This is a repository copy of Learning Hierarchical Models of Complex Daily Activities from Annotated Videos.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/128307/

Version: Accepted Version

Proceedings Paper:

https://doi.org/10.1109/WACV.2018.00182

© 2018, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.
Abstract

Effective recognition of complex long-term activities is becoming an increasingly important task in artificial intelligence. In this paper, we propose a novel approach for building models of complex long-term activities. First, we automatically learn the hierarchical structure of activities by learning about the ‘parent-child’ relation of activity components from a video using the variability in annotations acquired using multiple annotators. This variability allows for extracting the inherent hierarchical structure of the activity in a video. We consolidate hierarchical structures of the same activity from different videos into a unified stochastic grammar describing the overall activity. We then describe an inference mechanism to interpret new instances of activities. We use three datasets, which have been annotated by multiple annotators, of daily activity videos to demonstrate the effectiveness of our system.

1. Introduction

A challenging area of research in computer vision and artificial intelligence is the representation and recognition of human activities from observed visual data. One challenge is that different subjects perform similar tasks with high variation, and with activities extending for longer periods, their spatio-temporal structure tends to become more complex. Also, low-level visual trackers (such as skeleton/object trackers) are often noisy, causing errors due to occlusion, changing lighting, etc which creates a high degree of uncertainty. Most traditional methods for activity recognition were conceived to deal with short clips of simple human activities, e.g. [1, 2, 16], and despite the long history of this research, existing methods tend to become ineffective when dealing with activities over longer periods of time. We propose a novel method for building a hierarchical activity model from mark-up of activities acquired from multiple annotators in a video corpus. Multiple human annotators identify activities at different levels of conceptual granularity. Our method automatically infers a ‘part-of’ hierarchical activity model from this data using semantic similarity of textual annotations and temporal consistency. We use the resulting model to interpret previously unseen videos in terms of the conceptual categories of the acquired model, thereby providing a layered/compositional description that is naturally understandable by people. Our method is robust to noise and can deal with insertion, deletion and substitution errors resulting from misdetections of low-level action recognition.

A hierarchical model of complex activities is learned from training videos. We assume that these videos have been annotated by multiple subjects who naturally tend to describe activities at different levels of granularity. Then the hierarchical activity structure is extracted as follows. Using the annotations consisting of temporal intervals with activity labels, we reduce redundancy and noise by interval clustering using a distance measure that takes into account the temporal overlap of the intervals and the semantic similarity of the labels. Note that there is no information provided by the annotators as to which activity interval is a child of which other activity interval (‘part of’ relation). These parent-child relations among the clustered intervals is then automatically learned by optimising a cost function which examines the configuration of the clusters and their labels’ common theme. This results in a hierarchical model that represents the complex activity occurring within the annotated video. Hierarchies generated from multiple training videos are then combined together to produce a unified hierarchical and probabilistic model of all observed activities. The model is then represented as a grammar which naturally captures the variation in activities along with their proba-
Figure 1. Flowchart of the overall framework

bilistic likelihood through ‘OR’ rules. The grammar further allows for tractable inference and interpretation of new observations using a multi-threaded parsing algorithm inspired by the well-known Earley Parser [5], similar in some aspects to [24]. Figure 1 shows the flowchart of our framework.

2. Related Work

Activity recognition approaches can be categorised into simple action and complex activity recognition [22]. Complex activity recognition builds on modelling sub-parts of the activity within the overall model, and a hierarchical structure is naturally emergent. In the context of activity recognition, hierarchical models have been developed as extensions of traditional graphical models, such as hidden Markov models (HMM) and other dynamic Bayesian networks (DBN). For example, [20] present a hierarchical DBN that jointly models the activities along with their surrounding environment. DBN’s ability to deal with noise and uncertainty whilst capturing the temporal structure of complex activities make them an attractive approach. [11] were amongst the first to propose a hierarchical extension to the traditional HMM to recognise behavioural patterns. In [14], activity recognition is performed using an ‘Abstract HMM’ with only 2-levels of hierarchy in the model.

Notwithstanding their strength, graphical models become less effective as the length and complexity (hierarchy) of the activity increases, since introducing new latent variables and capturing the newly-introduced dependencies quickly becomes intractable. HMM models typically suffer from the Markovian assumption which prevents the representation of temporally-complex activities that exhibit a richer sub-activity model. Some work has been done to alleviate these problems [17, 6], however the models still suffer heavily from the aforementioned problems.

In order to capture longer temporal dependence whilst keeping the model tractable, inspiration has been drawn from the linguistic literature of using grammars that model highly-structured processes, like language. For example, sentences comprise of smaller parts (e.g. verb phrases, noun phrases, etc.) and each of those further comprise of verbs, determiners, etc., creating a rich hierarchical structure. Considering complex activities as sentences, [8] used a stochastic context-free grammar (SCFG) to represent the activities, coupled with HMMs at the lowest level to recognise primitive actions and provide them as terminals to the grammar. Then, [18] extended this work into multi-agent activities. In both systems, the rules of the grammar were manually designed and complex temporal relations between activities were not fully utilised. In contrast, the rules of grammar were learnt in [24, 15] and Allen’s temporal logic [4] was used to capture temporal relations. However, the approach was only evaluated and shown to be successful on simplistic activities e.g. jumping jacks, lifting arms, etc. Moreover, atomic actions were modeled from trajectory motion data of various elements in the video and each primitive describes a simple basic motion in terms of point coordinates and motion parameters. This representation is conceptually opaque and do not allow for interpretation of the learnt language at test time. [3] have applied hierarchical compositional structures to models complex activities however the hierarchies are bounded to fixed number of levels and are partially describable in the training language.

In this paper, our contributions are to learn activity models in the way humans perceive them by eliciting annotations of activity videos at different levels of granularity from multiple annotators. Our system uses semantic matching of activity components to a) unify variable description of the same activity and b) learn the parent-child pairing between intervals to generate an activity hierarchy. We also devise a method to learn an extended stochastic grammar representation capable of representing temporally complex parallel activities and parsing of automatically detected low-level actions into their most likely hierarchical interpretations. Multiple annotations are efficiently acquired using platforms such as Amazon Mechanical Turk, CrowdFlower etc. [13]. Crowdsourcing offers a large pool of partici-
pants which ensures variability in the annotations collected from participants, and sufficient coverage of possible interpretations. Furthermore, responses are unbiased since respondents have little or no knowledge of the research. This ensures that a wide human perspective of activities can be captured and utilized for model learning. The learned model is therefore capable of being semantically interpreted in the training language.

3. Modelling an Activity Hierarchy

In this section we describe the method used to build and learn the hierarchical model representing the activities from the datasets used. The model captures for a long-term activity: 1) the hierarchical structure of the activity (Figure 7), 2) the temporal arrangement of composite actions (Figure 4), and 3) the variability in activities (Figure 5). The hierarchical model is a graph structure with the nodes being activities and the directed edges connecting them representing the parent-child relationships. Activities can be expanded to their constituent sub-activity nodes, connected in-between with temporal relation nodes that define the temporal order in which those activities occur. This structure is referred to as an activity cluster. Each parent node might have multiple activity cluster children that present the variations of that activity and their probabilities.

3.1. Hierarchy from a single video

Clustering Interval Data per Video Multiple annotators were asked to specify the start and end time of activity intervals that they can identify in each video and label them with a simple English sentence. That is, an activity interval $\iota(s, e, l)$ is a representation of a candidate activity starting at time $s$, ending at time $e$, and carrying the label $l$. The length of $\iota$ is its time span, denoted by $\text{len}(\iota) = e - s$. The use of multiple annotators produces annotations at different levels of granularity which aids in describing the hierarchy of the entire activity. However, the challenge of unifying different and ‘noisy’ descriptions of similar meanings arises; for example, ‘picking up cup’ compared to ‘lifting cup’, ‘taking mug’, ‘retrieving the glass’, etc. To handle this redundancy, we define a distance function of activity intervals in a particular video that employs a measure of semantic similarity between labels proposed by [7]. If $\iota_1(s_1, e_1, l_1)$ and $\iota_2(s_2, e_2, l_2)$ are two activity intervals and $\gamma(l_1, l_2) \in [0, 1]$ is linguistic semantic similarity [7] between labels $l_1$ and $l_2$, we define the distance $\delta(\iota_1, \iota_2)$ to capture not only the temporal overlap of the two intervals, but also the semantic similarity of their labels, as shown in equation 1.

$$
\delta(\iota_1, \iota_2) = \begin{cases} 
\eta\left(|s_2 - s_1| + |e_2 - e_1|\right) & \text{if } \gamma(l_1, l_2) > \hat{\gamma} \\
1 & \text{otherwise}
\end{cases}
$$

$\eta$ is a normalisation term that forces $\delta$ to stay in the range $[0, 1]$, where $\delta$ is set to 1 for the most different pair of intervals that are considered. $\hat{\gamma} \in [0, 1]$ is the cut-off threshold of semantic similarity at which activity intervals do not relate semantically enough, setting $\delta$ to the maximum value 1.

If $n$ activity intervals are identified in a particular video by annotators, we compute the distance matrix $D_{n \times n}$ where $D_{ij} = \delta(\iota_i, \iota_j)$. We use agglomerative clustering with complete linkage on $D$ to cluster together similar activity intervals that virtually represent the same activity. We stop the clustering once $\delta$ of any cluster exceeds a threshold $\delta$. Then the prototype of each resulting cluster $C = \{\iota_1, \ldots, \iota_m\}$ produces an activity node: a multi-label activity interval $I(\pi, l, L)$ with start time $\pi = \sum_{i \in C} s_i/m$, end time $l = \sum_{i \in C} e_i/m$, and a collective set of labels $L = \{\iota\}_{i \in C}$, see Figure 2.

![Figure 2. Sample activity interval labels for one training video](image)

Learning a Hierarchy from a Video We learn a hierarchy which is defined as a set of parent-child relation between activity nodes. Activity node $I_i$ is a candidate child of $I_j$, denoted by $I_i \prec I_j$, if the two nodes 1) have semantically-similar labels up to some threshold, and 2) $I_j$ temporally subsumes or significantly overlaps $I_i$. We allow partial overlap along with full subsuming of intervals when deciding the relation since always observing fully subsumed intervals requires noise free annotations of temporal boundaries. This is rare when different annotators are annotating the same video. Note that, while a parent could have many children, only one node will be the actual parent of some $I_i$. Hence, an optimisation is required over candidate relations.

First, the semantic similarity filter is applied for every node and its candidate parents. This enables excluding ‘noise’ nodes that bear no semantically valid connection to the main activity (e.g., ‘scratching head’, ‘fidgeting with
At the same time, it allows separating parallel activities to individual hierarchies.

To find the optimal parent node for a child, temporal overlap is considered for the semantically-relevant candidates. Temporal overlap \( o_{ij} \) between two overlapping nodes \( I_i \) and \( I_j \) is defined as \( o_{ij} = \min(e_i, e_j) - \max(s_i, s_j) \), when \( \min(e_i, e_j) > \max(s_i, s_j) \) and 0 otherwise. We then define a cost function \( C \) that gauges the suitability of a ‘child-parent’ relation between \( I_i \) and \( I_j \) shown in equation 2 where \( I_i \) and \( I_j \) are two intervals, \( |I_s| \) denotes the length of any interval and \( o_{ij} \) is the overlap between the two intervals.

\[
C(I_i \prec I_j) = \begin{cases} 
\frac{|I_i|^2}{o_{ij}(|I_j| - |I_i|)} & \text{if } |I_j| > |I_i| \text{ and } o_{ij} > 0 \\
\infty & \text{otherwise}
\end{cases}
\] (2)

This function favours the shortest parent possible with the best possible overlap. The intuition behind this function is further illustrated in Figure 3. Notice from cases a and b that the cost \( c_{ij} \) will be larger for case b than case a since a larger section of \( I_j \) is overlapping \( I_i \). At the same time, observing cases c and d, it is clear that case d will obtain a lower cost as it has the shortest potential parent, which is the correct behavior.

By computing \( c_{ij} = C(I_i \prec I_j) \) for all possible pairs of nodes with positive overlap, we set \( I_i \prec I_j \iff j = \arg\min_{1 \leq j \leq N} c_{ij} \), for all \( i \), breaking ties randomly. Notice that the highest-level activity nodes would compute a cost of infinity with all other candidates. In this case, the node is paired with ‘root’ parent node which is always included in any hierarchy as the highest level activity.

**Temporal Sequence Encoding** The temporal sequence of sub-events is a highly descriptive feature. In our model, we capture the temporal sequence using the well established Allen’s Temporal Logic [4]. Inspired by [19], where this logic is used to encode temporal sequences of qualitative relations between objects over time, we abstract that concept to encode the temporal sequence occurring between activities at every level of the hierarchy. For example, in the interval graph of ‘making tea’ seen in Figure 4, note that the ‘Put Sugar’ and ‘Put Milk’ intervals are related through a temporal (overlaps) relation, and that ‘Put Milk’ interval occurs before ‘Mix Ingredients’ and thus uses the (before) relation, and so on. This logic can be used to define temporal relations between interval pairs.

The immediate children of an activity can be thought of as a (partially) ordered set of siblings, and the order is captured by the above described encoding. The siblings along with their temporal relations make up an activity cluster which is denoted as \( K = (S, R) \), where \( S \) is a vector of the cluster’s activities, and \( R : S \times S \rightarrow \{\text{meets, overlaps, before,}...\} \) captures the pairwise temporal relation between the members of \( S \). The example in Figure 4 illustrates this further.

### 3.2. General hierarchy from training videos

Previously learned hierarchies per video are taken as input and a single most generalised hierarchical model of the overall activities is abstracted. The model is augmented with probabilities to allow for uncertainty and robustness against noisy observations.

The model is generated by merging training hierarchies incrementally which allows for real-time learning. To illustrate this, we introduce the following notation. Let \( \mathcal{H} = \{H^1, H^2, ..., H^N\} \) be the set of all \( N \) training hierarchies where each \( H \in \mathcal{H} \) is represented as a 3-tuple, \( H = (\Pi, \Phi, P) \). \( \Pi \) denotes the list of all the parent (non-leaf) nodes. \( \Phi \) maps each \( \pi \in \Pi \) to the set of children activity clusters of \( \pi \). Finally, \( P \) maps a parent node \( \pi \in \Pi \) to a probability distribution over its children activity clusters \( \Phi(\pi) \). This distribution defines the likelihood of a child activity cluster, capturing the variation in the way a parent activity can be performed.

We can formally describe the abstraction process of the hierarchies in algorithm 1. Note that two hierarchies are taken as input, say \( a \) and \( b \), and an abstracted hierarchy is
Algorithm 1 Abstraction of Hierarchies

1: Initialise $\Pi, \Phi, P$
2: $\Pi := \Pi^0 \cup \Pi^1$
3: for each parent $\pi \in \Pi$ do
4: $\Phi^* := \Phi^*(\pi) \cup \Phi(\pi)$
5: $n := |\Phi^*|$
6: Initialise $P^* : \Phi^* \rightarrow [0,1]$
7: for each child cluster $c \in \Phi^*$ do
8: Append $c$ to $\Phi(\pi)$
9: Initialise $p := 1$
10: for each cluster $c' \in \Phi^* - c$ do
11: if $\text{cluster} \text{Matching}(c, c') < T$ then
12: $p = p + 1$
13: remove $c'$ from $\Phi^*$
14: $P^*(c) = p/n$
15: Add $P^*$ to $P$

An example of abstracting hierarchies is illustrated in Figure 5. Starting at the root node and for any non-leaf node, children activity clusters are compared using a cluster matching measure (Line 11) described in the next section. Unique clusters are extracted as the children of new parent nodes in the merged model (Line 8). The probabilities of children activities are computed from the frequency of observing each compared to the total number of possible ways (Line 14). The resulting model includes all unique children from both training hierarchies along with the probabilities of their occurrence based on their observed frequency during training.

Note that multiple child clusters with the same parent denote the alternative ways in which that parent activity could be observed with the associated probability, denoting an OR relationship. The siblings within a cluster, denote an AND relation. The resulting graph is analogous to an AND-OR graph.

Activity Cluster Matching Comparing two activity clusters involves comparison of the components within each cluster. The components of a cluster include the activity nodes of the cluster and the temporal relations in between them. We propose a mechanism to match two activity clusters and return a score of their similarity.

To measure the similarity $d(I_1, I_2)$ of two activity nodes $I_1$ and $I_2$, we look into the semantic similarity of the set of labels of each of the nodes. Assume $L_1$ and $L_2$ are the sets of labels of $I_1$ and $I_2$ of size $N_1$ and $N_2$ respectively and $N_1 > N_2$. We compute the semantic similarity for every pair $(l_1, l_2) \in L_1 \times L_2$ and compute $1 - \delta_{SS}(l_1, l_2)$, producing a cost matrix $\in \mathbb{R}^{N_1 \times N_2}$. We consider this to be an assignment problem and use the Hungarian algorithm [12] on that matrix to match labels based on semantic similarity, giving the matched label set $M_{1,2} \subset L_1 \times L_2$. The unmatched label set is denoted by $U$ and it contains all the labels from the longest list $L_1$ that are not matched in $M_{1,2}$. The pairs in $M_{1,2}$ contribute the sum of their similarities to $d(I_1, I_2)$. On the other hand, each of the unmatched labels in $L_1$ is also matched individually to one of the labels in $L_2$, so that labels which are somehow similar will improve the total similarity, while dissimilar labels contribute poorly. The similarity is defined as $d(I_1, I_2) = (\sum_{M_{1,2}} \delta_{SS}(l_1, l_2) + \sum_{l_1 \in U} \max_{l_2} (\delta_{SS}(l_1, l_2))/N_1$.

For temporal relations, we use the relations of Allen’s temporal logic, $\text{rel} = \{\text{before, meets, overlap, starts, during, finishes, equal}\}$ and their inverses, and define the distance between any two, $t(...)$, to be the separation in the list, normalised by the number of relations. If the temporal relations are edges that link two activity nodes in an activity cluster, we can define a measure of similarity between two such edges. Assume $I_1 r I_2$ means that node $I_1$ is related to node $I_2$ with relation $r$. Then, the distance between $I_1 r I_2$ and $I_1 r' I_2$ can be defined as $\alpha(d(I_1, I_1') + d(I_2, I_2') + (1 - \alpha)\{(1 - t(r, r'))\}$ where $\alpha$ is a parameter that controls the contribution of the node distances and temporal similarity terms. For two activity clusters, similar to node matching, we pair-match the children nodes using the Hungarian algorithm to find the best matching. An additional penalty term $p$ is added for all unmatched nodes between two clusters, penalising greater disparity in the number of components.

4. Inference

The recognition task in our framework is defined as following: given an input test video, we first automatically detect the low-level actions in that video. We then infer the most likely activity hierarchy that generates a similar set of low-level actions. To detect low-level actions, we simply train a discriminative model using the state-of-the-art action recognition approach proposed by [21]. Low-level action classes of our datasets are automatically emergent from previously learned optimum hierarchies whereby leaf nodes of the hierarchies represent the lowest level actions. We use this knowledge to extract training instances of each action class from training videos and train low-level activity models. For inferring the high-level activity hierarchy, we first transform the hierarchical activity model into an extended grammar. This allows for utilising well-established algorithms to parse a 'string' of temporally-complex atomic actions, recognised from the test video. An example of the grammar is shown in Figure 5 on the right. Leaf nodes in the final model are the terminals in the grammar. Each activity cluster translates into a production rule with the parent node being the left-hand side (lhs) of the grammar rule. The rules are extended with a relation matrix $R$ which describes the temporal dependence between the nodes of the activity cluster, and a probability $p$ of observing that alternative
which can generate the same lhs.

Many parsing algorithms in language literature exist that parse a given input stream of words into a tree using a grammar model [9, 23, 5]. However, sentences are normally presented as linear sequences of words. In our system, observations (atomic actions) are not linear sequences, but contain a more complex temporal structure modeled using Allen’s temporal logic. We use an extended version of the Earley parser proposed by [24] to parse a set of observations by maintaining multi-threaded parses over the input. This parser relaxes the constraint that grammar rules and input stream of intervals follow a sequential flow. This allows for parsing different symbols in the input stream at different positions as the grammar expects. Deletion and insertion errors are handled within the parsing algorithm. Differently from previous work, we extract the parse tree that best describes the input, by maximising probability along with coverage. Maximising probability returns the parse tree that has the highest likelihood according to the grammar rules. However, the maximally probable tree may include a high percentage of deletion errors, we therefore also introduce the use of coverage. Coverage defines a ratio of the number of input symbols described by the selected parse tree. This ratio does not accounts for deletion errors and is therefore a good indication of the descriptive strength of the parse tree. The most suitable parse tree which describes majority of the input symbols and is highly probable is retrieved.

5. Evaluation

To our knowledge, no previous approaches exist that tackle a similar research problem. For evaluation we use well-defined baselines and an evaluation methodology from recent literature [24] to demonstrate the effectiveness of our system.

Dataset To demonstrate the strength of our system, we found three publicly available datasets which have been annotated using multiple annotators: Leeds Activity Dataset (LAD)\(^1\), Cornell Activity Dataset (CAD-120)\(^2\) [10] and Complex Long Activities Dataset (CLAD)\(^3\). Examples from these datasets are shown in Figure 6, and their properties are summarized in Table 5. The CLAD dataset presents most complexity in terms of lengthy activities, for example ‘lunch in a restaurant’ or ‘working in an office’, deep activity hierarchies and crowdsourced annotations. These annotations consists of highly variable and inconsistent activity labels. Creators of this dataset ensure the quality of annotations by using spell checks, start/end frame consistency etc. We further process labels using typical NLP techniques including lemmatisation followed by another spell checker. Finally, through learning a probabilistic grammar, spurious rules exhibit a low probability and can be deleted from the final grammar. The CAD dataset natively has only two levels of annotations: top-level and atomic-level activities. To demonstrate our system using this dataset, we augmented the dataset with annotations of intermediate levels made by an independent annotator. Unlike the CLAD dataset, which has highly variable crowdsourced annotations, both the CAD and LAD datasets are more homogeneous since all the annotators used a fixed set of labels. However, they still present a challenge due to noisy temporal boundaries of activity intervals. The common factors in all datasets are that they were annotated by multiple annotators and they all comprise of tasks made up of many levels of sub-activities which gives rise to a hierarchy of events.

---

\(^1\)http://dartportal.leeds.ac.uk/dataset/lad
\(^2\)http://pr.cs.cornell.edu/humanactivities/data.php
\(^3\)https://doi.org/10.5518/249
In this experiment, we evaluate our semantic clustering against known methods, such as k-means and affinity propagation, using normalised mutual information (NMI) to validate the results against ground truth clustering. Furthermore, we evaluate the effect of semantic similarity included within clustering. K-means is applied on the label’s temporal boundary (start and end time) alone whereas affinity propagation uses the same \( D \) matrix, which includes semantic similarity, as our clustering method. Clustering is tested on the CLAD dataset since this dataset exhibits crowdsourced annotations. These annotations present multiple labels of the same activities from different annotators. Clustering is based on the label’s temporal boundaries and their semantic similarities. The LAD and CAD datasets used preset lists of labels to choose from thus avoiding language ambiguity and redundancy in annotations, therefore clustering their labels were not needed. The results of clustering are in Table 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>NMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Clustering Method with sem sim</td>
<td>94.5%</td>
</tr>
<tr>
<td>Our Clustering Method without sem sim</td>
<td>88.4%</td>
</tr>
<tr>
<td>K-Means using temporal boundary</td>
<td>65.4%</td>
</tr>
<tr>
<td>Affinity Propagation with sem sim</td>
<td>67.5%</td>
</tr>
</tbody>
</table>

Table 2. Clustering method comparison. Our method outperforms other methods, and semantic similarity gives a higher quality result.

Hierarchical clustering is captured with multiple descriptions produced by different annotators of the same activity label. The effect of noisy and ambiguous annotations is apparent from the significant difference in a crowdsourced labelled CLAD dataset versus clean preset in-house labelled LAD and CAD datasets. Despite the added challenge of incorporating linguistic semantics to find the optimum pairing in the CLAD dataset, a decent performance is still achieved as seen when compared to the baseline. Figure 7 shows some interesting snippets of sample hierarchies learned given intervals input data from each of the datasets.

Table 3. Accuracy for optimum pairing and learning of activity hierarchy on the three different datasets.

It can be seen from Table 3 that handling the temporal relations more carefully is beneficial, and that semantic similarity contributes significantly to the accuracy, especially to separate parallel activity hierarchies and noise. An example of this can be seen in Figure 7(c). The semantic similarity threshold at which a candidate interval is disqualified as a parent is learned by optimising the function shown in Figure 8. Lastly, describing each node with a set of labels rather than a single label also yields a higher performance. This is expected behaviour since more information is captured with multiple description produced by different annotators of the same activity label. The effect of noisy and ambiguous annotations is apparent from the significant difference in a crowdsourced labelled CLAD dataset versus clean preset in-house labelled LAD and CAD datasets. Despite the added challenge of incorporating linguistic semantics to find the optimum pairing in the CLAD dataset, a decent performance is still achieved as seen when compared to the baseline. Figure 7 shows some interesting snippets of sample hierarchies learned given intervals input data from each of the datasets.

**Parsing** Having learned individual activities from different training videos, we then merge the different hierarchies to build a probabilistic extended grammar defining the overall activity. We build the grammars describing the top-level tasks such as ‘breakfast at home’ and ‘working in an office’. To evaluate the correctness of the grammar model, we run leave-one-out cross validation on the models. We present in Table 4 and 5 the ratio of correctly-interpreted terminals. We use the multi-threaded parsing mechanism as described in section 4 that allows for inclusion of complex temporal streams as input. It can be seen that parsing with inclusion of temporal relations describing the sequence out-perform a parsing outcome that treats input as sequential. The evaluation metric used is similar to the one employed by [24] to demonstrate their multi-threaded parsing mechanism. It is the correctly parsed terminal percentage \( CPT \):

\[
CPT = \frac{NC + NI}{NT}
\]
Figure 7. Examples of a well learned hierarchy from optimum pairing of intervals for each dataset. Full trees are shown for the CAD and LAD datasets, while a snippet of a larger tree is shown for CLAD dataset. Notice that parallel activities have been assigned to the correct parents in the trees.

Figure 8. Effect of altering the semantic similarity threshold on accuracy of the CLAD dataset.

where $NC$ is the number of correctly parsed terminals, $NI$ is the number of identified insertion errors in the input stream and $NT$ is the total number of terminal. As having a single parse tree to describe all terminals is rare because it is unlikely that a natural activity occurs exactly as trained before, we devise an integer programming solution to extract the optimum set of parse trees that cover the input terminal with 1) maximum coverage, 2) high probability and 3) minimum amount of deletion errors. The resulting parse trees are then inspected and the $CPT$ is reported. It should be noted that the input stream of terminals is noisy, this causes a high number of insertion, deletion and substitution errors, however, our parser is able to handle these errors and produce the most likely parse of the activity given low-level actions input stream. We present results with automatically detected and ground truth labeled low-level actions input stream in table Table 4 and 5.

Table 5. Using automatically detected low-level actions, correctly-interpreted terminals (CPT) in a 5-fold validation process of hierarchical model building/parsing.

<table>
<thead>
<tr>
<th>Cluster Matching</th>
<th>LAD</th>
<th>CAD</th>
<th>CLAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential Input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exact</td>
<td>68%</td>
<td>70%</td>
<td>33%</td>
</tr>
<tr>
<td>Ours</td>
<td>71%</td>
<td>77%</td>
<td>34%</td>
</tr>
<tr>
<td>Multi-threaded</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exact</td>
<td>65%</td>
<td>73%</td>
<td>59%</td>
</tr>
<tr>
<td>Ours</td>
<td>72%</td>
<td>73%</td>
<td>58%</td>
</tr>
</tbody>
</table>

Table 4 and 5 shows the results from parsing the inputs from each datasets. Similar to the hierarchy generation results, note that CAD and LAD datasets outperform the CLAD dataset due to consistent and clean labels. Further, it is apparent that using multi-threaded parsing, non-sequential and temporally complex input streams are handled and outperform classic sequential parsing. This is particularly significant in the CLAD dataset, which exhibits more instances of parallel activities than the other datasets. Furthermore, notice that using our loosely constrained activity cluster matching builds more accurate models and achieves higher parsing accuracy than those with exact cluster matching. This demonstrates the ability to abstract variations in different instances of activities. Finally, note that the results using an automatically detected input stream, see table 5, are of slightly lower accuracy compared to using a ground truth input stream, see table 4.

6. Conclusion

In this paper, we introduce a novel approach for learning activity hierarchies using multiple annotations of videos. We first show a learning method to acquire hierarchical structures of activities from multiple annotations of a video. We present an algorithm for consolidation and abstraction of a general model of activity from multiple individually learned hierarchies. Finally, we demonstrate the effectiveness of our system using a hierarchical, multi-threaded parser. One limitation of this work is in handling repetitive activities and this is a direction of future work.
References


