Generating Notifications for Missing Actions: Don't forget to turn the lights off!

Bilge Soran Ali Farhadi Linda Shapiro
University of Washington
Dept. of Computer Science and Engineering
{bilge, ali, shapiro}@cs.washington.edu

Abstract

In this paper, we propose a solution to the problem of issuing notifications on actions that are missed. This involves learning about action-interdependencies and predicting an ongoing action while segmenting the input video. For a proof of concept, we collected an egocentric latte-making dataset. We show promising results on the extremely challenging task of issuing correct and timely reminders. We also show that our model reliably segments the actions, while predicting the ongoing one when only a few frames from the beginning of the action are observed. The overall prediction accuracy is 46.2%, using 10 frames. Moreover, the overall recognition and segmentation accuracy is 72.7% when the whole activity sequence is observed. Finally, the online prediction and segmentation accuracy is 68.3%.

1. Introduction

This paper studies the problem of automatically issuing reminders about actions that users might forget. Reminders can prevent catastrophic events or might be useful for people with memory problems. We envision that reminders can eventually be helpful for complex procedures. Egocentric cameras are suitable for this, as they move with the user. For a proof of concept, we collected an egocentric dataset.

We propose a novel system, which notifies people when they forget to perform an action before their current action ends. Such a system needs to understand the ongoing action before it completes and decide whether there is a missing action or not, and if it is necessary give a notification about that. For that, it needs to extract inter-action dependencies and possess a decision mechanism that takes the importance of the missing action into account. Furthermore, predicting the ongoing actions as early as possible is needed.

2. Related Work

Egocentric activity recognition is gaining a lot of popularity, for a review please see [5]. Action prediction is a probabilistic process estimating the action in progress using partial observations [7]. Our notification system requires prediction of actions. [8] uses videos from a camera worn by a robot and *predicts the actions of the people around the robot*. To the best of our knowledge, we are the first to study action prediction in egocentric videos, where the *camera wearer's actions* are analyzed.

Coupled segmentation/recognition models usually rely on generative models (eg. [10]). [4] proposed a max-margin temporal clustering. [1] used spatio-temporal graphs, [6] achieved online parsing by using specialized grammars. [2] segmented actions by encoding the change in the states.

3. Approach

Suppose we are given the segmentation, a function $F(t)$ that for each small window L before time t giving action category ($F(t) = a_j \in A = \{a_1, a_2, ..., a_M\}$, the set of actions), and a function giving action progression, $\psi(t, F(t))$:

$$\psi(t, F(t)) = \begin{cases} -1 & if\ t\ \in B_{F(t)} \\ 0 & if\ t\ \in M_{F(t)} \\ 1 & if\ t\ \in E_{F(t)} \end{cases}$$

where B, M, E, represents the action beginning, middle and end states. Let t_+ be the next time step after t; $t_+ = t + L$. Then we can formulate a scoring function $\mathcal{Z}$ for the utility of issuing a reminder at time t_+ as:

$$\mathcal{Z}(t_+) = \Delta(t) * [\mathcal{C}(F(t_+), \mathcal{N}(F(t), F(t_+))) + \lambda * \alpha]$$
$$\Delta(t) = -\psi(t_+, F(t_+)) * \psi(t, F(t))$$

where $\mathcal{C}(a_j, a_q)$ is the cost of performing action a_q after a_j, $\mathcal{N}(a_i, a_j)$ returns the first missing action between the last completed action a_i and the predicted one a_j, $a_i, a_j, a_q \in A$. $\Delta(t)$ is a function that specifies the candidate times for issuing a reminder; these are the times a prediction is made after a completed action. α is a constant penalty for a reminder about an unnecessary action or for missing a required one, and λ is a trade off factor. The value of $\mathcal{Z}(t_+)$ at time t_+ determines if a notification should be given. In order to obtain a missing action list and compute $\mathcal{N}$ and $\mathcal{C}$, we need to extract the dependencies between actions and calculate the cost of performing one action after another. We use action orderings to model the inter-action dependencies.

3.1. Extracting Dependencies Between Actions:

A flexible ordered graph is a directed graph $G = (V, E)$, in which the vertices V represent actions and the weighted directed edges E represent ordering constraints.

Flexible Ordered Graph Construction: For constructing the graph, we use the *complete set of actions*([9]). Over

this, we compute the transition probabilities, $T(a_i, a_j)$, between actions a_i, a_j. For each $T(a_i, a_j) > 0$, an edge $e_{ij} \in E$ with a weight $w(e_{ij}) = 1/T(a_i, a_j)$ is created.

Cost of performing one action after another ($\mathcal{C}$): We calculate the cost of taking action a_q after a_j, $\mathcal{C}(a_j, a_q)$ as the min-weighted geodesic distance from a_j to a_q on the G:

$$\mathcal{C}_q^j = \mathcal{C}(a_j, a_q) = min_P(\sum_{e \in P} w(e))$$

where $P(a_j, a_q)$ is *any* path from a_j to a_q, and $e \in P$. The path corresponding to $\mathcal{C}(a_j, a_q)$ is represented as P_{jq}^*.

Determining missing actions ($\mathcal{N}$): Assume that the most recently completed action is a_i, current action is a_j, and the first action on the path P_{ij}^* is a_q. Then, if $\mathcal{C}_q^j$ is high enough, the actions on P_{ij}^* are reported as missing.

3.2. Coupled Prediction and Segmentation Module

F, ψ requires segmentation, recognition and prediction.

Segmentation Using Action Part Classifiers: For an unsegmented video, we propose a discriminative HMM (Figure 1), inferring the past (recognition) and the current (prediction) actions, while segmenting the activity. Moreover, it provides the current progress of the predicted action: beginning, end or middle. In this model, the states are different action parts belonging to all action categories, which are connected according to their reachability. The observation probabilities come from the action part classifiers.

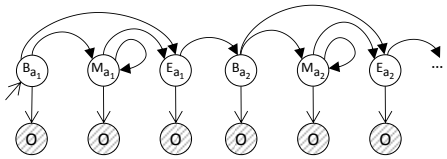

Figure 1: HMM model for segmentation and prediction.

Action Part Classifiers: For training *action part* classifiers, we partition the action samples in the training data into fixed length of L parts: beginning, middle and end.

Inference: We solve recognition, segmentation and prediction using the Viterbi algorithm. A test video is divided into parts and fed into the part classifiers for observations.

4. Experiments

Dataset: We collected a dataset, composed of 41 videos of about 20 subjects. We use 23 complete samples, where there is no missing action as our fixed part of training data and use 18 samples with missing actions in a leave-one-out-cross-validation fashion. We use GIST as our features and extend it to video part representation using bag of words.

4.1. Evaluation

Notification Module: Giving the correct reminders on time is extremely challenging. Figure 2 shows the precision-recall curve for the correctness and timeliness of notifications. As a baseline, we generated random notifications by making decision at the end of every action part, 10 times, getting an avg. precision of 0.0007 and recall of 0.1.

Segmentation and Prediction Model: We explored our model in different settings (Table 1), for a full evaluation see [9]. Row 1 shows coupled prediction and segmentation results. Row 2 shows the results of online prediction, where a prediction is made at each time step. Finally, we use the whole sequence to recognize and segment actions (row 3).

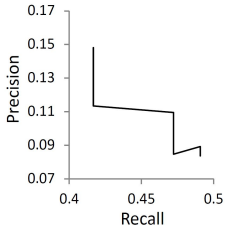

Figure 2: Evaluation of timeliness & correctness

	Avg. Acc.	Avg. per Class Acc.
Prediction	46.19	44.06
Online Prediction	68.32	56.18
Recognition	72.66	63.39

Table 1: Accuracy of our model in different settings.

We do prediction in unsegmented videos. For a comparison to max-margin early event detectors [3], see Table 2.

	Precision	Recall	F measure
MMED	0.32	0.25	0.25
Our Model	0.62	0.57	0.58

Table 2: Our model vs MMED.

5. Conclusion

In this paper, we introduce a framework that issues notifications on the actions that are missed. For a proof of concept we collected an egocentric latte making dataset and showed that we can produce action reminders. Generating reminders requires complex decision making. Our solution formulates this problem with a scoring function that trades off between the cost of missing an action and the penalty for issuing a reminder. We evaluated our model in correctness and timeliness of reminders and showed promising results.

References

[1] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011.

[2] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *CVPR*, 2013.

[3] M. Hoai and F. De la Torre. Max-margin early event detectors. In *CVPR*, 2012.

[4] M. Hoai and F. De la Torre. Maximum margin temporal clustering. In *AISTATS*, 2012.

[5] T. Kanade and M. Hebert. First-person vision. *Proceedings of the IEEE*, 2012.

[6] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, 2014.

[7] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011.

[8] M. S. Ryoo, T. J. Fuchs, L. Xia, J. Aggarwal, and L. Matthies. Robot-centric activity prediction from first-person videos: What will they do to me? In *HRI*, 2015.

[9] B. Soran, A. Farhadi, and L. Shapiro. Generating notifications for missing actions: Don't forget to turn the lights off! In *ICCV*, 2015.

[10] F. Zhou, F. De la Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *PAMI*, 2013.