ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Egocentric Video Understanding using Vision-Language Models

Master's Thesis

Arda Arslan
Department of Computer Science

Supervisor: Dr. Xi Wang
Supervising Professors: Prof. Dr. Luc van Gool, Prof. Dr. Otmar Hilliges

January 9, 2024

Abstract

Egocentric videos are videos recorded from the first-person point of view. These videos can be recorded by a camera mounted on the head of a person, or cameras on smart glasses [12]. Ego4D [14] is an egocentric video dataset introduced by Facebook AI Research (FAIR). Ego4D [14] Moment Queries Challenge (Ego4D-MQC) is a challenge that consists of two sub-tasks: action detection and action retrieval. Action detection is detecting segments of a given egocentric video that includes an action instance corresponding to one of the 111 pre-defined action categories. Action retrieval is the task of retrieving segments of a given egocentric video that includes an action instance corresponding to only the input action category.

In this work, in addition to working on action detection and action retrieval tasks, we also work on frame-wise action category classification task which is classifying each frame of a given egocentric video into one or multiple of 111 pre-defined action categories. In our experiments, we feed frames of a given video clip to BLIP-2 [20], and we use the caption generated by BLIP-2 [20], SBERT [28] embedding of this caption, and the output of BLIP-2's [20] large language model encoder.

The code for our work is available at https://github.com/ardarslan/egocentric-video-understanding.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr. Wang for her patience and guidance throughout this journey, and also to Prof. Dr. Luc van Gool and Prof. Dr. Otmar Hilliges for accepting me as a Master's Thesis student. I am also extremely thankful to my professors at ETH Zürich for their exceptional teaching. Finally, I would also like to express my greatest appreciation to my family and friends for their immense support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Egocentric videos are videos recorded from the first-person perception. These videos can be recorded by a camera mounted on the head of a person, or cameras on smart glasses [12]. Ego4D [14] is an egocentric video dataset introduced by Facebook AI Research (FAIR). Ego4D [14] Moment Queries Challenge (Ego4D-MQC) is a challenge that consists of two sub-tasks: action detection which is detecting segments of a given egocentric video that includes an action instance corresponding to one of the 111 pre-defined action categories, and action retrieval which is retrieving segments of a given egocentric video that includes an action instance corresponding to only the input action category.

Similar to the previous works that tackled one of the Ego4D [14] challenges ([25], [16]), we also make use of a pre-trained vision-language model in our experiments. To the best of our knowledge, our work is the first work that uses a pre-trained vision-language model for Ego4D-MQC.

1.1 Main Components of Our Work

Our work consists of the following main components:

1. **BLIP-2 [20] Prompt and Post-Processing Method Ablation:** We experiment with three prompts and two post-processing methods to check which prompt and post-processing pair generates the most informative caption. We use without-backbone evaluation metrics to compare the informativeness of these pairs.
2. **BLIP-2 [20] Caption - Action Category Match Success Evaluation:** For each video's each frame's each BLIP-2 caption, we check whether we have a match between this BLIP-2 [20] caption's (verb, noun, tool) pairs and (verb, noun, tool) pairs of this frame's ground truth action categories. We report the frequency of frames whose caption is matched to one of the ground truth action categories of this frame.
3. **Comparison of Our Predictions with ASL-Ego4D [30] Predictions:** We compare our predictions and the predictions of the state-of-the-art method (ASL-Ego4D [30]) in two ways:
    - **Action Detection and Retrieval:** We predict the start boundary, the end boundary, and the action category of each segment that corresponds to an action instance using both the state-of-the-art approach and our approach. We compare these approaches using with-backbone evaluation metrics.
    - **Frame-wise Action Category Classification:** We predict the action categories of each frame again using both the state-of-the-art approach and our approach. We compare these approaches using without-backbone evaluation metrics.

4. **Vision-Language Model Fine-Tuning Script Implementation:** We propose a script that fine-tunes a given vision-language model on a given Visual Question Answering (VQA) dataset which could be used for making the BLIP-2 [20] embeddings more informative for the Ego4D-MQC.

5. **Egocentric Video Visualization Tool Modification:** We modify an existing egocentric video visualization tool so that it could be used in our task.

6. **New Visualization Tool Implementation:** Finally, we propose a new visualization tool that allows the user to visualize the dependency parsing features, predicted action categories, ground truth action categories, and the correctness of the predictions in a per-frame setting.

1.2 Thesis Organization

- In chapter 2, we provide a summary of three of the state-of-the-art approaches of the Ego4D-MQC ([4], [31], [26]). The approaches we reviewed have a ranking in the first four places of the Ego4D-MQC, and we picked the ones with a published paper. Then, we provide a summary of two approaches ([25], [16]) that made use of pre-trained vision-language models for one of the Ego4D [14] challenges. Finally, we provide a summary of two pre-trained architectures we use in this work.

- In chapter 3, we provide an analysis on the annotations of our data, and we describe the evaluation metrics we use.

- In chapter 4, we describe each component of our work.

- In chapter 5, we provide the results of our experiments.

- In chapter 6, we comment on the results of our experiments, and provide the conclusions of our work.

- In chapter 7, we provide the possible directions for the future work.

Chapter 2

Related Work

2.1 State-of-the-Art Approaches in the Ego4D-MQC

1. **InternVideo-Ego4D: A Pack of Champion Solutions to Ego4D [14] Challenges [4]**

   For the Ego4D-MQC, the authors make use of architectures that were pre-trained on human action video datasets. They state that there is a domain gap between the videos these architectures were pre-trained on and the videos used in the Ego4D-MQC. Therefore, the authors fine-tune these architectures. They do two types of fine-tuning:

   - **One-Stage Fine-Tuning:** The authors fine-tune the pre-trained architectures listed below on the training split of the dataset used for the Ego4D-MQC.
     - ir-CSN-152 [33] is a convolutional neural network which was pre-trained on IG-Kinetics-65M dataset [11]. This dataset consists of over 65 million public videos from a social media website and their 359 unique action categories.
     - VideoMAE v1 [32] (ViT-L) is a Vision Transformer network [9] that was pre-trained on Kinetics 700 dataset [19]. This dataset consists of approximately 650K video clips and their 700 unique action categories.
     - VideoMAE v1 [32] (ViT-L, MVF) is the spatial-level Multi-View Fusion version of the VideoMAE v1 (ViT-L) (definition taken from [4]).

     In one-stage fine-tuning experiments, they experiment with only the Video Self-Stitching Graph Network (VSGN) architecture [40] without its Video Self-Stitching (VSS) component. VSGN [40] without VSS simply takes a video as input, extracts features from it using a SlowFast network [10], and uses the extracted features as input to a graph pyramid network. The authors report that using the features extracted using VideoMAE v1 [32] (ViT-L, MVF) performs better than the other two, however, it has a higher computational cost.
   - **Two-Stage Fine-Tuning:** All videos in the Ego4D dataset [14] have one or multiple noun and verb annotations. The authors create a new dataset (Ego4D-Verb) in the following setting: they iterate over the videos in the training split of the Ego4D dataset [14]. For each video, they iterate over its verb annotations. For each verb annotation of this video, they add the tuple (the current video and its current verb annotation) as a new sample to this new dataset. This dataset is used to fine-tune the pre-trained models so that the features they extract will be closer to the Ego4D [14] domain. In two-stage fine-tuning experiments, the authors compare VSGN [40] without VSS and ActionFormer [39] backbones. They report that using ActionFormer [39] instead of VSGN [40] without VSS as the backbone yields better results. They also report that fine-tuning the

ActionFormer [39] backbone on first Ego4D-Verb and then Ego4D-MQC yields better results than fine-tuning only on Ego4D-Verb.

[30] shows that using ASL-Ego4D [30] backbone instead of ActionFormer [39] backbone yields better results for Ego4D-MQC. Therefore, we use ASL-Ego4D [30] backbone in our experiments.

At the time of writing, this approach ranked 4th place on the leaderboard of the Ego4D-MQC.

2. **EgoVLPv2: Egocentric Video-Language Pre-training with Fusion in the Backbone [26]**

   EgoVLPv2 [26] uses TimeSformer [2] to encode the input video and RoBERTa [22] to encode the input text. To capture cross-modality interaction between the inputs, they use cross-modal fusion in the top layers of two backbones.

   This architecture is pre-trained on EgoClip [21], which is a dataset that consists of training videos of all Ego4D-MQC. The dataset includes narrations of these videos as well.

   To pre-train this architecture, they use the convex sum of three loss functions:

   - **EgoNCE [21]**: This is a contrastive loss. A video v_i and its corresponding text t_i are considered a positive pair. In addition to this, a video v_i and a text t_j (belonging to video v_j) are considered a positive pair if t_j and t_i have at least one matching verb or noun. A video v_i and a randomly sampled text t_j are considered as negative pair. Finally, for each video v_i, they sample a temporally close video $v_{\hat{i}}$. v_i and $t_{\hat{i}}$ (belonging to $v_{\hat{i}}$) are considered a negative sample. EgoNCE [21] tries to maximize the similarity of embeddings of positive pairs and minimize the similarity of embeddings of negative pairs.
   - **MLM:** This is masked language modeling loss. They mask 15% of text tokens. Then they reconstruct these tokens by using the patches of the corresponding video and unmasked tokens of this text.
   - **VTM:** This is video-text matching loss. The model predicts the probability of a given input pair (video, text) being a match or not. A video v_i and its corresponding text t_i are considered positive samples. The negative samples are selected by using the EgoNCE [21] loss.

   For the Ego4D-MQC, the authors extract features from pre-trained EgoVLPv2 [26] and feed these features to a VSGN [40] to train it.

   At the time of writing, this approach ranked 3rd place on the leaderboard of the Ego4D-MQC.

3. **Action Sensitivity Learning for the Ego4D Episodic Memory Challenge 2023 (ASL-Ego4D) [31]**

   The authors propose an anchor-free architecture that predicts the action categories and distances to the start and the end of the action instances in a single stage and in a per-frame level. This architecture can be examined in five main components:

   - **Video Feature Extractor:** This is a pre-trained 3D convolutional neural network.
   - **Feature Encoder:** This consists of a Transformer encoder and a feature pyramid network. The transformer encoder is used to enhance the features. It applies two types of attention mechanisms in parallel: one in the temporal dimension and the other one in the channel dimension. The outputs of these two attention operations are fused to get the video encoding $\in \mathbb{R}^{TxD}$ where T is the number of frames in the video and D is the embedding dimension.
   - **Action Sensitivity Evaluator:** The authors state that both while predicting the start and the end boundaries of a given video segment, and while assigning an action category to it, the

contribution of each frame is different. Therefore, they suggest assigning two action sensitivity values to each frame (one for each sub-task).

If a frame is assigned a high value of action sensitivity for a sub-task, this means this frame is important for this sub-task.

For each sub-task, and each frame taken from a ground truth action instance, a category-level action sensitivity value and an instance-level action sensitivity value are calculated. The final action sensitivity value is calculated by summing the instance-level and category-level action sensitivity values.

- **Action Sensitive Contrastive Loss:** The authors use this loss to enhance the video features extracted by the feature encoder.
- **Two Sub-Task Heads:** The features extracted using the feature encoder and their sensitivity values are used by a localization head to predict the start and end times of action instances, and they are used by a classification head to predict the action category of the predicted video segment.

To train this backbone they use features extracted from pre-trained architectures. These architectures are video encoder of EgoVLP [21], InternVideo-Ego4D (VideoMAE v1 (ViT-L)) [4], SlowFast [10], and video head of Omnivore [13]. They reduce the dimension of each feature by training a separate fully connected neural network for each feature.

At the time of writing, this approach ranked 1st place on the leaderboard of the Ego4D-MQC.

2.2 Previous Work which use Pre-Trained Vision-Language Models for an Ego4D Challenge

1. **Summarize the Past to Predict the Future: Natural Language Descriptions of Context Boost Multimodal Object Interaction Anticipation [25]**

   This work tackles the object interaction anticipation task on the Ego4D [14] dataset. The authors define this problem formally in the following setting: Given $\mathbf{T}$ frames as input, the task is to predict the future object interactions in the $\mathbf{T}$ + (FPS of the video * $\mathbf{\Delta}$)th frame (the activation frame). Since the activation frame is not in the inputs, the object interactions can be shown in the last available frame in the inputs (the $\mathbf{T}$th frame, the prediction frame). Since there might be multiple object interactions, each object interaction can be denoted by an index i. The ith object interaction that will occur in the activation frame, but will be shown in the prediction frame can be represented as $\mathbf{o_i^T} = (\mathbf{b_i^T}, \mathbf{n_i^T}, \mathbf{v_i^T}, \mathbf{t_i^T})$ where $\mathbf{b_i^T}$ is the bounding box of the object, $\mathbf{n_i^T}$ is the semantic object label of the object (noun), $\mathbf{v_i^T}$ is the action performed on the object (verb), $\mathbf{t_i^T}$ is equal to the time to contact the object ($\mathbf{\Delta}$).

   The authors extract three types of features from each input frame:

   - **Action description:** The authors use an image captioning model called OFA [36] to extract multiple full-sentence descriptions per frame. They input each frame to OFA [36], together with the following questions: "What does the image describe?", "What is the person in this picture doing?", and "What is happening in this picture?". Each query returns a full-sentence caption related to the frame. They use part-of-speech tagging on each caption using Flair [1] to find (verb, noun) pairs. Then using NLTK [3], they lemmatize each caption. They find instances of lemmatized verbs followed by lemmatized nouns with a distance of at most 4. They call the most frequent (verb, noun) pair in the captions generated for the given frame the "action description".

- **Held objects:** The authors use a hand-object interaction predictor [6] jointly with an object detector [41] to identify objects that were held by the actor in the given input frame. The hand-object interaction predictor alone returns labelless bounding boxes of the active objects in the given frame. The object detector returns bounding boxes with their labels. If a bounding box returned from the hand-object interaction predictor and a bounding box returned from the object detector has an intersection over union (IoU) larger than 0.25, then the bounding box returned from the hand-object interaction predictor is labeled with the label of the bounding box returned from the object detector.
- **Salient objects:** The authors state that the next object that the actor will interact with, might already be seen in the input frames. Therefore, they use CLIP [27] on each frame to detect the objects, and they keep 5 objects with the highest scores.

They have at most one action feature (a verb-noun pair), several objects that were held by the actor, and several salient objects that were seen for each frame. To compress this representation even further, the authors use cross-frame aggregation: They simply aggregate the frame-wise features in the temporal dimension. For each feature type $\mathbf{c}$, the aggregation is done independently and in a temporal setting.

With the help of cross-frame aggregation, the action context consists of only the dominant actions and objects in the input frames, it is less noisy, and it can be represented by short textual descriptions. They feed different combinations of these three action context representations (action descriptions, held objects, salient objects) as input to the model.

The authors use aggregated summaries as the language input, and they use the prediction frame as the image input to a neural network.

At the time the authors published their paper, their approach ranked 1st place on the leaderboard of the Ego4D [14] object interaction anticipation task.

The authors of BLIP-2 [20] show that BLIP-2 [20] performs better than OFA [36] on visual question-answering tasks. Therefore, in our work we use BLIP-2 [20] instead of OFA [36] as the pre-trained visual-language model.

Using part-of-speech tagging to find verbs in a given caption, and then searching for nouns that are within a distance of at most 4 to each verb have the following four problems:

- We can accidentally match a noun to a verb although this noun is not the object being affected by this verb. For example, for the sentence “The person is using a towel to clean the table.”, the word “towel” and the word “clean” are within a distance of 2, which is less than 4. Therefore, (“clean”, “towel”) is one of the extracted (verb, noun) pairs for this sentence.
- Some verbs consist of multiple words. For example, for the sentence “The person is trying out accessories.”, the extracted verb should be “try out”. However, the algorithm used in TransFusion [25] extracts “try” as the verb in this caption.
- Some nouns consist of multiple words. For example, for the sentence “The person is using a vacuum cleaner”, the algorithm used in TransFusion [25] extracts “vacuum” and “cleaner” as separate nouns of this caption, however it should have extract “vacuum cleaner” as the noun phrase being affected by the verb.
- Some action categories in Ego4D-MQC have similar (verb, noun) pairs, and differ by the tool. For example, a frame whose ground truth action category is “cleaning floor with broom” and a frame whose ground truth action category is “use a vacuum cleaner to clean” can have captions that yield the same (verb, noun) pair, which is (“clean”, “floor”). However, in the first one, a “broom” is being used as the tool, and in the second one, a “vacuum cleaner” is being used as

the tool. The algorithm used in TransFusion [25] does not extract the tool information from captions.

In our work, we propose a dependency parsing algorithm that handles all the problems mentioned above.

2. **Palm: Predicting Actions through Language Models @ Ego4D [14] Long-Term Action Anticipation Challenge 2023 [16]**

   This work tackles the Long-Term Action Anticipation (LTA) challenge on the Ego4D [14] dataset. The inputs of this task are an input video of approximately 5 minutes, the start timestep, and the end timestep of each action instance in this video. The expected output of this task is the predicted (verb, noun) pair of the next 20 action instances.

   The authors modify the transformer architecture used in the Ego4D [14] baseline of the LTA challenge by attaching two classification heads to it. The input of this modified architecture is the EgoVLP [21] features of 4 consecutive action instances in the input video. They train this modified architecture so that one of the classification heads predicts the noun of the next action instance and the other one predicts the verb of the next action instance. After training this architecture, they use it to make a (verb, noun) prediction for the last 8 action instances in a given input video.

   The authors also take the center frame of the last 8 action instances in each input video and they use BLIP-2 [20] to generate an image caption. They use "The person is" prompt while querying BLIP-2 [20].

   They feed the image captions and predicted (verb, noun) pairs of the last 8 action instances of a given video as input to a large language model that predicts the next 20 action instances.

   When the time the authors submitted their paper, this approach ranked 1st place in the leaderboard of the Ego4D [14] LTA challenge.

2.3 The Pre-Trained Architectures Used in This Work

1. **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks [28]**

   Sentence-BERT (SBERT) [28] inserts a pooling operation into the output of BERT [7]. This pooling operation can be using CLS token, computing the mean of output vectors, or computing the max-over-time of output vectors. They fine-tune this architecture on three objective functions:

   - **Classification Objective**: Assuming there are two sentences with the following sentence embeddings: (u, v) where $u, v \in \mathbb{R}^n$, and the same label c. The authors calculate $|u - v|$ which is the concatenation of the absolute difference of matching elements of u and v vectors. They create the following concatenated embedding: $(u, v, |u - v|) \in \mathbb{R}^{3n}$. They multiply this vector with a learnable weight matrix $W_t \in \mathbb{R}^{kx3n}$ and apply the softmax function on the result of the multiplication: $o = softmax(W_t(u, v, |u - v|)) \in \mathbb{R}^k$. They reduce the cross entropy loss between o and the ground truth one-hot vector whose entry corresponding to the label c is 1 and other entries are 0.
   - **Regression Objective Function**: They calculate the cosine similarity between sentence embeddings u and v. If two sentences are positive pairs, then they reduce the mean squared error between the cosine similarity and 1. If they are negative pairs, then they reduce the mean squared error between the cosine similarity and 0.

- **Triplet Objective Function**: For a given sentence a, a sentence p which is positive pair of a and a sentence n which is negative pair of a, they fine-tune the architecture so that the distance between embeddings of a and p are at least ϵ less than the distance between embeddings of a and n. To achieve this goal, they minimize the following objective function: $max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$ where s_x is the sentence embedding of sentence x.

In contrast to BERT [7] and RoBERTa [22] embeddings, SBERT [28] embeddings are semantically meaningful and the distance between sentence embeddings can be compared using cosine similarity. This is the main reason why we use SBERT [28] in our work instead of BERT [7] and RoBERTa [22].

2. **BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models [20]**

   We use both the BLIP-2 [20] with decoder-based large language model and the BLIP-2 [20] with encoder-decoder-based large language model in our experiments. The BLIP-2 [20] with decoder-based large language model is used in BLIP-2 [20] Prompt and Post-Processing Method Ablation, and BLIP-2 [20] Caption - Action Category Matching Success Evaluation experiments. The BLIP-2 [20] with encoder-decoder-based language model is used in BLIP-2 [20] Embedding Ablation experiment. The reason why we use BLIP-2 [20] with an encoder-decoder-based large language model in this experiment is that we wanted to explore the informativeness of the output of the large language model encoder for the prediction of the action category of the input image.

   Both architectures use a frozen image encoder to encode an input image and apply a transformer architecture (Q-Former) to the image encoding. This transformer architecture is applied to the same image encoding with different learned queries. The output of the Q-Former is fed to a fully connected neural network.

   In the decoder-based BLIP-2 [20], the output of the fully connected neural network is fed to a frozen large language model decoder which generates an output text.

   In the encoder-decoder-based BLIP-2 [20], the output of the fully connected neural network is concatenated with a prefix text and fed to a large language model encoder. The output of the large language model encoder is fed to a large language model decoder which generates the continuation of the prefix text.

Chapter 3

Annotation Analysis and Evaluation Metrics

3.1 Annotation Analysis

The analyses in subsections 2, 5, 6, 7, and 8 are done using annotations of train and validation split. The analyses in 1, 3, and 4 are done using annotations of all splits.

1. **Number of Clips per Subset Histogram**

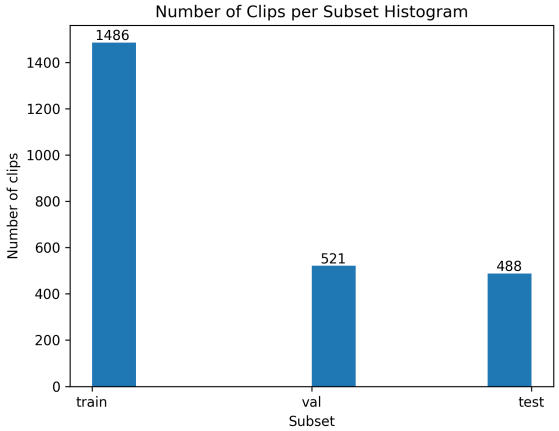

Figure 3.1: Number of Clips per Subset Histogram

2. **Number of Annotations per Clip Histogram**

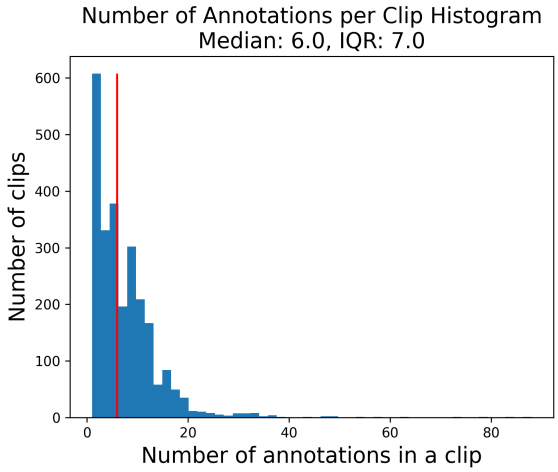

Figure 3.2: Number of Annotations per Clip Histogram

3. **Number of Clips per Video Histogram**

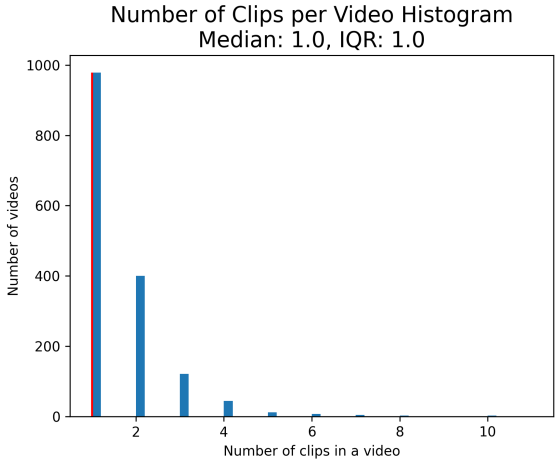

Figure 3.3: Number of Clips per Video Histogram

4. **Clip Duration Histogram**

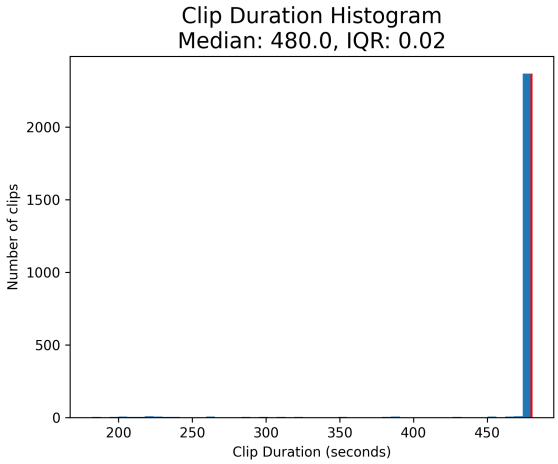

Figure 3.4: Clip Duration Histogram

5. **Annotation Duration Histogram**

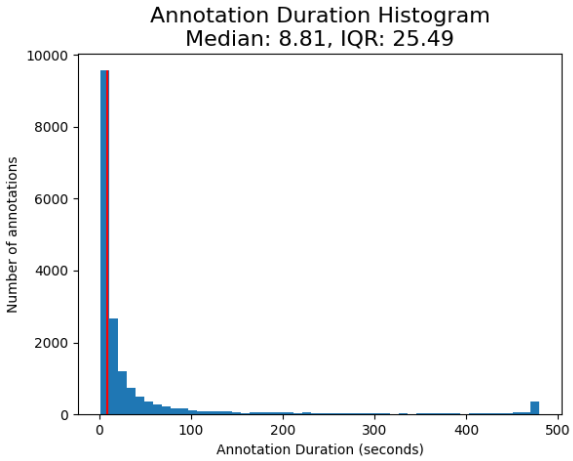

Figure 3.5: Annotation Duration Histogram

6. **Action Category Counts**

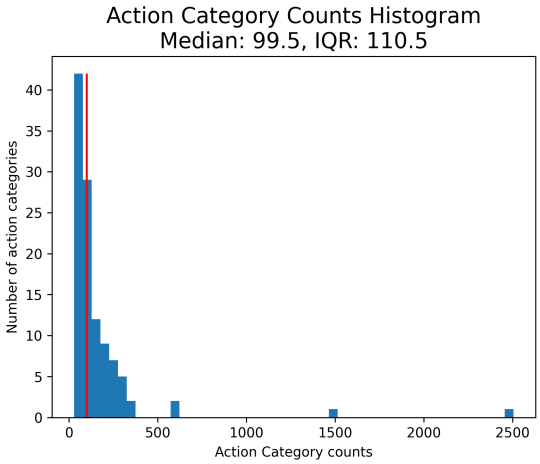

Figure 3.6: Action Category Counts Histogram

The action category - exact count mapping can be found at Table A.1, Table A.2 and Table A.3.

7. **Annotation tIoU Histogram**

For each annotation, we check the tIoU with other annotations in the same clip. We exclude tIoUs between non-intersecting annotations.

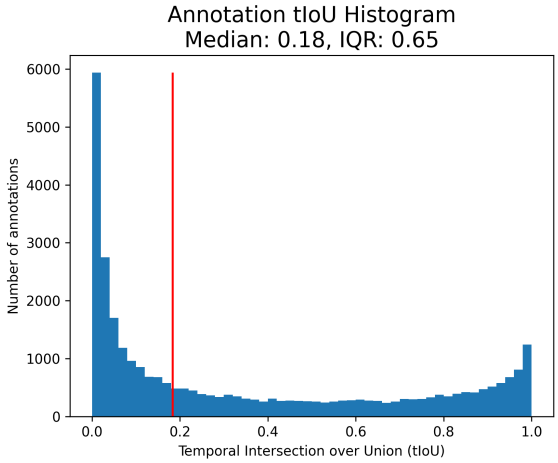

Figure 3.7: Annotation tIoU Histogram

8. **Number of Intersections Histogram**

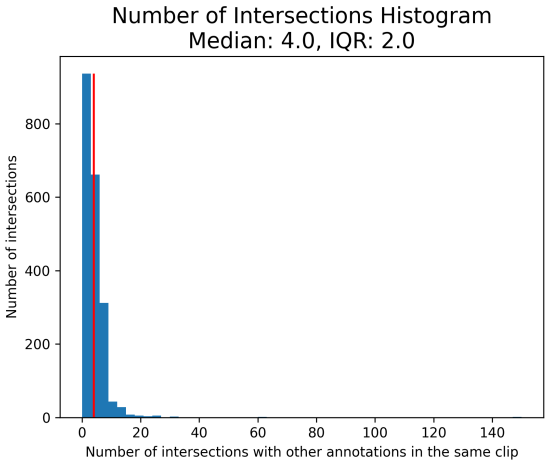

Figure 3.8: Number of Intersections Histogram

3.2 Evaluation Metrics

3.2.1 With-Backbone Evaluation Metrics

These evaluation metrics are the evaluation metrics used in Ego4D-MQC. These evaluation metrics are only used when a backbone is used while generating predictions.

1. **Average Mean Average Precision of Detection Score (AvgMeanAP)**

   To define the AvgMeanAP metric it is beneficial to first define the metrics that are used to calculate this metric.

   - **Temporal Intersection over Union (tIoU):** For each predicted action instance, a ground truth action instance that intersects with the predicted action instance is found. Temporal Intersection over Union is the length of the intersection divided by the length of the union of these two temporal regions [17].
   - **True Positive:** A segment of the video whose tIoU is larger than the given threshold.
   - **Precision:** Number of true positives divided by the number of video segments the model assigned to the given action category **c**.
   - **Recall:** Number of true positives divided by the actual number of video segments that correspond to the given action category **c**.
   - **Average Precision (AP):** Area under the precision-recall curve [34]. This metric is calculated for each action category separately.
   - **Mean Average Precision (MeanAP):** Mean over Average Precision values for different action categories.
   - **Average Mean Average Precision (AvgMeanAP):** Average of MeanAP values which were calculated using different thresholds. In Ego4D-MQC, the following thresholds are used: 0.1, 0.2, 0.3, 0.4, 0.5.

2. **Recall@kx,tIoU=m of Retrieval Score**

   Assuming there are $\mathbf{x}$ ground truth action instances for the action category $\mathbf{c}$ in a given video $\mathbf{V}$.

   Also assuming a model predicts that there are $\mathbf{N}$ video segments which is an instance of the action category $\mathbf{c}$ in this given video $\mathbf{V}$. The model also predicts the start time ($\mathbf{t_{n,s}}$), the end time ($\mathbf{t_{n,e}}$), and the probability of being an instance of the action category $\mathbf{c}$ for each of the $\mathbf{N}$ predicted video segments.

   Among the $\mathbf{N}$ predicted video segments, $\mathbf{x} * \mathbf{k}$ segments with the highest estimated probabilities are selected. These segments will be called "top-kx predicted video segments".

   Among the ground truth action instances, we find the ones which have a tIoU larger than $\mathbf{m}$ with at least one of these "top-kx predicted video segments". These ground truth action instances will be called "ground truth action instances that were predicted correctly by the top-kx predicted video segments".

   Recall@kx,tIoU=m measures 100 * total number of ground truth action instances that were predicted correctly by the top-kx predicted video segments in the dataset / total number of ground truth action instances in the dataset.

   Unlike the AvgMeanAP metric, this metric does not penalize false positives when the action category has no ground truth instances in a given video [5].

3.2.2 Without-Backbone Evaluation Metrics

These evaluation metrics are used when the predictions are made without using a backbone. We first provide definitions of the evaluation metrics used while defining without-backbone evaluation metrics. Then we provide the definitions of without-backbone evaluation metrics.

Evaluation metrics used while defining without-backbone evaluation metrics

- **True Positive Count of c:** The number of frames that are predicted to be an instance of action category c and that are an instance of action category c.
- **False Positive Count of c:** The number of frames that are predicted to be an instance of action category c, however, that are not an instance of action category c.
- **False Negative Count of c:** The number of frames that are an instance of action category c, however, that are not predicted to be an instance of action category c.
- **Precision Score of c:**
$$\frac{\text{True Positive Count of } c}{\text{True Positive Count of } c + \text{False Positive Count of } c}$$
- **Recall Score of c:**
$$\frac{\text{True Positive Count of } c}{\text{True Positive Count of } c + \text{False Negative Count of } c}$$
- **F1 Score of c:**
$$\frac{2 \,.\, \text{Precision Score of } c \,.\, \text{Recall Score of } c}{\text{Precision Score of } c + \text{Recall Score of } c}$$
- **Weighted X Score:** X Score is calculated for each action category. Weighted X Score is the weighted average of X Scores where the weight of action category c =
$$\frac{\text{Number of frames that are an instance of action category } c}{\text{Total number of frames}}$$

Without-backbone evaluation metrics

- **Weighted Precision Score with Background:** Weighted precision score calculated on all frames including the background frames.
- **Weighted Recall Score with Background:** Weighted recall score calculated on all frames including the background frames.
- **Weighted F1 Score with Background:** Weighted F1 score calculated on all frames including the background frames.
- **Weighted Precision Score without Background:** Weighted precision score calculated on all frames except the background frames.
- **Weighted Recall Score without Background:** Weighted recall score calculated on all frames except the background frames.
- **Weighted F1 Score without Background:** Weighted F1 score calculated on all frames except the background frames.
- **Background Precision Score:** Precision score calculated on only background frames.
- **Background Recall Score:** Recall score calculated on only background frames.
- **Background F1 Score:** F1 score calculated on only background frames.

Chapter 4

Our Work

1. **BLIP-2 [20] Prompt and Post-Processing Method Ablation:** We use the same questions used in TransFusion [25] to query BLIP-2 [20] with a decoder-based large-language model. These questions are "What does the image describe?", "What is the person in this picture doing?" and "What is happening in this picture?". We apply two types of post-processing methods to each BLIP-2 [20] caption:

   (a) **Dependency Parsing + Matching-Based Scoring:** We use the dependency parsing algorithm (provided in section A.2) to extract (verb, noun, tool) pairs from each caption. We manually extract (verb, noun, tool) pairs from each action category (provided in section A.4). We then use matching-based scoring algorithm (provided in section A.3) to assign scores to each (action category, caption) pair. For a given threshold $t \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, and a caption c, the predicted action categories of c are the action categories whose matching-based score is larger than or equal to t.

   Explanation of Dependency Parsing Algorithm:

   - Clean the BLIP-2 caption such that the cleaned version only involves letters, whitespaces, and apostrophes.
   - Split the text into words.
   - Apply Stanford CoreNLP's [24] CoreNLPDependencyParser [23] to the splitted caption.
   - Find the addresses of "using" words, and store them for future use.
   - Iterate over words. For each word,
     - If the word's part-of-speech tag starts with "VB", the current word is "using" and the previous word is "is" then continue.
     - If the word's part of speech tag starts with "VB" and the lemma of this word is one of ["be", "describe", "show", "have", "tell"], then continue.
     - If the word's part-of-speech tag starts with "VB", however, none of the previous two conditions hold, then
       - Find the verb lemma:
         - Check whether there is a word that depends on the verb with a "compound:prt" relationship. If there is, then take the lemma of the word that depends on the verb, and append it to the verb's lemma in the following setting: lemma of the verb + " " + lemma of the word that depends on the verb. This concatenation becomes the new verb lemma in this case. By this way, we take care of the case when the verb consists of two words (e.g. "shut down").

        - If no word depends on the verb with a "compound:prt" relationship then the verb lemma is the current verb's lemma.
        - Replace each "-" with a " " in the verb's lemma.
    - Find the tool lemmas:
        - Find the words that are related to the verb with an "obl" relationship (call them "obl" words). For each "obl" word, find the words that depend on the "obl" word with a "case" relationship (call them "case" words of the "obl" word). If a "case" word of the "obl" word is "with", then this "case" word is a tool.
        - Find the words that are dependent on one of the "using" words we found before with an "obj" dependency. These words are tools.
        - Find the words that are dependent on "using" words we found before with an "acl" dependency (call them "acl" words). Then find the words that are dependent on one of the "acl" words with an "obj" dependency. These words are also tools.
        - For each tool word, check whether there is a word (call it a compound word) that depends on this tool word with a compound relationship. If there is, the tool lemma becomes the following: lemma of the compound word + " " + lemma of the tool word. Otherwise, keep the tool lemma as the same.
        - Replace each "-" with a " " in all tool lemmas.
    - Find the noun lemmas:
        - Find the words that depend on the verb with an "obj" or "nsubj:pass" relationship.
        - For each noun word, check whether there is a word (call it compound word) that depends on this noun word with a compound relationship. If there is, the current noun lemma becomes the following: lemma of the compound word + " " + lemma of the noun word. Otherwise, the current noun lemma is the noun word's lemma.
        - Replace "-" with " " from all noun lemmas.
    - For each noun lemma, and for each tool lemma we found, store the following tuple: (current verb's lemma, noun lemma, tool lemma).
- If the word's part-of-speech tag starts with "NN"
    - Check whether there is a word (call it a compound word) that depends on this noun word with a compound relationship. If there is, attach the lemma of the compound word to the beginning of the lemma of the noun word with a whitespace between them. Otherwise, keep the lemma of the noun word as the same.
    - Replace each "-" with a " " in the noun word's lemma.
    - For each noun lemma, store the following tuple: ("NaN", noun lemma, "NaN")

Explanation of Dictionary-Matching Scoring Algorithm: This algorithm takes all (verb, noun, tool) pairs extracted from a given BLIP-2 caption using the dependency parsing algorithm and all (verb, noun, tool) pairs that are manually extracted from an action category. For each caption (verb, noun, tool) pair and for each action category (verb, noun, tool) pair it checks whether there is a match between these two pairs. If there are multiple matches between a caption's list of (verb, noun, tool) pairs and action category's list of (verb, noun, tool) pairs, the highest match score is returned. The match types and their scores are as follows:

- Caption (verb, noun, tool) == Action Category (verb, noun, tool): 1.0
- Caption (verb, noun) == Action Category (verb, noun): 0.8
- Caption (verb, tool) == Action Category (verb, tool): 0.6
- Caption (noun) == Action Category (noun): 0.4

- Caption (verb) == Action Category (verb): 0.2

(b) **SBERT [28] Embedding + Scaled Cosine Similarity:** For each caption, we calculate the scaled cosine similarities between the SBERT [28] embedding of this caption and the SBERT [28] embedding of each action category's manually extracted phrases. For a given threshold $t \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, and a caption c, the predicted action categories of c are the action categories whose at least one of the SBERT [28] embeddings has a scaled cosine similarity with the SBERT [28] embedding of c that is larger than or equal to t. The scaled cosine similarity metric is the linear transformation $(x \rightarrow (x + 1.0)/2.0)$ of the cosine similarity metric so that its output is between 0.0 and 1.0.

2. **BLIP-2 [20] Caption - Action Category Match Success Evaluation:** For each video's each frame's each BLIP-2 [20] caption, we check whether we have a match between this BLIP-2 [20] caption's (verb, noun, tool) pairs and (verb, noun, tool) pairs of this frame's ground truth action categories. We report the frequency of frames in the validation split whose caption is matched to one of the ground truth action categories of this frame.

3. **Comparison of Our Predictions with ASL-Ego4D [30] Predictions:**

   In our comparisons, in addition to the feature combination (Omnivore [13] + Slowfast [10] + EgoVLP [21] + InternVideo [4]) used by the state-of-the-art approach (ASL-Ego4D [30]), we also experiment with the following feature combinations:

   - **Proposed Features v1:** BLIP-2 [20] LLM Encoder Output
   - **Proposed Features v2:** (BLIP-2 [20] Output + SBERT [28])
   - **Proposed Features v3:** BLIP-2 [20] LLM Encoder Output + (BLIP-2 [20] Output + SBERT [28])
   - **Proposed Features v4:** Omnivore [13] + Slowfast [10] + EgoVLP [21] + InternVideo [4] + BLIP-2 [20] LLM Encoder Output
   - **Proposed Features v5:** Omnivore [13] + Slowfast [10] + EgoVLP [21] + InternVideo [4] + (BLIP-2 [20] Output + SBERT [28])
   - **Proposed Features v6:** Omnivore [13] + Slowfast [10] + EgoVLP [21] + InternVideo [4] + BLIP-2 [20] LLM Encoder Output + (BLIP-2 [20] Output + SBERT [28])

   We use the prompt "What is the person in this picture doing?" while extracting (BLIP-2 [20] Output + SBERT [28]) embeddings.

   We do two comparisons:

   (a) **Action Detection and Retrieval:** We use BLIP-2 [20] on frames of a given video clip. We apply SBERT [28] on the generated caption. We use the extracted embeddings as input to the backbone of the current state-of-the-art approach (ASL-Ego4D [30]) both with and without the video features already being used by ASL-Ego4D [30] approach. To report the ValAvgMeanAP and ValRec@1x,tIoU=0.5, we train the ASL-Ego4D [30] backbone on the train split, and evaluate it on the validation split. To report the TestAvgMeanAP and TestRec@1x,tIoU=0.5, we train this backbone on the concatenation of train and validation splits and evaluate it on the test split. In both cases, we use the model weights of the 11th epoch during inference, as suggested by the authors of ASL-Ego4D [30] in one of their replies in their repository [29].

Feature	Dimension
Omnivore [13]	1536
Slowfast [10]	2304
EgoVLP [21]	256
InternVideo [4]	1024
BLIP-2 [20] Output + SBERT [28]	768
BLIP-2 [20] LLM Encoder Output	94208

Table 4.1: Feature Dimensions

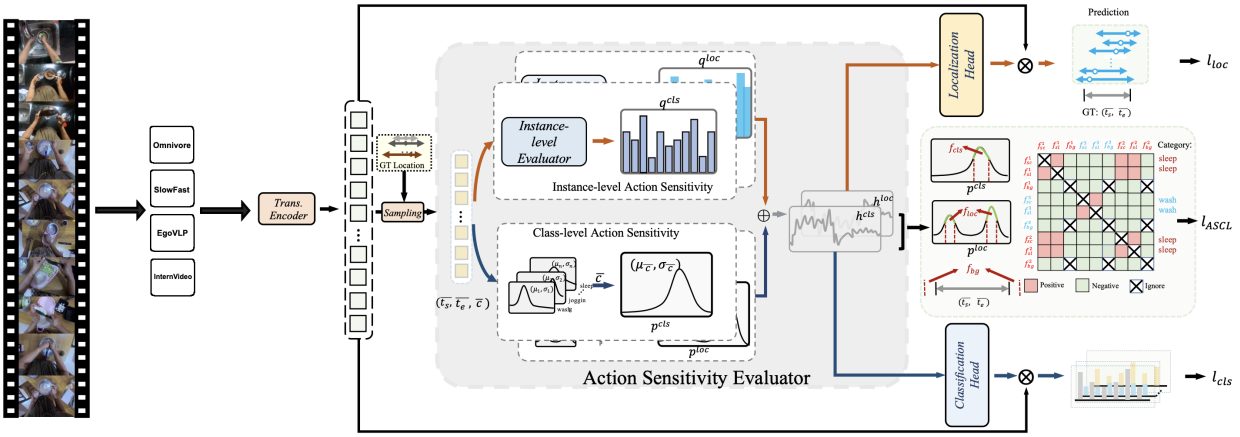

Figure 4.1: Visualization of the State-of-the-Art Approach (Taken from [30])

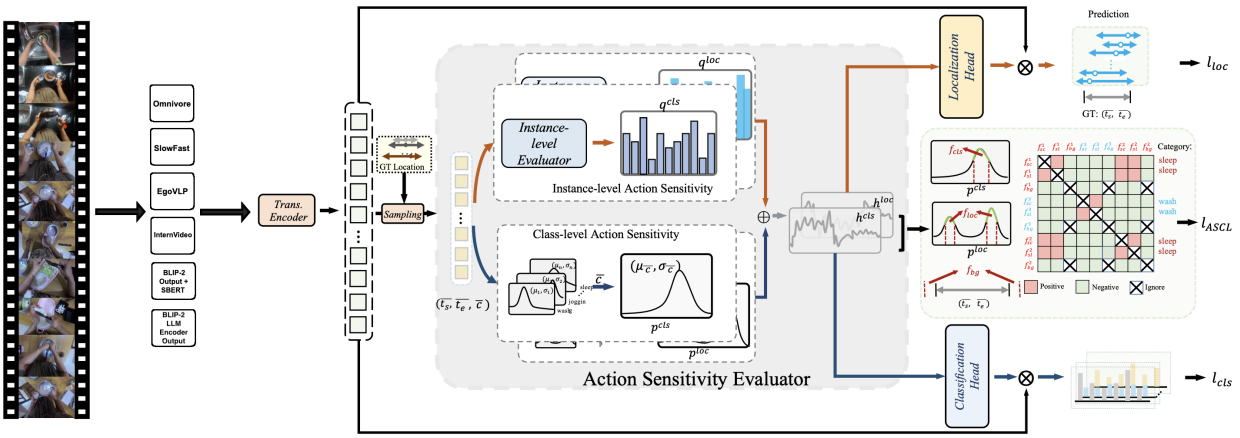

Figure 4.2: Visualization of Our Approach (Proposed Features v6, Adapted from [30])

(b) **Frame-wise Action Category Classification:** We use the without-backbone evaluation metrics on the predictions made on the validation split using the ASL-Ego4D [30] backbone in Action Detection and Retrieval section.

4. **Vision-Language Model Fine-Tuning Script Implementation:** We propose a script that fine-tunes a given vision-language model on a given Visual Question Answering (VQA) dataset which can be used for making the BLIP-2 [20] embeddings more informative for the Ego4D-MQC. This script only fine-tunes the parameters of the Q-Former and the Vision Encoder of the language model. We use accelerate [15] and transformers [37] packages of Python3 [35]. This script can be found at https://github.com/ardarslan/egocentric-video-understanding/blob/master/scripts/06_blip2_caption_analysis/06_fine_tune_blip2_frame_wise/02_fine_tune_blip2_frame_wise.py. The required GPU memory to run this script was larger than the GPU memory that was available to us, therefore we do not have the results from fine-tuned BLIP-2 [20] in our report.

5. **Egocentric Video Visualization Tool Modification:** We modify an existing egocentric video visualization tool (implemented by Alexey Gavryushin) so that it could be used in our task. The existing egocentric video visualization tool uses Python3 [35] and Django [8]. The code for this tool can be found at https://github.com/ardarslan/egocentric-video-understanding/tree/master/scripts/05_visualize_frame_features/02_stream_video. Two examples for the egocentric video visualization tool are as follows:

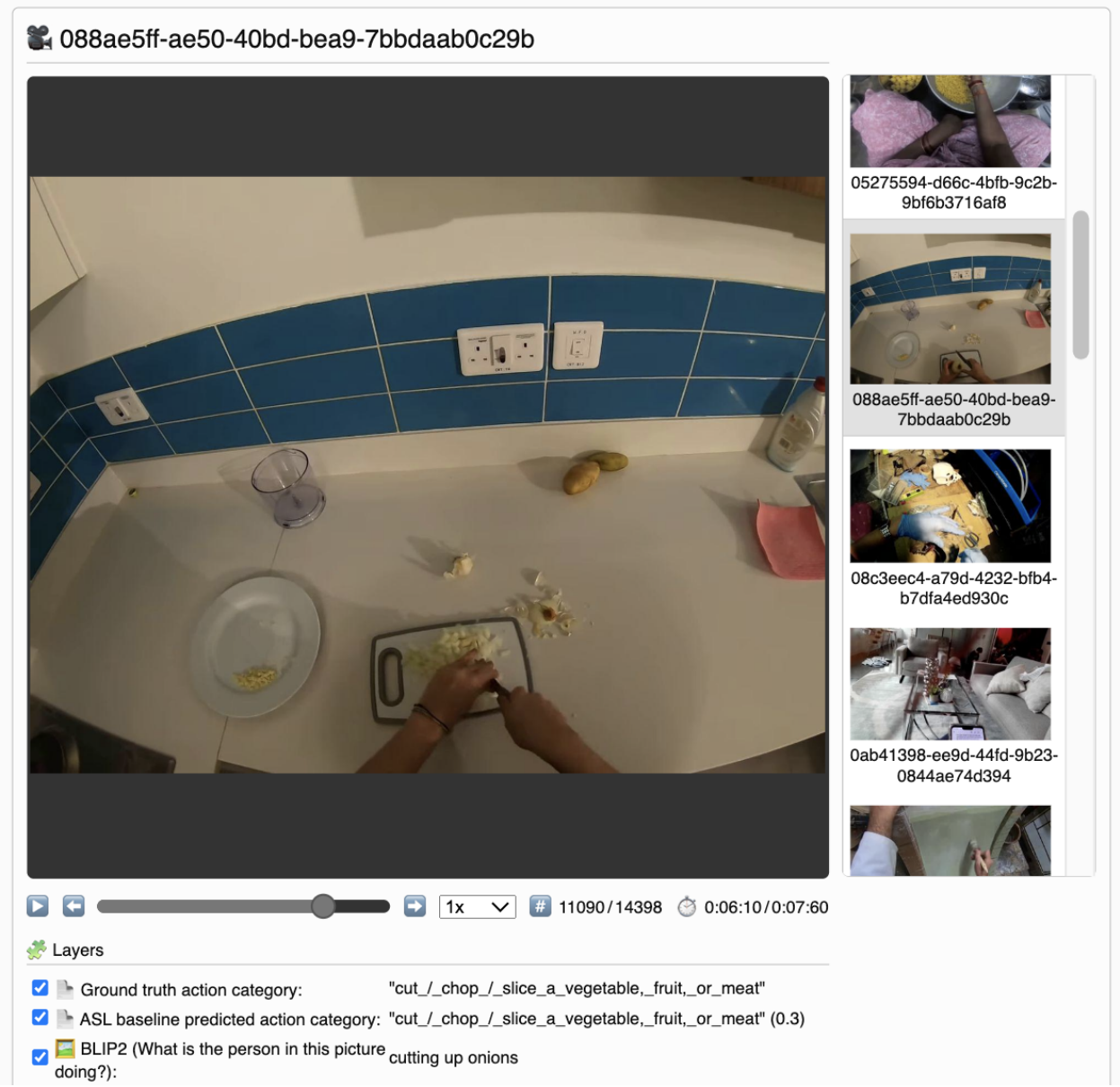

Figure 4.3: Screenshot of the Modified Egocentric Video Visualization Tool (Successful Caption)

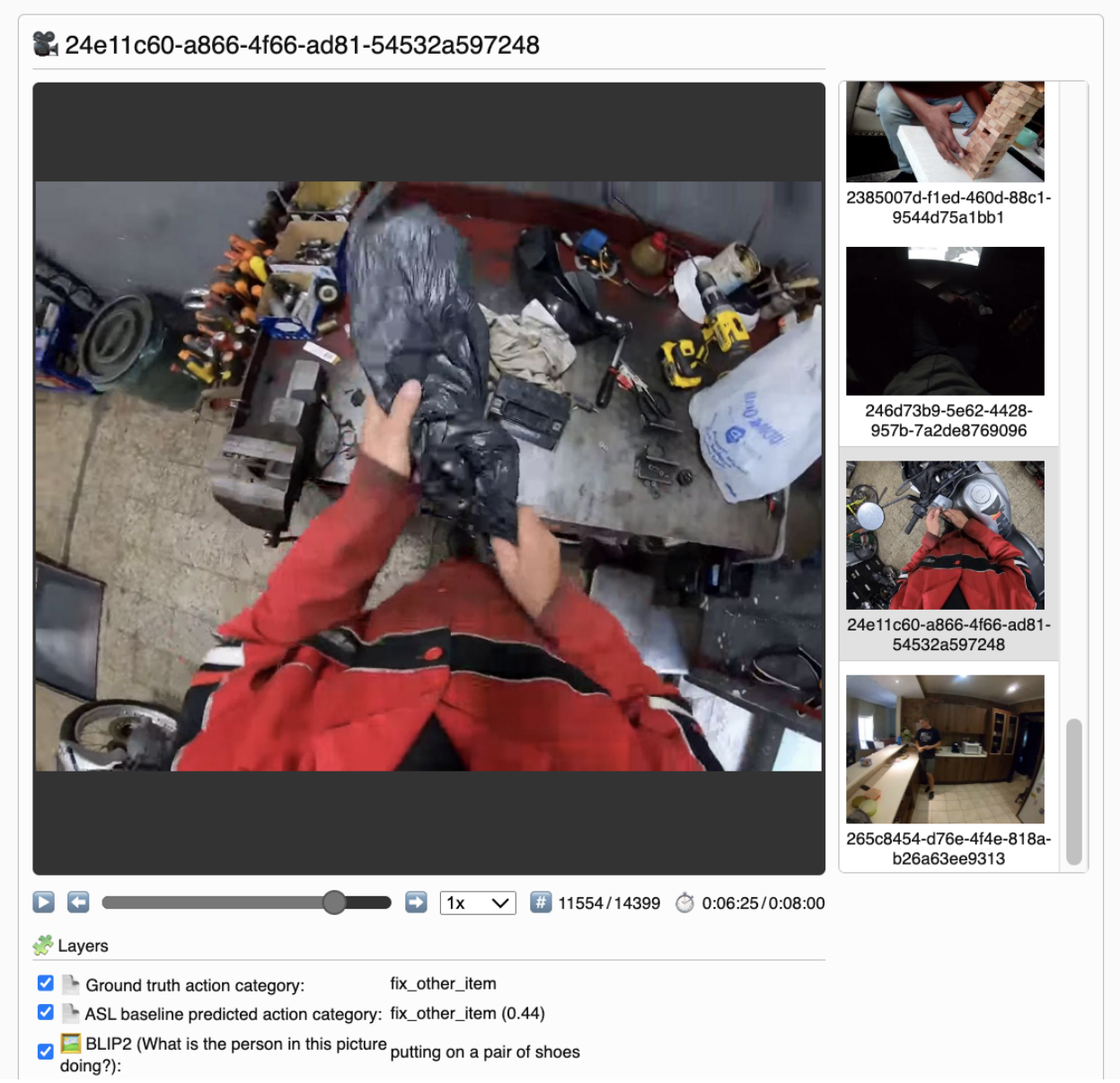

Figure 4.4: Screenshot of the Modified Egocentric Video Visualization Tool (Unsuccessful Caption)

6. **New Visualization Tool Implementation:** Finally, we propose a new visualization tool that allows the user to visualize the dependency parsing features, predicted action categories, ground truth action categories, and the correctness of the predictions in a per-frame setting. We use Python3 [35], Plotly [18], and Dash extension of Plotly [18] to implement this tool. The code for this tool can be found at https://github.com/ardarslan/egocentric-video-understanding/blob/master/scripts/05_visualize_frame_features/03_horizontal_bar_plots/horizontal_bar_plots.py. Two screenshots of this visualization tool is as follows:

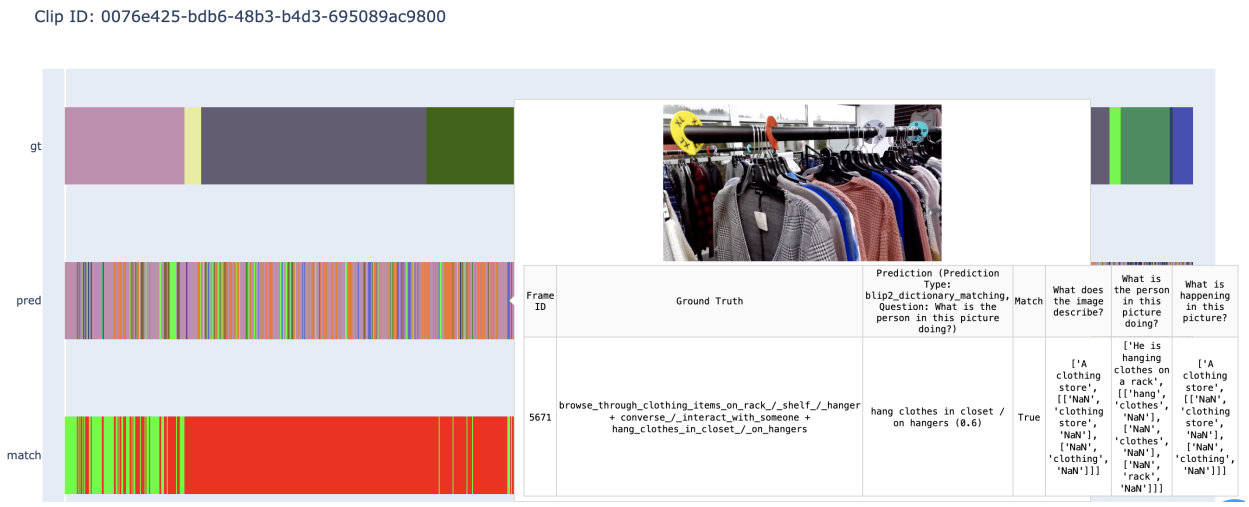

Figure 4.5: Screenshot of the New Visualization Tool (Match)

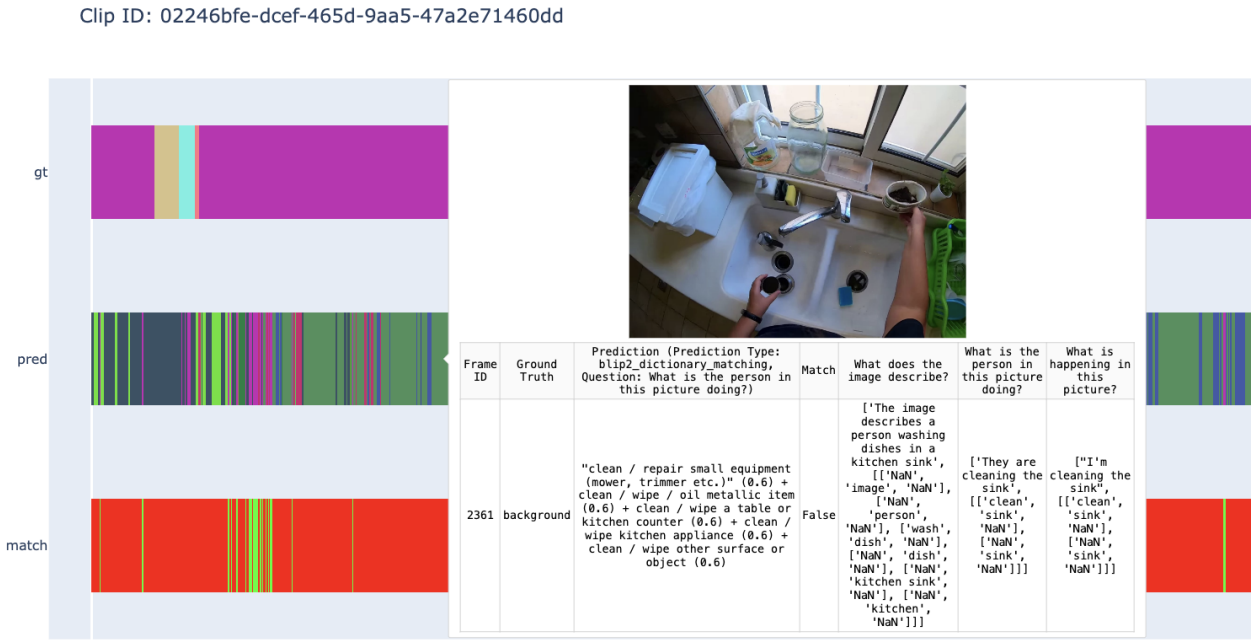

Figure 4.6: Screenshot of the New Visualization Tool (No Match)

Chapter 5

Experiment Results

In all tables, and for all evaluation metrics, the higher the scores the better. In the tables below, we use some abbreviations. Their explanations are as follows:

- **Q:** Question
- **B.F1:** Background F1 Score
- **B.P:** Background Precision Score
- **B.R:** Background Recall Score
- **WWB.F1:** Weighted F1 Score with Background
- **WWB.P:** Weighted Precision Score with Background
- **WWB.R:** Weighted Recall Score with Background
- **WNB.F1:** Weighted F1 Score without Background
- **WNB.P:** Weighted Precision Score without Background
- **WNB.R:** Weighted Recall Score without Background
- **BMR:** Background Match Ratio. Among all frames that do not belong to an action instance, the ratio of frames which is predicted as background.
- **VNTMR:** Verb-Noun-Tool Match Ratio. Among all frames that belong to an action instance, the ratio of frames whose at least one ground truth action category and at least one BLIP-2 [20] caption has the same (verb, noun, tool) pair.
- **VNMR:** Verb-Noun Match Ratio. Among all frames that belong to an action instance, the ratio of frames whose at least one ground truth action category and at least one BLIP-2 [20] caption has the same (verb, noun) pair.
- **VTMR:** Verb-Tool Match Ratio. Among all frames that belong to an action instance, the ratio of frames whose at least one ground truth action category and at least one BLIP-2 [20] caption has the same (verb, tool) pair.
- **VMR:** Verb Match Ratio. Among all frames that belong to an action instance, the ratio of frames whose at least one ground truth action category and at least one BLIP-2 [20] caption has the same verb.

- **NMR:** Noun Match Ratio. Among all frames that belong to an action instance, the ratio of frames whose at least one ground truth action category and at least one BLIP-2 [20] caption has the same noun.
- **Q = 0:** What does the image describe?
- **Q = 1:** What is the person in this picture doing?
- **Q = 2:** What is happening in this picture?
- **Q = 0-1-2:** Use the replies given to all questions (0-1-2).
- **Threshold=t where t $\in$ {0.2, 0.4, 0.6, 0.8, 1.0}:** For a given frame, an action category c is predicted to be true if its score is larger than or equal to the threshold t.
- **Threshold=max:** For a given frame, only the action category with the largest score is predicted to be true.

1. **BLIP-2 [20] Prompt and Post-Processing Method Ablation**

   In most of the cases, we get better results by using replies given to all questions instead of using only the replies given to a single question.

   (a) **Dependency Parsing + Matching-Based Scoring**

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	**0.354**	0.483	**0.279**	**0.195**	0.240	**0.330**	**0.070**	0.046	**0.370**
1	0.292	**0.516**	0.204	0.154	0.246	0.251	0.045	0.030	0.290
2	0.246	0.505	0.163	0.142	0.248	0.217	0.058	0.043	0.260
0-1-2	0.105	0.506	0.058	0.082	**0.255**	0.128	0.065	**0.055**	0.184

Table 5.1: Dependency Parsing + Matching-Based Scoring Results (Threshold=Max)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.016	0.475	0.008	0.023	0.315	0.015	0.028	0.187	0.019
1	0.003	**0.574**	0.001	0.023	**0.377**	0.016	0.039	**0.220**	0.027
2	0.017	0.561	0.009	0.035	0.368	0.022	0.050	0.214	0.033
0-1-2	**0.034**	0.518	**0.018**	**0.060**	0.352	**0.041**	**0.080**	**0.220**	**0.060**

Table 5.2: Dependency Parsing + Matching-Based Scoring Results (Threshold=1.0)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.016	0.475	0.008	0.025	0.313	0.016	0.031	0.183	0.023
1	0.003	**0.574**	0.001	0.025	**0.375**	0.019	0.044	0.216	0.033
2	0.017	0.561	0.009	0.043	0.372	0.033	0.063	**0.222**	0.052
0-1-2	**0.034**	0.518	**0.018**	**0.064**	0.353	**0.051**	**0.088**	0.221	**0.078**

Table 5.3: Dependency Parsing + Matching-Based Scoring Results (Threshold=0.8)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.016	0.475	0.008	0.043	0.323	0.041	0.065	0.202	0.068
1	0.003	**0.574**	0.001	0.060	**0.384**	0.084	0.105	**0.233**	0.150
2	0.017	0.561	0.009	0.063	0.377	0.077	0.099	0.230	0.132
0-1-2	**0.034**	0.518	**0.018**	**0.083**	0.352	**0.121**	**0.122**	0.220	**0.204**

Table 5.4: Dependency Parsing + Matching-Based Scoring Results (Threshold=0.6)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.016	0.475	0.008	0.065	0.314	0.093	0.105	0.185	0.160
1	0.003	**0.574**	0.001	0.066	**0.366**	0.112	0.116	0.201	0.200
2	0.017	0.561	0.009	0.074	0.360	0.126	0.120	**0.200**	0.220
0-1-2	**0.034**	0.518	**0.018**	**0.085**	0.331	**0.181**	**0.126**	0.182	**0.311**

Table 5.5: Dependency Parsing + Matching-Based Scoring Results (Threshold=0.4)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.016	0.475	0.008	0.065	0.309	0.103	0.104	0.177	0.180
1	0.003	**0.574**	0.001	0.065	**0.363**	0.125	0.115	**0.194**	0.224
2	0.017	0.561	0.009	0.072	0.357	0.145	0.116	0.193	0.253
0-1-2	**0.034**	0.518	**0.018**	**0.082**	0.327	**0.195**	**0.120**	0.175	**0.337**

Table 5.6: Dependency Parsing + Matching-Based Scoring Results (Threshold=0.2)

(b) **SBERT [28] + Cosine Similarity**

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.0	0.0	0.0	0.084	0.144	0.104	0.151	0.259	0.186
1	0.0	0.0	0.0	0.086	0.151	0.108	0.155	0.272	0.194
2	0.0	0.0	0.0	0.092	0.147	0.111	0.165	0.264	0.200
0-1-2	0.0	0.0	0.0	**0.094**	**0.156**	**0.118**	**0.169**	**0.281**	**0.212**

Table 5.7: SBERT [28] + Cosine Similarity (Threshold=Max)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.016	0.475	0.008	0.023	0.315	0.014	0.028	0.187	0.019
1	0.003	**0.574**	0.001	0.023	**0.377**	0.016	0.040	**0.220**	0.027
2	0.017	0.561	0.009	0.035	0.368	0.023	0.050	0.214	0.033
0-1-2	**0.034**	0.518	**0.018**	**0.059**	0.352	**0.041**	**0.080**	**0.220**	**0.060**

Table 5.8: SBERT [28] + Cosine Similarity (Threshold=1.0)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.0	0.0	0.0	0.037	0.099	0.029	0.066	0.177	0.052
1	0.0	0.0	0.0	0.040	**0.222**	0.034	0.072	**0.400**	0.061
2	0.0	0.0	0.0	0.040	0.167	0.032	0.072	0.300	0.058
0-1-2	0.0	0.0	0.0	**0.065**	0.162	**0.065**	**0.118**	0.291	**0.116**

Table 5.9: SBERT [28] + Cosine Similarity (Threshold=0.8)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.0	0.0	0.0	0.061	**0.073**	0.281	0.110	**0.131**	0.504
1	0.0	0.0	0.0	0.066	0.052	0.352	0.119	0.094	0.632
2	0.0	0.0	0.0	**0.071**	0.053	0.308	**0.127**	0.095	0.553
0-1-2	0.0	0.0	0.0	0.065	0.045	**0.408**	0.118	0.080	**0.733**

Table 5.10: SBERT [28] + Cosine Similarity (Threshold=0.6)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.0	0.0	0.0	**0.040**	**0.022**	0.552	**0.0716**	**0.0393**	0.992
1	0.0	0.0	0.0	**0.040**	**0.022**	**0.556**	0.0712	0.0390	0.999
2	0.0	0.0	0.0	**0.040**	**0.022**	**0.556**	0.0712	0.0390	0.999
0-1-2	0.0	0.0	0.0	**0.040**	**0.022**	**0.556**	0.0712	0.0390	**1.0**

Table 5.11: SBERT [28] + Cosine Similarity (Threshold=0.4)

Q	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
0	0.0	0.0	0.0	**0.040**	**0.022**	0.552	**0.072**	**0.039**	0.992
1	0.0	0.0	0.0	**0.040**	**0.022**	**0.556**	0.071	**0.039**	0.999
2	0.0	0.0	0.0	**0.040**	**0.022**	**0.556**	0.071	**0.039**	**1.0**
0-1-2	0.0	0.0	0.0	**0.040**	**0.022**	**0.556**	0.071	**0.039**	**1.0**

Table 5.12: SBERT [28] + Cosine Similarity (Threshold=0.2)

2. **BLIP-2 [20] Caption - Action Category Match Success Evaluation**

   The prompt "What is the person in this picture doing?" yields the highest verb match ratio (0.205) and the highest (verb, tool) match ratio (0.168).

   The prompt "What is happening in this picture?" yields the highest background match ratio (0.009), the highest (verb, noun, tool) match ratio (0.037), the highest (verb, noun) match ratio, and the highest noun match ratio (0.157).

Q	BMR	VNTMR	VNMR	VTMR	VMR	NMR
0	0.008	0.022	0.028	0.072	0.100	0.132
1	0.001	0.030	0.040	**0.168**	**0.205**	0.102
2	**0.009**	**0.037**	**0.064**	0.129	0.194	**0.157**

Table 5.13: BLIP-2 [20] Caption - Action Category Match Success Evaluation Results

3. **Comparison of Our Predictions with ASL-Ego4D [30] Predictions**

(a) **Action Detection and Retrieval:**

When we use only the SBERT [28] embedding of the BLIP-2 [20] captions as input to the ASL-Ego4D [30] backbone or only the output of the encoder of the large language model of BLIP-2 [20], the model performance decreases severely. When we use both of these features together with the video features already being used by the state-of-the-art approach (ASL-Ego4D [20]), we get better results compared to the results of the state-of-the-art approach.

Approach	ValAvgMeanAP	ValRec@1x,tIoU=0.5	TestAvgMeanAP	TestRec@1x,tIoU=0.5
Feats of [30]+[30] Backbone	26.13	44.62	25.41	42.75
Proposed Feats v1+[30] Backbone	12.87	30.38	13.16	28.51
Proposed Feats v2+[30] Backbone	14.42	27.70	13.77	25.91
Proposed Feats v3+[30] Backbone	14.67	32.65	14.79	29.99
Proposed Feats v4+[30] Backbone	25.64	44.37	25.98	42.61
Proposed Feats v5+[30] Backbone	26.44	45.58	25.90	42.79
Proposed Feats v6+[30] Backbone	**26.85**	**46.92**	**26.49**	**43.67**

Table 5.14: BLIP-2 [20] Embedding Ablation With-Backbone Evaluation Results

(b) **Frame-Wise Action Category Classification:**

In the frame-wise action category classification experiments, we did not see a significant performance change in the frame-wise action classification task when we used the additional features we proposed.

Approach	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
[30] Feats+[30] Backbone	**0.01**	**0.82**	0.00	**0.23**	**0.57**	**0.29**	0.41	0.36	0.51
P. Feats v1+[30] Backbone	**0.01**	0.78	0.00	0.16	0.52	0.21	0.29	0.30	0.38
P. Feats v2+[30] Backbone	0.00	0.56	0.00	0.19	0.43	0.25	0.35	0.33	0.45
P. Feats v3+[30] Backbone	**0.01**	0.81	0.00	0.18	0.52	0.24	0.31	0.29	0.42
P. Feats v4+[30] Backbone	**0.01**	0.80	**0.01**	**0.23**	0.56	0.28	0.40	**0.37**	0.50
P. Feats v5+[30] Backbone	**0.01**	**0.82**	0.00	**0.23**	**0.57**	0.28	0.40	0.36	0.50
P. Feats v6+[30] Backbone	**0.01**	0.80	0.00	**0.23**	0.56	**0.29**	**0.42**	**0.37**	**0.52**

Table 5.15: Frame-Wise Action Category Classification - Without Backbone Evaluation Results (Validation Split, Threshold=Maximum)

Approach	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
[30] Feats+[30] Backbone	**0.01**	**0.82**	0.00	0.00	**0.36**	0.00	0.00	0.00	0.00
P. Feats v1+[30] Backbone	**0.01**	0.78	0.00	0.00	0.35	0.00	0.00	0.00	0.00
P. Feats v2+[30] Backbone	0.00	0.56	0.00	0.00	0.25	0.00	0.00	0.00	0.00
P. Feats v3+[30] Backbone	**0.01**	0.81	0.00	0.00	**0.36**	0.00	0.00	0.00	0.00
P. Feats v4+[30] Backbone	**0.01**	0.80	**0.01**	0.00	0.35	0.00	0.00	0.00	0.00
P. Feats v5+[30] Backbone	**0.01**	**0.82**	0.00	0.00	**0.36**	0.00	0.00	0.00	0.00
P. Feats v6+[30] Backbone	**0.01**	0.80	0.00	0.00	0.35	0.00	0.00	0.00	0.00

Table 5.16: Frame-Wise Action Category Classification - Without Backbone Evaluation Results (Validation Split, Threshold=1.0)

Approach	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
[30] Feats+[30] Backbone	**0.01**	**0.82**	0.00	0.00	**0.36**	0.00	0.00	0.00	0.00
P. Feats v1+[30] Backbone	**0.01**	0.78	0.00	0.00	0.35	0.00	0.00	0.00	0.00
P. Feats v2+[30] Backbone	0.00	0.56	0.00	0.00	0.25	0.00	0.00	0.00	0.00
P. Feats v3+[30] Backbone	**0.01**	0.81	0.00	0.00	**0.36**	0.00	0.00	0.00	0.00
P. Feats v4+[30] Backbone	**0.01**	0.80	**0.01**	0.00	0.35	0.00	0.00	0.00	0.00
P. Feats v5+[30] Backbone	**0.01**	**0.82**	0.00	0.00	**0.36**	0.00	0.00	0.00	0.00
P. Feats v6+[30] Backbone	**0.01**	0.80	0.00	0.00	0.35	0.00	0.00	0.00	0.00

Table 5.17: Frame-Wise Action Category Classification - Without Backbone Evaluation Results (Validation Split, Threshold=0.8)

Approach	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
[30] Feats+[30] Backbone	**0.01**	**0.82**	0.00	0.04	0.55	0.02	0.06	0.33	0.04
P. Feats v1+[30] Backbone	**0.01**	0.78	0.00	0.00	0.35	0.00	0.00	0.00	0.00
P. Feats v2+[30] Backbone	0.00	0.56	0.00	0.00	0.25	0.00	0.00	0.00	0.00
P. Feats v3+[30] Backbone	**0.01**	0.81	0.00	0.00	0.37	0.00	0.00	0.02	0.00
P. Feats v4+[30] Backbone	**0.01**	0.80	**0.01**	0.04	0.52	**0.03**	0.06	0.30	0.04
P. Feats v5+[30] Backbone	**0.01**	**0.82**	0.00	**0.05**	**0.56**	**0.03**	**0.09**	**0.36**	**0.06**
P. Feats v6+[30] Backbone	**0.01**	0.80	0.00	0.04	0.55	**0.03**	0.07	**0.36**	0.05

Table 5.18: Frame-Wise Action Category Classification - Without Backbone Evaluation Results (Validation Split, Threshold=0.6)

Approach	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
[30] Feats+[30] Backbone	**0.01**	**0.82**	0.00	**0.18**	**0.73**	**0.15**	**0.32**	**0.65**	**0.27**
P. Feats v1+[30] Backbone	0.00	0.78	0.00	0.07	0.53	0.05	0.12	0.33	0.09
P. Feats v2+[30] Backbone	0.00	0.56	0.00	0.09	0.47	0.07	0.17	0.40	0.12
P. Feats v3+[30] Backbone	**0.01**	0.81	0.00	0.06	0.62	0.05	0.11	0.47	0.09
P. Feats v4+[30] Backbone	**0.01**	0.80	**0.01**	0.16	0.59	**0.15**	0.29	0.42	0.26
P. Feats v5+[30] Backbone	**0.01**	**0.82**	0.00	**0.18**	0.62	**0.15**	0.31	0.46	**0.27**
P. Feats v6+[30] Backbone	**0.01**	0.80	0.00	**0.18**	0.69	**0.15**	**0.32**	0.60	**0.27**

Table 5.19: Frame-Wise Action Category Classification - Without Backbone Evaluation Results (Validation Split, Threshold=0.4)

Approach	B.F1	B.P	B.R	WWB.F1	WWB.P	WWB.R	WNB.F1	WNB.P	WNB.R
[30] Feats+[30] Backbone	**0.01**	**0.82**	0.00	**0.26**	0.56	**0.39**	**0.46**	0.36	**0.70**
P. Feats v1+[30] Backbone	**0.01**	0.78	0.00	0.18	0.48	0.33	0.32	0.24	0.59
P. Feats v2+[30] Backbone	0.00	0.56	0.00	0.20	0.42	0.35	0.37	0.30	0.62
P. Feats v3+[30] Backbone	**0.01**	0.81	0.00	0.21	0.52	0.35	0.37	0.29	0.63
P. Feats v4+[30] Backbone	**0.01**	0.80	**0.01**	**0.26**	0.55	**0.39**	0.45	0.35	0.69
P. Feats v5+[30] Backbone	**0.01**	**0.82**	0.00	**0.26**	**0.57**	0.38	**0.46**	**0.37**	0.67
P. Feats v6+[30] Backbone	**0.01**	0.80	0.00	**0.26**	0.55	**0.39**	**0.46**	0.36	**0.70**

Table 5.20: Frame-Wise Action Category Classification - Without Backbone Evaluation Results (Validation Split, Threshold=0.2)

Chapter 6

Discussion and Conclusion

In this work, we work on the Ego4D [14] Moment Queries Challenge, which consists of two subtasks: action detection, which is detecting segments of a given egocentric video that includes an action instance corresponding to one of the 111 pre-defined action categories, and action retrieval, which is the task of retrieving segments of a given egocentric video that includes an action instance corresponding to only the input action category. We also work on frame-wise action category classification task. Similar to the previous works ([25], [16]) that use a pre-trained vision-language model for an Ego4D [14] challenge, we also use a pre-trained vision-language model (BLIP-2 [20]) in our tasks. In our experiments, we feed frames of a given video clip to BLIP-2 [20], and we use the caption generated by BLIP-2 [20] and SBERT [28] embedding of this caption.

For each video's each frame's each BLIP-2 [20] caption, we check whether we have a match between this BLIP-2 [20] caption's (verb, noun, tool) pairs and (verb, noun, tool) pairs of this frame's ground truth action categories. We use different prompts while querying BLIP-2 [20]. We find that the ratio of frames that have a match between their captions and action categories is very low (the best noun match rate is 0.205, the best verb match rate is 0.157, the best (verb, noun) match rate is 0.064, the best (verb, noun, tool) match rate is 0.037).

We feed the SBERT [28] embeddings and the output of BLIP-2 [20]'s LLM encoder as input to the backbone of the state-of-the-art approach (ASL-Ego4D [30]) with and without the video features already being used by this approach. We evaluate our predictions for action detection and action retrieval tasks using with-backbone evaluation metrics, and we evaluate our predictions for the frame-wise action classification task using without-backbone evaluation metrics.

We get an improvement over the state-of-the-art approach (ASL-Ego4D [30]) by using SBERT [28] embedding of the BLIP-2 [20] caption and the output of the BLIP-2's [20] LLM encoder as additional features to video features already being used by the state-of-the-art approach.

We do not see a significant performance change in the frame-wise action classification task when we use the additional features we propose.

We propose three potential ways of improving the results, which are provided in chapter 7.

Chapter 7

Future Work

We provide three possible ways of improving the results:

1. One can fine-tune BLIP-2 [20] on (frame, action category phrase) pairs before extracting the (verb, noun, tool) features and the embeddings. We provide the fine-tuning script in our repository.

2. One can use VideoBLIP [38] instead of BLIP-2 [20] to extract the (verb, noun, tool) features and the embeddings. There are two main advantages of using VideoBLIP [38] instead of BLIP-2 [20]:

   - The first one is that BLIP-2 [20] takes only a single image as input and VideoBLIP [38] takes frames of a video clip as input (at most 10 at a time). In VideoBLIP [38], for a given video clip, the authors use the frozen image encoder of BLIP-2 [20] to independently encode sampled frames. The concatenation of these image encodings is fed to a Q-Former, and the output of the Q-Former is concatenated with the embedding of the input text. The rest of the pipeline is the same as the pipeline of BLIP-2 [20]. The embeddings extracted from VideoBLIP [38] will benefit from the temporal dimension.
   - The Q-Former, learnable query tokens, and fully connected neural network of VideoBLIP [38] are fine-tuned on Ego4D [14] video clip-text pairs so that the output of the language model is in the Ego4D [14] domain. These components are not fine-tuned on Ego4D [14] video clip-text pairs in the BLIP-2 [20] case.

   We provide the VideoBLIP [38] embedding extraction script in our repository.

3. In ASL-Ego4D [30], the authors use 5 different hyperparameter settings to train the ASL-Ego4D [30] backbone, and the final predictions are generated by ensembling the predictions made by each architecture. While reproducing the results of ASL-Ego4D [30] and producing our results, we do not use ensembled predictions. We use only one hyperparameter setting to train the ASL-Ego4D [30] backbone and use its predictions. One can also produce the ensembled predictions by using the combination of video features and BLIP-2 [30] embeddings. This will allow a direct comparison with the with-backbone evaluation metrics reported in ASL-Ego4D [30] paper.

Appendix A

Appendix

A.1 Action Category Exact Count Mapping

Action Category	Count
converse_/_interact_with_someone	2506
use_phone	1500
browse_through_clothing_items_on_rack_/_shelf_/_hanger	613
read_a_book_/_magazine_/_shopping_list_etc.	582
look_at_clothes_in_the_mirror	343
put_away_(or_take_out)_food_items_in_the_fridge	336
cut_/_chop_/_slice_a_vegetable,_fruit,_or_meat	312
throw_away_trash_/_put_trash_in_trash_can	294
stir_/_mix_food_while_cooking	294
walk_down_stairs_/_walk_up_stairs	292
clean_/_wipe_other_surface_or_object	279
knead_/_shape_/_roll-out_dough	272
browse_through_groceries_or_food_items_on_rack_/_shelf	267
wash_dishes_/_utensils_/_bakeware_etc.	264
dig_or_till_the_soil_with_a_hoe_or_other_tool	258
hang_clothes_in_closet_/_on_hangers	249
drink_beverage	248
arrange_/_organize_other_items	248
wash_hands	217
fix_other_item	212
use_hammer_/_nail-gun_to_fix_nail	193
watch_television	192
place_items_in_shopping_cart	182
enter_a_supermarket_/_shop	180
remove_weeds_from_ground	179
take_photo_/_record_video_with_a_camera	178
browse_through_other_items_on_rack_/_shelf	177
fold_clothes_/_sheets	172
put_away_(or_take_out)_dishes_/_utensils_in_storage	168

Table A.1: Action Category Counts Part 1

Action Category	Count
clean_/_sweep_floor_with_broom	167
play_board_game_or_card_game	160
fill_a_pot_/_bottle_/_container_with_water	155
use_a_laptop_/_computer	154
put_on_safety_equipment_(e.g._gloves,_helmet,_safety_goggles)	145
paint_using_paint_brush_/_roller	145
serve_food_onto_a_plate	142
cut_/_trim_grass_with_other_tools	135
pack_food_items_/_groceries_into_bags_/_boxes	131
try-out_/_wear_clothing_items_(e.g._shirt,_jeans,_sweater)	128
stir_/_mix_ingredients_in_a_bowl_or_pan_(before_cooking)	127
stand_in_the_queue_/_line_at_a_shop_/_supermarket	117
trim_hedges_or_branches	110
clean_/_wipe_a_table_or_kitchen_counter	110
exit_a_supermarket_/_shop	109
drill_into_wall_/_wood_/_floor_/_metal	107
turn-on_/_light_the_stove_burner	107
interact_or_play_with_pet_/_animal	106
peel_a_fruit_or_vegetable	105
cut_other_item_using_tool	104
pay_at_billing_counter	103
plaster_wall_/_surface	102
hang_clothes_to_dry	101
pack_other_items_into_bags_/_boxes	101
fix_/_remove_/_replace_a_tire_or_wheel	100
play_a_video_game	100
climb_up_/_down_a_ladder	99
cut_tree_branch	98
do_some_exercise	94
chop_/_cut_wood_pieces_using_tool	93
water_soil_/_plants_/_crops	92
load_/_unload_a_washing_machine_or_dryer	92
eat_a_snack	90
wash_vegetable_/_fruit_/_food_item	89
cut_dough	87
weigh_food_/_ingredient_using_a_weighing_scale	86
put_away_(or_take_out)_ingredients_in_storage	85
cut_/_trim_grass_with_a_lawnmower	84
fry_dough	79
withdraw_money_from_atm_/_operate_atm	77
level_ground_/_soil_(eg._using_rake,_shovel,_etc)	75
use_a_vacuum_cleaner_to_clean	74

Table A.2: Action Category Counts Part 2

Action Category	Count
tie_up_branches_/_plants_with_string	73
collect_/_rake_dry_leaves_on_ground	73
pack_soil_into_the_ground_or_a_pot_/_container	70
fix_bonnet_/_engine_of_car	67
fix_wiring	66
count_money_before_paying	66
arrange_/_organize_clothes_in_closet/dresser	65
try-out_/_wear_accessories_(e.g._tie,_belt,_scarf)	63
browse_through_accessories_on_rack_/_shelf	63
prepare_or_apply_cement_/_concrete_/_mortar	62
clean_/_wipe_kitchen_appliance	62
harvest_vegetables_/_fruits_/_crops_from_plants_on_the_ground	60
compare_two_clothing_items	60
fix_pipe_/_plumbing	57
make_the_bed_/_arrange_pillows,_sheets_etc._on_bed	55
cut_thread_/_paper_/_cardboard_using_scissors_/_knife_/_cutter	55
iron_clothes_or_sheets	55
move_/_shift_around_construction_material	55
smooth_wood_using_sandpaper_/_sander_/_tool	54
measure_wooden_item_using_tape_/_ruler	54
drive_a_vehicle	53
move_/_shift_/_arrange_small_tools	52
plant_seeds_/_plants_/_flowers_into_ground	52
arrange_/_organize_items_in_fridge	51
clean_/_repair_small_equipment_(mower,_trimmer_etc.)	50
mark_item_with_pencil_/_pen_/_marker	49
make_coffee_or_tea_/_use_a_coffee_machine	49
clean_/_wipe_/_oil_metallic_item	47
dig_or_till_the_soil_by_hand	45
fry_other_food_item	44
arrange_pillows_on_couch_/_chair	44
rinse_/_drain_other_food_item_in_sieve_/_colander	43
cut_open_a_package_(e.g._with_scissors)	43
write_notes_in_a_paper_/_book	43
smoke_cigar_/_cigarette_/_vape	42
dismantle_other_item	42
taste_food_while_cooking	39
remove_food_from_the_oven	32
put_food_into_the_oven_to_bake	28

Table A.3: Action Category Counts Part 3

A.2 Dependency Parsing Algorithm

```
def dependency_parsing(
    blip2_answer: str,
    dependency_parser: CoreNLPDependencyParser,
):
    if blip2_answer == "" or pd.isnull(blip2_answer):
        return []

    blip2_answer_clean = ""

    for char in blip2_answer:
        if char.isalpha() or char in [" ", "'"]:
            blip2_answer_clean += char
    blip2_answer_clean_splitted = blip2_answer_clean.split()

    verb_noun_tool_pairs = []

    try:
        dependency_parser_results = dependency_parser.parse(
                                                    blip2_answer_clean_splitted)
        dependency_parser_result = dependency_parser_results.__next__()
    except:
        return []

    using_addresses = []
    for current_word_address in dependency_parser_result.nodes.keys():
        current_word_node = dependency_parser_result.nodes[current_word_address]
        if current_word_node["word"] == "using":
            using_addresses.append(current_word_address)

    for current_word_address in dependency_parser_result.nodes.keys():
        current_word_node = dependency_parser_result.nodes[current_word_address]
        current_word_pos_tag = current_word_node["tag"]

        # We are only interested in sentences which include a verb. We extract nouns
                                                    and tools that are dependent on the
                                                     verb.
        if current_word_pos_tag.startswith("VB"):
            verb_word = current_word_node["word"]

            # If current word is "using" and the word before the current word is not "
                                                        is", then continue.
            if current_word_address > 0 and verb_word == "using":
                previous_word_address = current_word_address - 1
                previous_word_node = dependency_parser_result.nodes[
                    previous_word_address
                ]
                verb_previous_word = previous_word_node["word"]
                if verb_previous_word != "is":
                    continue

            verb_lemma = current_word_node["lemma"]

            # If the lemma of the verb is one of the following verb lemmas, then
                                                        continue.
            if verb_lemma in ["be", "describe", "show", "have", "tell"]:
                continue
```

```
if "compound:prt" in current_word_node["deps"].keys():
    verb_compound_prt_address = current_word_node["deps"][
        "compound:prt"
    ][0]
    verb_compound_prt_node = dependency_parser_result.nodes[
        verb_compound_prt_address
    ]
    verb_compound_prt_lemma = verb_compound_prt_node["lemma"]
    verb_lemma = verb_lemma + " " + verb_compound_prt_lemma
verb_lemma = verb_lemma.replace("-", " ")

# Extract tools that are related to the verb with a "with".
tool_addresses = current_word_node["deps"].get("obl", [])
filtered_tool_addresses = set()
for tool_address in tool_addresses:
    tool_node = dependency_parser_result.nodes[tool_address]
    tool_case_addresses = tool_node["deps"].get("case", [])
    for tool_case_address in tool_case_addresses:
        tool_case_node = dependency_parser_result.nodes[
            tool_case_address
        ]
        if tool_case_node["lemma"] == "with":
            filtered_tool_addresses.add(tool_address)

if verb_lemma != "use":
    # Extract tools that are related to the verb with a "using".
    for using_address in using_addresses:
        using_node = dependency_parser_result.nodes[using_address]
        for using_obj_address in using_node["deps"].get("obj", []):
            filtered_tool_addresses.add(using_obj_address)

        for using_acl_address in using_node["deps"].get("acl", []):
            using_acl_node = dependency_parser_result[using_acl_address]
            for using_acl_obj_address in using_acl_node["deps"].get(
                "obj", []
            ):
                filtered_tool_addresses.add(using_acl_obj_address)

# Extract noun addresses.
noun_addresses = current_word_node["deps"].get(
    "obj", []
) + current_word_node["deps"].get("nsubj:pass", [])

# Extract noun lemmas.
noun_lemmas = []
for noun_address in noun_addresses:
    noun_node = dependency_parser_result.nodes[noun_address]
    noun_lemma = noun_node["lemma"]
    if "compound" in noun_node["deps"]:
        noun_compound_address = noun_node["deps"]["compound"][0]
        noun_compound_node = dependency_parser_result.nodes[
            noun_compound_address
        ]
        noun_compound_lemma = noun_compound_node["lemma"]
        noun_lemma = noun_compound_lemma + " " + noun_lemma
    noun_lemma = noun_lemma.replace("-", " ")
    noun_lemmas.append(noun_lemma)
```

```
            # Extract tool lemmas.
            tool_lemmas = []

            for tool_address in filtered_tool_addresses:
                tool_node = dependency_parser_result.nodes[tool_address]
                tool_lemma = tool_node["lemma"]
                if "compound" in tool_node["deps"]:
                    tool_compound_address = tool_node["deps"]["compound"][0]
                    tool_compound_node = dependency_parser_result.nodes[
                        tool_compound_address
                    ]
                    tool_compound_lemma = tool_compound_node["lemma"]
                    tool_lemma = tool_compound_lemma + " " + tool_lemma
                tool_lemma = tool_lemma.replace("-", " ")
                tool_lemmas.append(tool_lemma)

            if len(noun_lemmas) == 0:
                noun_lemmas = ["NaN"]
            if len(tool_lemmas) == 0:
                tool_lemmas = ["NaN"]

            # For the current verb, and for each associated noun lemma and tool
            # lemma combination, add them to verb_noun_tool_pairs.
            for noun_lemma, tool_lemma in itertools.product(
                noun_lemmas, tool_lemmas
            ):
                verb_noun_tool_pairs.append((verb_lemma, noun_lemma, tool_lemma))

        elif current_word_pos_tag.startswith("NN"):
            noun_lemma = current_word_node["lemma"]
            if "compound" in current_word_node["deps"]:
                noun_compound_address = current_word_node["deps"]["compound"][0]
                noun_compound_node = dependency_parser_result.nodes[
                    noun_compound_address
                ]
                noun_compound_lemma = noun_compound_node["lemma"]
                noun_lemma = noun_compound_lemma + " " + noun_lemma

            verb_noun_tool_pairs.append(("NaN", noun_lemma, "NaN"))

    return verb_noun_tool_pairs
```

A.3 Matching-Based Score Calculation Algorithm

```
def matching_based_score_calculation(
    action_category_verb_noun_tool_pairs: List[Tuple[str, str, str]],
    caption_verb_noun_tool_pairs: List[Tuple[str, str, str]]
    ):
    max_score = 0.0
    for action_category_verb_noun_tool_pair in action_category_verb_noun_tool_pairs:
        for caption_verb_noun_tool_pair in caption_verb_noun_tool_pairs:
            action_category_verb = action_category_verb_noun_tool_pair[0]
            action_category_noun = action_category_verb_noun_tool_pair[1]
            action_category_tool = action_category_verb_noun_tool_pair[2]
            caption_verb = caption_verb_noun_tool_pair[0]
            caption_noun = caption_verb_noun_tool_pair[1]
            caption_tool = caption_verb_noun_tool_pair[2]
            if action_category_verb == caption_verb and
               action_category_noun == caption_noun and
               action_category_tool == caption_tool:
                if max_score < 1.0:
                    max_score = 1.0
            elif action_category_verb == caption_verb and
                 action_category_noun == caption_noun:
                if max_score < 0.8:
                    max_score = 0.8
            elif action_category_verb == caption_verb and
                 action_category_tool == caption_tool:
                if max_score < 0.6:
                    max_score = 0.6
            elif action_category_noun == caption_noun:
                if max_score < 0.4:
                    max_score = 0.4
            elif action_category_verb == caption_verb:
                if max_score < 0.2:
                    max_score = 0.2
    return max_score
```

A.4 Manually Extracted Action Category Verb-Noun-Tool Pairs


```
{
    "\"clean_/_repair_small_equipment_(mower,_trimmer_etc.)\"": [
        ["clean", "equipment", "NaN"],
        ["clean", "mower", "NaN"],
        ["clean", "trimmer", "NaN"],
        ["clean", "small equipment", "NaN"],
        ["clean", "item", "NaN"],
        ["clean", "small item", "NaN"],

        ["repair", "equipment", "NaN"],
        ["repair", "mower", "NaN"],
        ["repair", "trimmer", "NaN"],
        ["repair", "small equipment", "NaN"],
        ["repair", "item", "NaN"],
        ["repair", "small item", "NaN"]
    ],
    "\"cut_/_chop_/_slice_a_vegetable,_fruit,_or_meat\"": [
        ["cut", "vegetable", "NaN"],
        ["cut", "fruit", "NaN"],
        ["cut", "meat", "NaN"],

        ["chop", "vegetable", "NaN"],
        ["chop", "fruit", "NaN"],
        ["chop", "meat", "NaN"],

        ["slice", "vegetable", "NaN"],
        ["slice", "fruit", "NaN"],
        ["slice", "meat", "NaN"]
    ],
    "\"level_ground_/_soil_(eg._using_rake,_shovel,_etc)\"": [
        ["rake", "ground", "NaN"],
        ["shovel", "ground", "NaN"],

        ["level", "ground", "rake"],
        ["level", "ground", "shovel"],
        ["level", "soil", "rake"],
        ["level", "soil", "shovel"],

        ["use", "shovel", "NaN"]
    ],
    "\"make_the_bed_/_arrange_pillows,_sheets_etc._on_bed\"": [
        ["make", "bed", "NaN"],
        ["arrange", "bed", "NaN"],
        ["arrange", "pillow", "NaN"],
        ["arrange", "sheet", "NaN"]
    ],
    "\"put_on_safety_equipment_(e.g._gloves,_helmet,_safety_goggles)\"": [
        ["wear", "safety equipment", "NaN"],
        ["wear", "glove", "NaN"],
        ["wear", "helmet", "NaN"],
        ["wear", "goggles", "NaN"],
        ["wear", "safety goggles", "NaN"],

        ["put on", "safety equipment", "NaN"],
        ["put on", "glove", "NaN"],
        ["put on", "helmet", "NaN"],
```

```
        ["put on", "goggles", "NaN"],
        ["put on", "safety goggles", "NaN"],

        ["try on", "safety equipment", "NaN"],
        ["try on", "glove", "NaN"],
        ["try on", "helmet", "NaN"],
        ["try on", "goggles", "NaN"],
        ["try on", "safety goggles", "NaN"],

        ["try out", "safety equipment", "NaN"],
        ["try out", "glove", "NaN"],
        ["try out", "helmet", "NaN"],
        ["try out", "goggles", "NaN"],
        ["try out", "safety goggles", "NaN"]
    ],
    "\"try-out_/_wear_accessories_(e.g._tie,_belt,_scarf)\"": [
        ["wear", "accessory", "NaN"],
        ["wear", "tie", "NaN"],
        ["wear", "scarf", "NaN"],
        ["wear", "belt", "NaN"],

        ["put on", "accessory", "NaN"],
        ["put on", "tie", "NaN"],
        ["put on", "scarf", "NaN"],
        ["put on", "belt", "NaN"],

        ["try on", "accessory", "NaN"],
        ["try on", "tie", "NaN"],
        ["try on", "scarf", "NaN"],
        ["try on", "belt", "NaN"],

        ["try out", "accessory", "NaN"],
        ["try out", "tie", "NaN"],
        ["try out", "scarf", "NaN"],
        ["try out", "belt", "NaN"]
    ],
    "\"try-out_/_wear_clothing_items_(e.g._shirt,_jeans,_sweater)\"": [
        ["wear", "cloth", "NaN"],
        ["wear", "clothing item", "NaN"],
        ["wear", "shirt", "NaN"],
        ["wear", "jeans", "NaN"],
        ["wear", "sweater", "NaN"],

        ["put on", "cloth", "NaN"],
        ["put on", "clothing item", "NaN"],
        ["put on", "shirt", "NaN"],
        ["put on", "jeans", "NaN"],
        ["put on", "sweater", "NaN"],

        ["try on", "cloth", "NaN"],
        ["try on", "clothing item", "NaN"],
        ["try on", "shirt", "NaN"],
        ["try on", "jeans", "NaN"],
        ["try on", "sweater", "NaN"],

        ["try out", "cloth", "NaN"],
        ["try out", "clothing item", "NaN"],
        ["try out", "shirt", "NaN"],
        ["try out", "jeans", "NaN"],
```

```
        ["try out", "sweater", "NaN"]
    ],
    "arrange_/_organize_clothes_in_closet/dresser": [
        ["arrange", "cloth", "NaN"],
        ["arrange", "closet", "NaN"],
        ["arrange", "dresser", "NaN"],

        ["organize", "cloth", "NaN"],
        ["organize", "closet", "NaN"],
        ["organize", "dresser", "NaN"]
    ],
    "arrange_/_organize_items_in_fridge": [
        ["arrange", "fridge", "NaN"],
        ["arrange", "kitchen", "NaN"],

        ["organize", "fridge", "NaN"],
        ["organize", "kitchen", "NaN"]
    ],
    "arrange_/_organize_other_items": [
        ["arrange", "item", "NaN"],
        ["organize", "item", "NaN"]
    ],
    "arrange_pillows_on_couch_/_chair": [
        ["arrange", "pillow", "NaN"],
        ["arrange", "couch", "NaN"],
        ["arrange", "chair", "NaN"],

        ["organize", "pillow", "NaN"],
        ["organize", "couch", "NaN"],
        ["organize", "chair", "NaN"]
    ],
    "browse_through_accessories_on_rack_/_shelf": [
        ["browse", "accessory", "NaN"],
        ["look", "accessory", "NaN"]
    ],
    "browse_through_clothing_items_on_rack_/_shelf_/_hanger": [
        ["browse", "cloth", "NaN"],
        ["browse", "clothing item", "NaN"]
    ],
    "browse_through_groceries_or_food_items_on_rack_/_shelf": [
        ["browse", "grocery", "NaN"],
        ["browse", "food", "NaN"],
        ["browse", "food item", "NaN"]
    ],
    "browse_through_other_items_on_rack_/_shelf": [["browse", "item", "NaN"]],
    "chop_/_cut_wood_pieces_using_tool": [
        ["chop", "wood", "tool"],
        ["cut", "wood", "tool"]
    ],
    "clean_/_sweep_floor_with_broom": [
        ["clean", "floor", "broom"],
        ["sweep", "floor", "broom"]
    ],
    "clean_/_wipe_/_oil_metallic_item": [
        ["clean", "metal", "NaN"],
        ["clean", "metallic item", "NaN"],
        ["wipe", "metal", "NaN"],
        ["wipe", "metallic item", "NaN"],
        ["oil", "metal", "NaN"],
```

```
        ["oil", "metallic item", "NaN"]
    ],
    "clean_/_wipe_a_table_or_kitchen_counter": [
        ["clean", "table", "NaN"],
        ["clean", "kitchen counter", "NaN"],
        ["clean", "counter", "NaN"],
        ["wipe", "table", "NaN"],
        ["wipe", "counter", "NaN"],
        ["wipe", "kitchen counter", "NaN"]
    ],
    "clean_/_wipe_kitchen_appliance": [
        ["clean", "kitchen appliance", "NaN"],
        ["wipe", "kitchen appliance", "NaN"]
    ],
    "clean_/_wipe_other_surface_or_object": [
        ["clean", "surface", "NaN"],
        ["wipe", "surface", "NaN"],
        ["clean", "object", "NaN"],
        ["wipe", "object", "NaN"]
    ],
    "climb_up_/_down_a_ladder": [
        ["climb up", "ladder", "NaN"],
        ["climb down", "ladder", "NaN"],
        ["climb", "ladder", "NaN"]
    ],
    "collect_/_rake_dry_leaves_on_ground": [
        ["collect", "leave", "NaN"],
        ["rake", "leave", "NaN"]
    ],
    "compare_two_clothing_items": [
        ["compare", "cloth", "NaN"],
        ["compare", "clothing item", "NaN"]
    ],
    "converse_/_interact_with_someone": [
        ["converse", "someone", "NaN"],
        ["interact", "someone", "NaN"],
        ["speak", "someone", "NaN"],
        ["talk", "someone", "NaN"]
    ],
    "count_money_before_paying": [["count", "money", "NaN"]],
    "cut_/_trim_grass_with_a_lawnmower": [
        ["cut", "grass", "lawnmower"],
        ["trim", "grass", "lawnmower"]
    ],
    "cut_/_trim_grass_with_other_tools": [
        ["cut", "grass", "tool"],
        ["trim", "grass", "tool"]
    ],
    "cut_dough": [["cut", "dough", "NaN"]],
    "cut_open_a_package_(e.g._with_scissors)": [
        ["cut", "package", "scissors"],
        ["open", "package", "scissors"]
    ],
    "cut_other_item_using_tool": [["cut", "item", "tool"]],
    "cut_thread_/_paper_/_cardboard_using_scissors_/_knife_/_cutter": [
        ["cut", "thread", "scissors"],
        ["cut", "thread", "knife"],
        ["cut", "thread", "cutter"],
        ["cut", "paper", "scissors"],
```

```
        ["cut", "paper", "knife"],
        ["cut", "paper", "cutter"],
        ["cut", "cardboard", "scissors"],
        ["cut", "cardboard", "knife"],
        ["cut", "cardboard", "cutter"],
        ["use", "cutter", "NaN"],
        ["use", "knife", "NaN"]
    ],
    "cut_tree_branch": [
        ["cut", "tree", "NaN"],
        ["cut", "branch", "NaN"],
        ["cut", "tree branch", "NaN"]
    ],
    "dig_or_till_the_soil_by_hand": [
        ["dig", "soil", "hand"],
        ["till", "soil", "hand"]
    ],
    "dig_or_till_the_soil_with_a_hoe_or_other_tool": [
        ["dig", "soil", "hoe"],
        ["till", "soil", "hoe"],
        ["dig", "soil", "tool"],
        ["till", "soil", "tool"]
    ],
    "dismantle_other_item": [["dismantle", "item", "NaN"]],
    "do_some_exercise": [["do", "exercise", "NaN"]],
    "drill_into_wall_/_wood_/_floor_/_metal": [
        ["drill", "wall", "NaN"],
        ["drill", "wood", "NaN"],
        ["drill", "floor", "NaN"],
        ["drill", "metal", "NaN"]
    ],
    "drink_beverage": [["drink", "beverage", "NaN"]],
    "drive_a_vehicle": [["drive", "vehicle", "NaN"]],
    "eat_a_snack": [["eat", "snack", "NaN"]],
    "enter_a_supermarket_/_shop": [
        ["enter", "supermarket", "NaN"],
        ["enter", "shop", "NaN"]
    ],
    "exit_a_supermarket_/_shop": [
        ["exit", "supermarket", "NaN"],
        ["exit", "shop", "NaN"]
    ],
    "fill_a_pot_/_bottle_/_container_with_water": [
        ["fill", "pot", "water"],
        ["fill", "bottle", "water"],
        ["fill", "container", "water"]
    ],
    "fix_/_remove_/_replace_a_tire_or_wheel": [
        ["fix", "tire", "NaN"],
        ["fix", "wheel", "NaN"],
        ["remove", "tire", "NaN"],
        ["replace", "wheel", "NaN"],
        ["replace", "tire", "NaN"]
    ],
    "fix_bonnet_/_engine_of_car": [
        ["fix", "bonnet", "NaN"],
        ["fix", "engine", "NaN"]
    ],
    "fix_other_item": [["fix", "item", "NaN"]],
```

```
    "fix_pipe_/_plumbing": [["fix", "pipe", "NaN"], ["fix", "plumbing", "NaN"]],
    "fix_wiring": [["fix", "wiring", "NaN"]],
    "fold_clothes_/_sheets": [["fold", "cloth", "NaN"], ["fold", "sheet", "NaN"]],
    "fry_dough": [["fry", "dough", "NaN"]],
    "fry_other_food_item": [["fry", "food", "NaN"], ["fry", "food item", "NaN"]],
    "hang_clothes_in_closet_/_on_hangers": [["hang", "cloth", "NaN"]],
    "hang_clothes_to_dry": [["dry", "cloth", "NaN"]],
    "harvest_vegetables_/_fruits_/_crops_from_plants_on_the_ground": [
        ["harvest", "vegetable", "NaN"],
        ["harvest", "fruit", "NaN"],
        ["harvest", "crop", "NaN"]
    ],
    "interact_or_play_with_pet_/_animal": [
        ["interact", "pet", "NaN"],
        ["interact", "animal", "NaN"],
        ["play", "pet", "NaN"],
        ["play", "animal", "NaN"]
    ],
    "iron_clothes_or_sheets": [["iron", "cloth", "NaN"], ["iron", "sheet", "NaN"]],
    "knead_/_shape_/_roll-out_dough": [
        ["knead", "dough", "NaN"],
        ["shape", "dough", "NaN"],
        ["roll out", "dough", "NaN"]
    ],
    "load_/_unload_a_washing_machine_or_dryer": [
        ["load", "washing machine", "NaN"],
        ["load", "dryer", "NaN"],
        ["unload", "washing machine", "NaN"],
        ["unload", "dryer", "NaN"]
    ],
    "look_at_clothes_in_the_mirror": [["look", "cloth", "NaN"]],
    "make_coffee_or_tea_/_use_a_coffee_machine": [
        ["make", "coffee", "NaN"],
        ["use", "coffee machine", "NaN"],
        ["make", "tea", "NaN"]
    ],
    "mark_item_with_pencil_/_pen_/_marker": [
        ["mark", "item", "pencil"],
        ["mark", "item", "pen"],
        ["mark", "item", "marker"]
    ],
    "measure_wooden_item_using_tape_/_ruler": [
        ["measure", "wooden", "tape"],
        ["measure", "wooden", "ruler"],
        ["measure", "wooden item", "tape"],
        ["measure", "wooden item", "tape"]
    ],
    "move_/_shift_/_arrange_small_tools": [
        ["move", "small tool", "NaN"],
        ["shift", "small tool", "NaN"],
        ["arrange", "small tool", "NaN"]
    ],
    "move_/_shift_around_construction_material": [
        ["move", "construction material", "NaN"],
        ["shift", "construction material", "NaN"]
    ],
    "pack_food_items_/_groceries_into_bags_/_boxes": [
        ["pack", "food", "NaN"],
        ["pack", "food item", "NaN"],
```

```
        ["pack", "grocery", "NaN"]
    ],
    "pack_other_items_into_bags_/_boxes": [["pack", "item", "NaN"]],
    "pack_soil_into_the_ground_or_a_pot_/_container": [["pack", "soil", "NaN"]],
    "paint_using_paint_brush_/_roller": [
        ["paint", "NaN", "brush"],
        ["paint", "NaN", "roller"],
        ["paint", "NaN", "paint brush"]
    ],
    "pay_at_billing_counter": [["pay", "NaN", "NaN"]],
    "peel_a_fruit_or_vegetable": [
        ["peel", "fruit", "NaN"],
        ["peel", "vegetable", "NaN"]
    ],
    "place_items_in_shopping_cart": [["place", "item", "NaN"]],
    "plant_seeds_/_plants_/_flowers_into_ground": [
        ["plant", "seed", "NaN"],
        ["plant", "plant", "NaN"],
        ["plant", "flower", "NaN"]
    ],
    "plaster_wall_/_surface": [
        ["plaster", "wall", "NaN"],
        ["plaster", "surface", "NaN"]
    ],
    "play_a_video_game": [
        ["play", "video game", "NaN"],
        ["play", "computer game", "NaN"]
    ],
    "play_board_game_or_card_game": [
        ["play", "board game", "NaN"],
        ["play", "card game", "NaN"]
    ],
    "prepare_or_apply_cement_/_concrete_/_mortar": [
        ["prepare", "cement", "NaN"],
        ["apply", "cement", "NaN"],
        ["prepare", "concrete", "NaN"],
        ["apply", "concrete", "NaN"],
        ["prepare", "mortar", "NaN"],
        ["apply", "mortar", "NaN"]
    ],
    "put_away_(or_take_out)_dishes_/_utensils_in_storage": [
        ["put away", "dish", "NaN"],
        ["put away", "utensil", "NaN"],
        ["take out", "dish", "NaN"],
        ["take out", "utensil", "NaN"]
    ],
    "put_away_(or_take_out)_food_items_in_the_fridge": [
        ["put away", "food", "NaN"],
        ["take out", "food", "NaN"],
        ["put away", "food item", "NaN"],
        ["take out", "food item", "NaN"]
    ],
    "put_away_(or_take_out)_ingredients_in_storage": [
        ["put away", "ingredient", "NaN"],
        ["take out", "ingredient", "NaN"]
    ],
    "put_food_into_the_oven_to_bake": [
        ["bake", "food", "NaN"],
        ["put", "food", "NaN"]
```

```
    ],
    "read_a_book_/_magazine_/_shopping_list_etc.": [
        ["read", "book", "NaN"],
        ["read", "magazine", "NaN"],
        ["read", "shopping list", "NaN"]
    ],
    "remove_food_from_the_oven": [["remove", "food", "NaN"]],
    "remove_weeds_from_ground": [["remove", "weed", "NaN"]],
    "rinse_/_drain_other_food_item_in_sieve_/_colander": [
        ["rinse", "food", "NaN"],
        ["rinse", "food item", "NaN"],
        ["drain", "food", "NaN"],
        ["drain", "food item", "NaN"]
    ],
    "serve_food_onto_a_plate": [["put", "food", "NaN"], ["serve", "food", "NaN"]],
    "smoke_cigar_/_cigarette_/_vape": [
        ["smoke", "cigar", "NaN"],
        ["smoke", "cigarette", "NaN"],
        ["smoke", "vape", "NaN"]
    ],
    "smooth_wood_using_sandpaper_/_sander_/_tool": [
        ["smooth", "wood", "sandpaper"],
        ["smooth", "wood", "sander"],
        ["smooth", "wood", "tool"]
    ],
    "stand_in_the_queue_/_line_at_a_shop_/_supermarket": [["stand", "NaN", "NaN"]],
    "stir_/_mix_food_while_cooking": [
        ["stir", "food", "NaN"],
        ["mix", "food", "NaN"]
    ],
    "stir_/_mix_ingredients_in_a_bowl_or_pan_(before_cooking)": [
        ["stir", "ingredient", "NaN"],
        ["mix", "ingredient", "NaN"]
    ],
    "take_photo_/_record_video_with_a_camera": [
        ["take", "photo", "camera"],
        ["record", "video", "camera"]
    ],
    "taste_food_while_cooking": [["taste", "food", "NaN"]],
    "throw_away_trash_/_put_trash_in_trash_can": [
        ["throw away", "trash", "NaN"],
        ["throw", "trash", "NaN"],
        ["put", "trash", "NaN"]
    ],
    "tie_up_branches_/_plants_with_string": [
        ["tie", "branch", "string"],
        ["tie", "plant", "string"],
        ["tie up", "branch", "string"],
        ["tie up", "plant", "string"]
    ],
    "trim_hedges_or_branches": [
        ["trim", "hedge", "NaN"],
        ["trim", "branch", "NaN"]
    ],
    "turn-on_/_light_the_stove_burner": [
        ["turn on", "stove burner", "NaN"],
        ["light", "stove burner", "NaN"]
    ],
    "use_a_laptop_/_computer": [
```

```
        ["use", "laptop", "NaN"],
        ["use", "computer", "NaN"]
    ],
    "use_a_vacuum_cleaner_to_clean": [
        ["use", "cleaner", "NaN"],
        ["use", "vacuum cleaner", "NaN"],
        ["clean", "floor", "vacuum cleaner"],
        ["clean", "floor", "cleaner"],
        ["vacuum", "floor", "NaN"]
    ],
    "use_hammer_/_nail-gun_to_fix_nail": [
        ["fix", "nail", "hammer"],
        ["fix", "nail", "nail gun"],
        ["use", "hammer", "NaN"],
        ["use", "nail gun", "NaN"]
    ],
    "use_phone": [["use", "phone", "NaN"]],
    "walk_down_stairs_/_walk_up_stairs": [
        ["walk", "stair", "NaN"],
        ["walk up", "stair", "NaN"],
        ["walk down", "stair", "NaN"],
        ["climb", "stair", "NaN"],
        ["climb up", "stair", "NaN"],
        ["climb down", "stair", "NaN"]
    ],
    "wash_dishes_/_utensils_/_bakeware_etc.": [
        ["wash", "dish", "NaN"],
        ["wash", "utensil", "NaN"],
        ["wash", "bakeware", "NaN"]
    ],
    "wash_hands": [["wash", "hand", "NaN"]],
    "wash_vegetable_/_fruit_/_food_item": [
        ["wash", "vegetable", "NaN"],
        ["wash", "fruit", "NaN"],
        ["wash", "food", "NaN"],
        ["wash", "food item", "NaN"]
    ],
    "watch_television": [["watch", "television", "NaN"]],
    "water_soil_/_plants_/_crops": [
        ["water", "soil", "NaN"],
        ["water", "plant", "NaN"],
        ["water", "crop", "NaN"]
    ],
    "weigh_food_/_ingredient_using_a_weighing_scale": [
        ["weigh", "food", "weighing scale"],
        ["weigh", "ingredient", "weighing scale"]
    ],
    "withdraw_money_from_atm_/_operate_atm": [["withdraw", "money", "NaN"]],
    "write_notes_in_a_paper_/_book": [
        ["write", "note", "NaN"],
        ["take", "note", "NaN"]
    ]
}
```

A.5 Manually Extracted Action Category Phrase Pairs


```
{

    "clean_/_repair_small_equipment_(mower,_trimmer_etc.)":
        ["cleaning small equipment",
         "cleaning mower",
         "cleaning trimmer",
         "repairing small equipment",
         "repairing mower",
         "repairing trimmer"],

    "cut_/_chop_/_slice_a_vegetable,_fruit,_or_meat":
        ["cutting vegetable",
         "cutting fruit",
         "cutting meat",
         "chopping vegetable",
         "chopping fruit",
         "chopping meat",
         "slicing vegetable",
         "slicing fruit",
         "slicing meat"],

    "level_ground_/_soil_(eg._using_rake,_shovel,_etc)":
        ["leveling ground using rake",
         "leveling soil using rake",
         "leveling ground using shovel",
         "leveling soil using shovel"],

    "make_the_bed_/_arrange_pillows,_sheets_etc._on_bed":
        ["making the bed",
         "arranging pillows on bed",
         "arranging sheets on bed"],

    "put_on_safety_equipment_(e.g._gloves,_helmet,_safety_goggles)":
        ["putting on safety equipment",
         "putting on gloves",
         "putting on helmet",
         "putting on safety goggles"],

    "try-out_/_wear_accessories_(e.g._tie,_belt,_scarf)":
       ["trying out accessories",
        "trying out tie",
        "trying out belt",
        "trying out scarf",
        "wearing accessories",
        "wearing tie",
        "wearing belt",
        "wearing scarf"],

    "try-out_/_wear_clothing_items_(e.g._shirt,_jeans,_sweater)":
       ["trying out clothing items",
        "trying out shirt",
        "trying out jeans",
        "trying out sweater",
        "wearing clothing items",
        "wearing shirt",
        "wearing jeans",
```

```
    "wearing sweater"],

"arrange_/_organize_clothes_in_closet/dresser":
    ["arranging clothes in closet",
     "arranging clothes in dresser",
     "organizing clothes in closet",
     "organizing clothes in dresser"],

"arrange_/_organize_items_in_fridge":
    ["arranging items in fridge",
     "organizing items in fridge"],

"arrange_/_organize_other_items":
    ["arranging items that are not in fridge, closet or dresser and that are not
                                                on couch or chair",
     "organizing items that are not in fridge, closet or dresser and that are not
                                                 on couch or chair"],

"arrange_pillows_on_couch_/_chair":
    ["arranging pillows on couch",
     "arranging pillows on chair"],

"browse_through_accessories_on_rack_/_shelf":
   ["browsing through accessories on rack",
    "browsing through accessories on shelf"],

"browse_through_clothing_items_on_rack_/_shelf_/_hanger":

    ["browsing through clothing items on rack",
     "browsing through clothing items on shelf",
     "browsing through clothing items on hanger"],

"browse_through_groceries_or_food_items_on_rack_/_shelf":

    ["browsing through groceries on rack",
     "browsing through food items on rack",
     "browsing through groceries on shelf",
     "browsing through food items on shelf"],

"browse_through_other_items_on_rack_/_shelf":
    ["browsing through items on rack which are not accessories, clothing items,
                                                groceries or food items",
     "browsing through items on shelf which are not accessories, clothing items,
                                                 groceries or food items"],

"chop_/_cut_wood_pieces_using_tool":
    ["chopping wood pieces using tool",
     "cutting wood pieces using tool"],

"clean_/_sweep_floor_with_broom":
    ["cleaning floor with broom",
     "sweeping floor with broom"],

"clean_/_wipe_/_oil_metallic_item":
    ["cleaning metallic item",
     "wiping metallic item",
     "oiling metallic item"],

"clean_/_wipe_a_table_or_kitchen_counter":
```

```
    ["cleaning table",
     "cleaning kitchen counter",
     "wiping table",
     "wiping kitchen counter"],

"clean_/_wipe_kitchen_appliance":
    ["cleaning kitchen appliance",
     "wiping kitchen appliance"],

"clean_/_wipe_other_surface_or_object":
    ["cleaning surface which is not table, kitchen counter or kitchen appliance",
     "wiping surface which is not table, kitchen counter or kitchen appliance",
     "cleaning object which is not table, kitchen counter or kitchen appliance",
     "wiping object which is not table, kitchen counter or kitchen appliance"],

"climb_up_/_down_a_ladder":
    ["climbing up a ladder",
     "climbing down a ladder"],

"collect_/_rake_dry_leaves_on_ground":
    ["collecting dry leaves on ground",
     "raking dry leaves on ground"],


"compare_two_clothing_items":
    ["comparing two clothing items"],

"converse_/_interact_with_someone":
    ["conversing with someone",
     "interacting with someone"],

"count_money_before_paying":
    ["counting money before paying"],

"cut_/_trim_grass_with_a_lawnmower":
    ["cutting grass with a lawnmower",
     "trimming grass with a lawnmower"],

"cut_/_trim_grass_with_other_tools":
    ["cutting grass with a tool other than a lawnmower",
     "trimming grass with a tool other than a lawnmower"],

"cut_dough":
    ["cutting dough"],

"cut_open_a_package_(e.g._with_scissors)":
    ["cutting a package with scissors",
     "opening a package with scissors"],

"cut_other_item_using_tool":
    ["cutting other item using tool"],

"cut_thread_/_paper_/_cardboard_using_scissors_/_knife_/_cutter":
    ["cutting thread using scissors",
     "cutting paper using scissors",
     "cutting cardboard using scissors",
     "cutting thread using knife",
     "cutting paper using knife",
     "cutting cardboard using knife",
```

```
     "cutting thread using cutter",
     "cutting paper using cutter",
     "cutting cardboard using cutter"],

"cut_tree_branch":
    ["cutting tree branch"],

"dig_or_till_the_soil_by_hand":
    ["digging the soil by hand",
     "tilling the soil by hand"],

"dig_or_till_the_soil_with_a_hoe_or_other_tool":
    ["digging the soil with a hoe",
     "tilling the soil with a hoe",
     "digging the soil with a tool",
     "tilling the soil with a tool"],

"dismantle_other_item":
    ["dismantling an item"],

"do_some_exercise":
    ["doing exercise"],

"drill_into_wall_/_wood_/_floor_/_metal":
    ["drilling into wall",
     "drilling into wood",
     "drilling into floor",
     "drilling into metal"],

"drink_beverage":
    ["drinking beverage"],

"drive_a_vehicle":
    ["driving a vehicle"],

"eat_a_snack":
    ["eating a snack"],

"enter_a_supermarket_/_shop":
    ["entering a supermarket",
     "entering a shop"],

"exit_a_supermarket_/_shop":
    ["exiting a supermarket",
     "exiting a shop"],

"fill_a_pot_/_bottle_/_container_with_water":
    ["filling a pot with water",
     "filling a bottle with water",
     "filling a container with water"],

"fix_/_remove_/_replace_a_tire_or_wheel":
    ["fixing a tire",
     "fixing a wheel",
     "removing a tire",
     "removing a wheel",
     "replacing a tire",
     "replacing a wheel"],
```

```
    "fix_bonnet_/_engine_of_car":
        ["fixing bonnet of car",
         "fixing engine of car"],

    "fix_other_item":
        ["fixing an item other than tire, wheel, bonnet of a car or engine of a car"],

    "fix_pipe_/_plumbing":
        ["fixing pipe",
         "fixing plumbing"],

    "fix_wiring":
        ["fixing wiring"],

    "fold_clothes_/_sheets":
        ["folding clothes",
         "folding sheets"],

    "fry_dough":
        ["frying dough"],

    "fry_other_food_item":
        ["frying food item other than dough"],

    "hang_clothes_in_closet_/_on_hangers":
        ["hanging clothes in closet",
         "hanging clothes on hangers"],

    "hang_clothes_to_dry":
        ["hanging clothes to dry"],

    "harvest_vegetables_/_fruits_/_crops_from_plants_on_the_ground":
        ["harvesting vegetables",
         "harvesting fruits",
         "harvesting crops from plants on the ground"],

    "interact_or_play_with_pet_/_animal":
        ["interacting with pet",
         "interacting with animal",
         "playing with pet",
         "playing with animal"],

    "iron_clothes_or_sheets":
        ["ironing clothes",
         "ironing sheets"],

    "knead_/_shape_/_roll-out_dough":
        ["kneading dough",
         "shaping dough",
         "rolling out dough"],

    "load_/_unload_a_washing_machine_or_dryer":
        ["loading a washing machine",
         "loading a dryer",
         "unloading a washing machine",
         "unloading a dryer"],

    "look_at_clothes_in_the_mirror":
        ["looking at clothes in the mirror"],
```

```
"make_coffee_or_tea_/_use_a_coffee_machine":
    ["making coffee",
     "making tea",
     "using a coffee machine"],

"mark_item_with_pencil_/_pen_/_marker":
    ["marking item with pencil",
     "marking item with pen",
     "marking item with marker"],

"measure_wooden_item_using_tape_/_ruler":
    ["measuring wooden item using tape",
     "measuring wooden item using ruler"],

"move_/_shift_/_arrange_small_tools":
    ["moving small tools",
     "shifting small tools",
     "arranging small tools"],

"move_/_shift_around_construction_material":
    ["moving around construction material",
     "shifting around construction material"],

"pack_food_items_/_groceries_into_bags_/_boxes":
    ["packing food items into bags",
     "packing food items into boxes",
     "packing groceries into bags",
     "packing groceries into boxes"],

"pack_other_items_into_bags_/_boxes":
    ["packing items that are not food items and that are not groceries into bags",
    "packing items that are not food items and that are not groceries into bags"],

"pack_soil_into_the_ground_or_a_pot_/_container":
    ["packing soil into the ground",
     "packing soil into a pot",
     "packing soil into a container"],

"paint_using_paint_brush_/_roller":
    ["painting using paint brush",
     "painting using roller"],

"pay_at_billing_counter":
    ["paying at billing counter"],

"peel_a_fruit_or_vegetable":
    ["peeling a fruit or vegetable"],

"place_items_in_shopping_cart":
    ["placing items in shopping cart"],

"plant_seeds_/_plants_/_flowers_into_ground":
    ["planting seeds into ground",
     "planting plants into ground",
     "planting flowers into ground"],

"plaster_wall_/_surface":
    ["plastering wall",
```

```
        "plastering surface"],

    "play_a_video_game":
        ["playing a video game"],

    "play_board_game_or_card_game":
        ["playing board game",
         "playing card game"],

    "prepare_or_apply_cement_/_concrete_/_mortar":
        ["preparing cement",
         "preparing concrete",
         "preparing mortar",
         "applying cement",
         "applying concrete",
         "applying mortar"],

    "put_away_(or_take_out)_dishes_/_utensils_in_storage":
        ["putting away dishes in storage",
         "putting away utensils in storage",
         "taking out dishes in storage",
         "taking out utensils in storage"],

    "put_away_(or_take_out)_food_items_in_the_fridge":
        ["putting away food items in the fridge",
         "taking out food items in the fridge"],

    "put_away_(or_take_out)_ingredients_in_storage":
        ["putting away ingredients in storage",
         "taking out ingredients in storage"],

    "put_food_into_the_oven_to_bake":
        ["putting food into the oven to bake"],

    "read_a_book_/_magazine_/_shopping_list_etc.":
        ["reading a book",
         "reading a magazine",
         "reading a shopping list"],

    "remove_food_from_the_oven":
        ["removing food from the oven"],

    "remove_weeds_from_ground":
        ["removing weeds from ground"],

    "rinse_/_drain_other_food_item_in_sieve_/_colander":
        ["rinsing food item in sieve",
         "rinsing food item in colander",
         "draining food item in sieve",
         "draining food item in colander"],

    "serve_food_onto_a_plate":
        ["serving food onto a plate"],

     "smoke_cigar_/_cigarette_/_vape":
        ["smoking cigar",
         "smoking cigarette",
         "smoking vape"],
```

```
"smooth_wood_using_sandpaper_/_sander_/_tool":
    ["smoothing wood using sandpaper",
     "smoothing wood using sander",
     "smoothing wood using tool"],

"stand_in_the_queue_/_line_at_a_shop_/_supermarket":
    ["standing in the queue at a shop",
     "standing in the line at a shop",
     "standing in the queue at a supermarket",
     "standing in the line at a supermarket"],

"stir_/_mix_food_while_cooking":
    ["stirring food while cooking",
     "mixing food while cooking"],

"stir_/_mix_ingredients_in_a_bowl_or_pan_(before_cooking)":
    ["stirring ingredients in a bowl before cooking",
     "stirring ingredients in a pan before cooking",
     "mixing ingredients in a bowl before cooking",
     "mixing ingredients in a pan before cooking"],

"take_photo_/_record_video_with_a_camera":
    ["taking photo with a camera",
     "recording video with a camera"],

"taste_food_while_cooking":
    ["tasting food while cooking"],

"throw_away_trash_/_put_trash_in_trash_can":
    ["throwing away trash in trash can",
     "putting trash in trash can"],

"tie_up_branches_/_plants_with_string":
    ["tieing up branches with string",
     "tieing up plants with string"],

"trim_hedges_or_branches":
    ["trimming hedges",
     "trimming branches"],

"turn-on_/_light_the_stove_burner":
    ["turning on the stove burner",
     "lighting the stove burner"],

"use_a_laptop_/_computer":
    ["using a laptop",
     "using a computer"],

"use_a_vacuum_cleaner_to_clean":
    ["using a vacuum cleaner to clean"],

"use_hammer_/_nail-gun_to_fix_nail":
    ["using hammer to fix nail",
     "using nail-gun to fix nail"],

"use_phone":
    ["using phone"],

"walk_down_stairs_/_walk_up_stairs":
```

```
        ["walking down stairs",
         "walking up stairs"],

    "wash_dishes_/_utensils_/_bakeware_etc.":
        ["washing dishes",
         "washing utensils",
         "washing bakeware"],

    "wash_hands":
        ["washing hands"],

    "wash_vegetable_/_fruit_/_food_item":
        ["washing vegetable",
         "washing fruit",
         "washing food item"],

    "watch_television":
        ["watching television"],

    "water_soil_/_plants_/_crops":
        ["watering soil",
         "watering plants",
         "watering crops"],

    "weigh_food_/_ingredient_using_a_weighing_scale":
        ["weighing food using a weighing scale",
         "weighing ingredient using a weighing scale"],

    "withdraw_money_from_atm_/_operate_atm":
        ["withdrawing money from atm",
         "operating atm"],

    "write_notes_in_a_paper_/_book":
        ["writing notes in a paper",
         "writing notes in a book"]
}
```

Bibliography

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding?, 2021.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] Guo Chen, Sen Xing, Zhe Chen, Yi Wang, Kunchang Li, Yizhuo Li, Yi Liu, Jiahao Wang, Yin-Dong Zheng, Bingkun Huang, Zhiyu Zhao, Junting Pan, Yifei Huang, Zun Wang, Jiashuo Yu, Yinan He, Hongjie Zhang, Tong Lu, Yali Wang, Limin Wang, and Yu Qiao. Internvideo-ego4d: A pack of champion solutions to ego4d challenges, 2022.

[5] CloudCV. Evaluating state-of-the-art in ai, 2023.

[6] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. Epic-kitchens visor benchmark: Video segmentations and object relations, 2022.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[8] Django Software Foundation. Django.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[10] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2019.

[11] Deepti Ghadiyaram, Matt Feiszli, Du Tran, Xueting Yan, Heng Wang, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition, 2019.

[12] Bernard Ghanem. Egocentric video understanding, 2023.

[13] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A single model for many visual modalities, 2022.

[14] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video, 2022.

[15] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

[16] Daoji Huang, Otmar Hilliges, Luc Van Gool, and Xi Wang. Palm: Predicting actions through language models @ ego4d long-term action anticipation challenge 2023, 2023.

[17] Matthew S. Hutchinson and Vijay N. Gadepally. Video action understanding. *IEEE Access*, 9:134611–134637, 2021.

[18] Plotly Technologies Inc. Collaborative data science, 2015.

[19] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.

[20] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[21] Kevin Qinghong Lin, Alex Jinpeng Wang, Mattia Soldan, Michael Wray, Rui Yan, Eric Zhongcong Xu, Difei Gao, Rongcheng Tu, Wenzhe Zhao, Weijie Kong, Chengfei Cai, Hongfa Wang, Dima Damen, Bernard Ghanem, Wei Liu, and Mike Zheng Shou. Egocentric video-language pretraining, 2022.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[23] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In Kalina Bontcheva and Jingbo Zhu, editors, *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[24] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[25] Razvan-George Pasca, Alexey Gavryushin, Yen-Ling Kuo, Luc Van Gool, Otmar Hilliges, and Xi Wang. Summarize the past to predict the future: Natural language descriptions of context boost multimodal object interaction, 2023.

[26] Shraman Pramanick, Yale Song, Sayan Nag, Kevin Qinghong Lin, Hardik Shah, Mike Zheng Shou, Rama Chellappa, and Pengchuan Zhang. Egovlpv2: Egocentric video-language pre-training with fusion in the backbone, 2023.

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[28] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[29] Jiayi Shao. ego4d_asl, 2024. Reproducing the MQ results. Available at https://github.com/JonnyS1226/ego4d_asl/issues/2#issuecomment-1684062480.

[30] Jiayi Shao, Xiaohan Wang, Ruijie Quan, and Yi Yang. Action sensitivity learning for the ego4d episodic memory challenge 2023, 2023.

[31] Jiayi Shao, Xiaohan Wang, Ruijie Quan, and Yi Yang. Action sensitivity learning for the ego4d episodic memory challenge 2023, 2023.

[32] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training, 2022.

[33] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks, 2019.

[34] Stanford University. Advanced evaluation metrics, section 8, cs230, 2022.

[35] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[36] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *CoRR*, abs/2202.03052, 2022.

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[38] Keunwoo Peter Yu, Zheyuan Zhang, Fengyuan Hu, and Joyce Chai. Efficient in-context learning in vision-language models for egocentric videos, 2023.

[39] Chenlin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers, 2022.

[40] Chen Zhao, Ali Thabet, and Bernard Ghanem. Video self-stitching graph network for temporal action localization, 2021.

[41] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Simple multi-dataset detection, 2022.