Learning Hierarchical Models of Complex Daily Activities from Annotated Videos

Jawad Tayyub
University of Leeds
sc12jbmt@leeds.ac.uk

Majd Hawasly
Five AI
m.hawasly@five.ai

David C. Hogg
University of Leeds
d.c.hogg@leeds.ac.uk

Anthony G. Cohn
University of Leeds
a.g.cohn@leeds.ac.uk

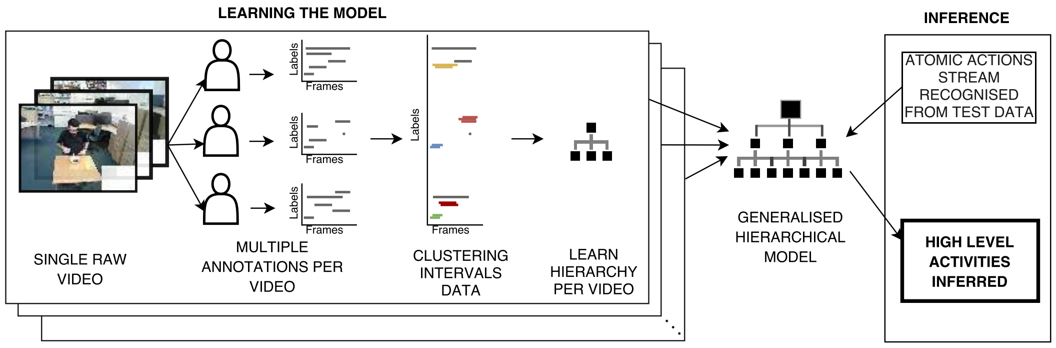
Abstract

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 activiites along with their proba-



LEARNING THE MODEL
INFERENCE
Labels
Frames
SINGLE RAW VIDEO
MULTIPLE ANNOTATIONS PER VIDEO
CLUSTERING INTERVALS DATA
LEARN HIERARCHY PER VIDEO
GENERALISED HIERARCHICAL MODEL
ATOMIC ACTIONS STREAM RECOGNISED FROM TEST DATA
HIGH LEVEL ACTIVITIES INFERRED

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 participants 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 utilised 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 ι is its time span, denoted by $len(\iota) = e - s$. 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.

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

η is a normalisation term that forces δ to stay in the range $[0, 1]$, where δ 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 δ 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 δ of any cluster exceeds a threshold $\hat{\delta}$. Then the prototype of each resulting cluster $\mathcal{C} = \{\iota_1, \ldots, \iota_m\}$ produces an activity node: a multi-label activity interval $I(\bar{s}, \bar{e}, L)$ with start time $\bar{s} = \sum_{\iota\in\mathcal{C}} s/m$, end time $\bar{e} = \sum_{\iota\in\mathcal{C}} e/m$, and a collective set of labels $L = \{l\}_{\iota\in\mathcal{C}}$, see Figure 2.

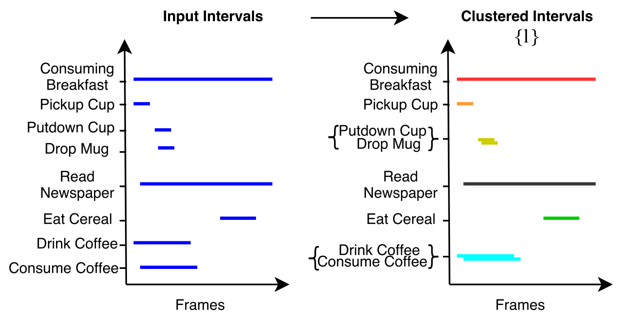


Figure 2. Sample activity interval labels for one training video from combined multiple annotations and their corresponding result from clustering using semantic similarity to produce activity nodes by clustering redundant labels. Note that parallel activities, though sharing the same interval start and end times, are not clustered together due to semantic dissimilarity.

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 $I_i \lhd I_j$, 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 pen', etc.) At the same time, it allows separating parallel activities to individual hierarchies.

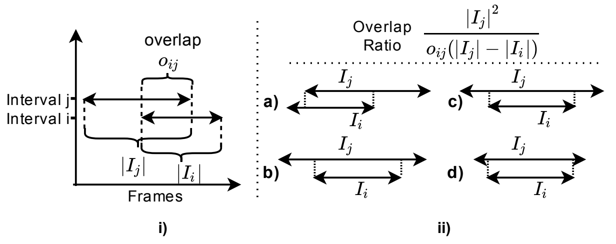


Figure 3. i) Two conflicting intervals i and j, ii) a, b show that the ratio prefers longer overlap preferring b over a, whilst c and d show preference of a shorter parent preferring d over c

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, $|I_*|$ denotes the length of any interval and o_{ij} is the overlap between the two intervals.

$$C(I_i \lhd I_j) = \begin{cases} \dfrac{|I_j|^2}{o_{ij}(|I_j| - |I_i|)} & \text{if } |I_j| > |I_i| \text{ and } o_{ij} > 0 \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

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 $c_{ij} = C(I_i \lhd I_j)$ for all possible pairs of nodes with positive overlap, we set $I_i \lhd I_j \iff j = \arg\min_{1 \leq j \leq N} c_{ij}$, 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 $\langle$*overlaps*$\rangle$ relation, and that *'Put Milk'* interval occurs before *'Mix Ingredients'* and thus uses the $\langle$*before*$\rangle$ relation, and so on. This logic can be used to define temporal relations between interval pairs.

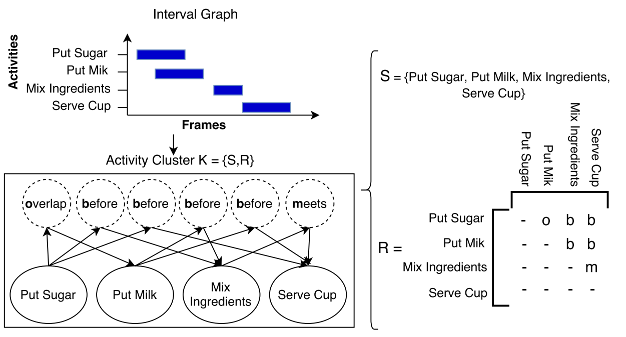
The immediate children of an activity can be thought of as a (partially) ordered set of *siblings*, and the order is captured by the above described encoding. The siblings along with their temporal relations make up an *activity cluster* which is denoted as $K = (S, R)$, where S is a vector of the cluster's activities, and $R : S \times S \rightarrow \{$*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, $H = \langle \Pi, \Phi, P \rangle$. Π denotes the list of all the parent (non-leaf) nodes. Φ maps each $\pi \in \Pi$ to the set of children activity clusters of π. 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



$$R = \begin{array}{c|cccc} & \text{Put Sugar} & \text{Put Milk} & \text{Mix Ingredients} & \text{Serve Cup} \\ \hline \text{Put Sugar} & - & o & b & b \\ \text{Put Mik} & - & - & b & b \\ \text{Mix Ingredients} & - & - & - & m \\ \text{Serve Cup} & - & - & - & - \end{array}$$

Figure 4. An example of encoding temporal information of an activity cluster C represented by an activities vector S and the corresponding temporal matrix R.

returned. The resultant hierarchy is then input back with a new training hierarchy to produce a further abstracted hierarchy. This process repeats until all training hierarchies are incrementally used to build the final model.

Algorithm 1 Abstraction of Hierarchies

```
 1: Initialise $\Pi, \Phi, P$
 2: $\Pi := \Pi^a \cup \Pi^b$
 3: for each parent $\pi \in \Pi$ do
 4:     $\Phi^* := \Phi^a(\pi) \cup \Phi^b(\pi)$
 5:     $n := |\Phi^*|$
 6:     Initialise $P^\pi : \Phi^* \rightarrow [0, 1]$
 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 $clusterMatching(c, c') < T$ then
12:                 $p = p + 1$
13:                 remove $c'$ from $\Phi^*$
14:         $P^\pi(c) = p/n$
15:     Add $P^\pi$ 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 $\in \mathbb{R}^{N_1 N_2}$. 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 longer list $L1$ 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, *rel* = {*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 α 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

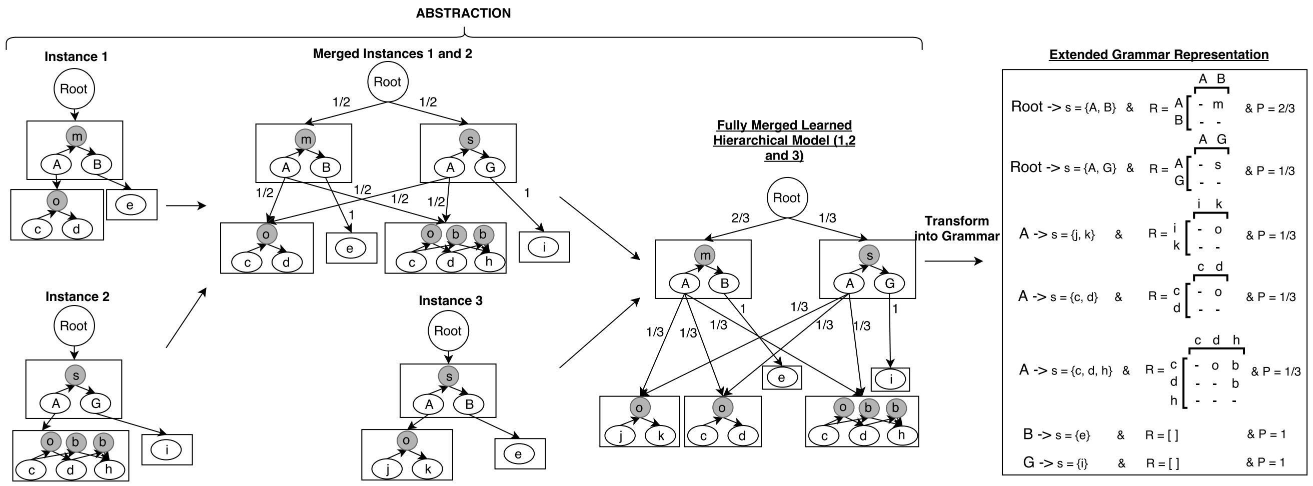

ABSTRACTION
Instance 1
Root
m
A
B
o
c
d
e
Merged Instances 1 and 2
1/2
s
G
h
b
i
1
Instance 2
Instance 3
j
k
Fully Merged Learned Hierarchical Model (1,2 and 3)
2/3
1/3
Transform into Grammar
Extended Grammar Representation
Root -> s = {A, B} & R = [- m; - -] & P = 2/3
Root -> s = {A, G} & R = [- s; - -] & P = 1/3
A -> s = {j, k} & R = [- o; - -] & P = 1/3
A -> s = {c, d} & R = [- o; - -] & P = 1/3
A -> s = {c, d, h} & R = [- o b; - - b; - - -] & P = 1/3
B -> s = {e} & R = [ ] & P = 1
G -> s = {i} & R = [ ] & P = 1

Figure 5. Incremental abstraction of hierarchies. Two training hierarchies are abstracted starting at 'Root'. Then, a third hierarchy produces a further abstracted model. Finally, the model produces an extended grammar representation.

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 handed 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 therefor 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

	LAD	CAD	CLAD
Number of Videos	13	120	62
Length of Videos (Minutes)	0.5-1	0.3-0.5	3-5
Multiple Subjects	×	×	✓
Crowdsourced Annotations	×	×	✓
Depth Of Hierarchy (Levels)	3	2-3	up to 5
Labels Noisy Temporal Boundries	✓	✓	✓
Linguistic Variation in Labels	✓	×	✓

Table 1. Properties of the datasets used.

Figure 6. Examples from the LAD, CLAD and CAD datasets.

Clustering 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 similairty, 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.

Method	**NMI**
Our Clustering Method with sem sim	94.5%
Our Clustering Method without sem sim	88.4%
K-Means using temporal boundary	65.4%
Affinity Propagation with sem sim	67.5%

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

Hierarchy building Individually-learned hierarchies per video are evaluated for correctness using ground truth trees as presented in Table 3. The hierarchies are created by pairing intervals in *'parent-child'* relations. A baseline in which any interval fully-subsumed by another will be linked as a child-parent pair is presented as a point of comparison. This is a stricter pairing paradigm that does not handle i) the variability in start and end times, ii) proper pairing of parallel activities and iii) semantic analysis of parent's and the corresponding children's labels. The results are presented for all three datasets in Table 3.

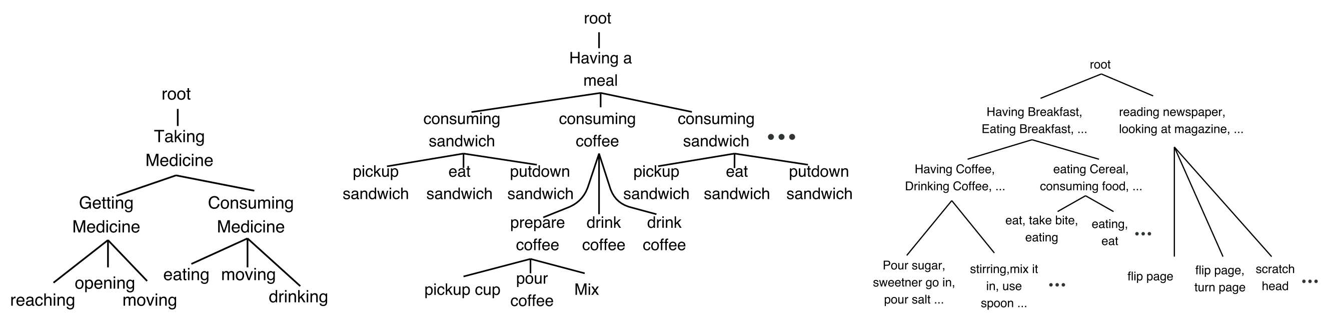
	LAD	CAD	CLAD		
			Semantic Similarity	Multi Label	Single Label
Baseline	68%	24%	✓	61%	55%
			×	53%	53%
Learned Hierarchy	91%	89%	✓	79%	70%
			×	63%	63%

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:

$$CPT = \frac{NC + NI}{NT}$$



(a) CAD example hierarchy (b) LAD example hierarchy (c) CLAD example hierarchy

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.

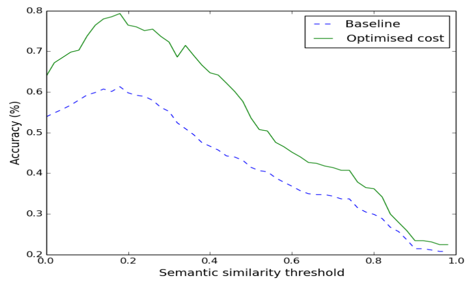


Figure 8. Effect of altering the semantic similarity threshold on accuracy of the CLAD dataset.

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.

	Cluster Matching	LAD	CAD	CLAD
Sequential Input	Exact	76%	90%	66%
	Ours	78%	93%	66%
Multi-threaded	Exact	80%	90%	71%
	Ours	80%	93%	76%

Table 4. Using **ground-truth labeled** low-level actions, correctly-interpreted terminals (CPT) in a 5-fold validation process of hierarchical model building/parsing.

	Cluster Matching	LAD	CAD	CLAD
Sequential Input	Exact	68%	70%	33%
	Ours	71%	77%	34%
Multi-threaded	Exact	65%	73%	59%
	Ours	72%	73%	58%

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

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

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 90–102. IEEE, 1997.

[2] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[3] E. E. Aksoy, A. Orhan, and F. Wörgötter. Semantic decomposition and recognition of long and complex manipulation action sequences. *International Journal of Computer Vision*, 122(1):84–115, Mar 2017.

[4] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[5] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[6] R. Hamid, S. Maddi, A. Bobick, and I. Essa. Structure from statistics-unsupervised activity analysis using suffix trees. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[7] L. Han, A. Kashyap, T. Finin, J. Mayfield, and J. Weese. Umbc ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 44–52, 2013.

[8] Y. A. Ivanov and A. F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[9] T. Kasami. An efficient recognition and syntaxanalysis algorithm for context-free languages. Technical report, DTIC Document, 1965.

[10] H. Koppula and A. Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *International Conference on Machine Learning*, pages 792–800, 2013.

[11] S. Luhr, H. H. Bui, S. Venkatesh, and G. A. West. Recognition of human activity through hierarchical stochastic learning. In *PerCom 2003: Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications*, pages 416–422. IEEE, 2003.

[12] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[13] L.-V. Nguyen-Dinh, C. Waldburger, D. Roggen, and G. Tröster. Tagging human activities in video by crowdsourcing. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 263–270. ACM, 2013.

[14] S. Osentoski, V. Manfred, and S. Mahadevan. Learning hierarchical models of activity. Technical report, DTIC Document, 2005.

[15] M. Pei, Z. Si, B. Z. Yao, and S.-C. Zhu. Learning and parsing video events with goal and intent prediction. *Computer Vision and Image Understanding*, 117(10):1369–1383, 2013.

[16] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

[17] C. S. Pinhanez and A. F. Bobick. Human action detection using PNF propagation of temporal constraints. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 898–904. IEEE, 1998.

[18] M. S. Ryoo and J. K. Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1709–1718. IEEE, 2006.

[19] M. Sridhar, A. G. Cohn, and D. C. Hogg. Discovering an event taxonomy from video using qualitative spatio-temporal graphs. In *ECAI 2010-19th European Conference on Artificial Intelligence, Proceedings*, volume 215, pages 1103–1104. IOS Press, 2010.

[20] A. Subramanya, A. Raj, J. Bilmes, and D. Fox. Hierarchical models for activity recognition. In *2006 IEEE Workshop on Multimedia Signal Processing*, pages 233–237. IEEE, 2006.

[21] J. Tayyub, A. Tavanai, Y. Gatsoulis, A. G. Cohn, and D. C. Hogg. Qualitative and quantitative spatio-temporal relations in daily living activity recognition. In *Asian Conference on Computer Vision*, pages 115–130. Springer, 2014.

[22] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.

[23] D. H. Younger. Recognition and parsing of context-free languages in time n 3. *Information and control*, 10(2):189–208, 1967.

[24] Z. Zhang, T. Tan, and K. Huang. An extended grammar system for learning and recognizing complex visual events. *IEEE transactions on pattern analysis and machine intelligence*, 33(2):240–255, 2011.