generated synthetic data. The goal of fine-tuning is to adapt and specialize this general knowledge to the specific style, vocabulary, and characteristics of the official Ego4D NLQ benchmark. We are now switching from the synthetic dataset to the high-quality official benchmark data so the model will now learn from the human-annotated ground truth. Of course for this part the model does not start with a randomly initialized weights. Instead, it loads the weights and the AdamW optimizer state from the pretrained checkpoint obtained just before. The learning rate is the same as seen before for the training of the previous models, 0.0025. In general a smaller learning rate would allow the model to make small, careful adjustments to its weights, gently refining the pretrained knowledge without moving in directions where the loss is worse, however our main purpose it to confront the standard VSLnet model with the fine-tuned one. The number of epochs was set to 10, this choice was based on the fact that it would be unfair to compare the two models with the same learning rate but different epochs. In a real setting we would push this value to 15-20.

5.12. Testing the final model

We compare the results of the pretrained model obtained with the VSLnet trained in the previous sections. As stated before our goal was not to create the best performing model but to prove the effectiveness of the pretraining using synthetic data. As we can see in table 5 the pretrained model provides better results in almost every categories. We should note here that the results are only slightly better than the other model because the pre-training had at hand just 1300 synthetic data generated, this put a stict limitations on the possible improvements. Moreover this lack of synthetic data was a factor in the number of epochs usable. We found that even on just 5 epochs the pretraining part tends to overfit and its loss is rather low. All these problems would be solved (at least partially if we had more computing time or multiple GPUs).

Table 5. Model Performance Comparison between VSLnet and VSLnet pre-trained

IoU Threshold	Model	R@1 (%)	R@5 (%)
0.3	VSLNet	6.43	12.67
	VSLnet pre-trained	6.48	13.53
0.5	VSLNet	3.74	8.03
	VSLNet pre-trained	3.51	8.29

6. Conclusion

This work addresses the Natural Language Queries (NLQ) benchmark on the large-scale Ego4D corpus, where models must localize temporal segments of long video recordings that answer free-form textual questions. Firstly, we explored the dataset at hand and tested existing models such as VSLBase and VSLNet. Then we compared variants of VSLNet: one with features of EgoVLP and one with GloVe encoder. We then tackled the challenge of lack of training data by creating a synthetic dataset to pretrain the VSLNet model. To do so we used a LLM (Gemma) to turn simple video descriptions into new question-answer pairs. Our experiments showed that pretraining our model on this new, synthetic data before training it on the official data made it more accurate. However, there are limitations to this method: the computing power is a very limiting factor to the scalability of the improvements. The generation of synthetic data should be much more extensive than the one carried out in this project.

References

[1] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari *et al.*, “Ego4d: Around the world in 3,000 hours of egocentric video,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[2] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, "Naq: Leveraging narrations as queries to supervise episodic memory," *arXiv:2301.00746, [Online]. Available: https://arxiv.org/abs/2301.00746*, 2023. 2

[3] Y. Wang, D. Sun, R. Chen, Y. Yang, and M. Ren, "Egocentric video comprehension via large language model inner speech," *Proceedings of the 3rd International Ego4D Workshop*, 2023. 2

[4] H. Zhang, A. Sun, W. Jing, L. Zhen *et al.*, "Natural language video localization: A revisit in span-based question answering framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Feb 2021. 2

[5] H. Zhang, A. Sun, W. Jing, and J. T. Zhou, "Span-based localizing network for natural language video localization," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. 2

[6] Z. Hou, L. Ji, D. Gao, W. Zhong, K. Yan, C. Li *et al.*, "Groundnlq @ ego4d natural language queries challenge 2023," *arXiv:2306.15255, [Online]. Available: https://arxiv.org/abs/2306.15255*, Jun 2023, challenge report, CVPR-Ego4D Workshop. 2

[7] W. C. Lin, C. M. Lien, C. Lo, and C. H. Yeh, "Gazenlq @ ego4d natural language queries challenge 2025," Jun 2025. 2

[8] K. Q. Lin, A. J. Wang, M. Soldan, M. Wray, R. Yan, E. Z. Xu, D. Gao, R. Tu, W. Zhao, W. Kong, C. Cai, H. Wang, D. Damen, B. Ghanem, W. Liu, and M. Z. Shou, "Egocentric video-language pretraining @ ego4d challenge 2022," Jul 2022. 2, 7

[9] Y. Wang, Y. Yang, and M. Ren, "Lifelongmemory: Leveraging large language models for answering queries in long-form egocentric videos," *arXiv:2312.05269, [Online]. Available: https://arxiv.org/abs/2312.05269*, Dec 2023. 2