A Generalized Earley Parser for Human Activity Parsing and Prediction

Siyuan Qi, Baoxiong Jia, Siyuan Huang, Ping Wei, and Song-Chun Zhu

Abstract—Detection, parsing, and future predictions on sequence data (*e.g.*, videos) require the algorithms to capture non-Markovian and compositional properties of high-level semantics. Context-free grammars are natural choices to capture such properties, but traditional grammar parsers (*e.g.*, Earley parser) only take symbolic sentences as inputs. In this paper, we generalize the Earley parser to parse sequence data which is neither segmented nor labeled. Given the output of an arbitrary probabilistic classifier, this generalized Earley parser finds the optimal segmentation and labels in the language defined by the input grammar. Based on the parsing results, it makes *top-down* future predictions. The proposed method is generic, principled, and widely applicable. Experiment results clearly show the benefit of our method for both human activity parsing and prediction on three video datasets.

Index Terms—video understanding, high-level vision, activity recognition, activity prediction, grammar models, grammar parser

1 Introduction

We propose an algorithm to tackle the task of understanding complex human activities from (partially-observed) videos from two important aspects: activity recognition and prediction. This is a ubiquitous problem driven by a wide range of applications in many perceptual tasks. Some scenarios further require the algorithm to have both recognition and prediction capabilities, *e.g.*, assistive robots would need to recognize the current human activity and provide future-aware assistance.

To find a joint solution of activity recognition and prediction, we need to consider two questions: 1) what is a good representation for the structure of human activities/tasks, and 2) what is a good inference algorithm to cope with such a representation. A popular family of representations for events is the Markov models (*e.g.*, hidden Markov Model). However, Markov models are not expressive enough since human tasks often exhibit non-Markovian and compositional properties. Hence we argue that 1) a representation should reflect the hierarchical/compositional task structure of long-term human activities, and 2) an inference algorithm should recover the hierarchical structure given the past observations, and be able to predict the future.

We refer to the Chomsky hierarchy to choose a model to capture the hierarchical structure of the entire history. The Chomsky hierarchy is a containment hierarchy of classes of formal grammars in the formal languages of computer science and linguistics. The reason is that activities are analogous to languages: actions are like words and activities are like languages. The Chomsky hierarchy categorizes language models into four levels: 1) Turing machines, 2) context-sensitive grammars, 3) context-free grammars, and 4) regular grammars. Higher-level models contain lower-level models, and Markov models belongs to the lowest level (regular grammars). In this paper, we propose to use *context-free grammars* to parse and predict human activities. In the definition of formal language theory, a grammar is a set of production rules for sentences in a formal language. In our case, the rules describe how to form sentences (activities) from the language's alphabet (actions) that are valid.

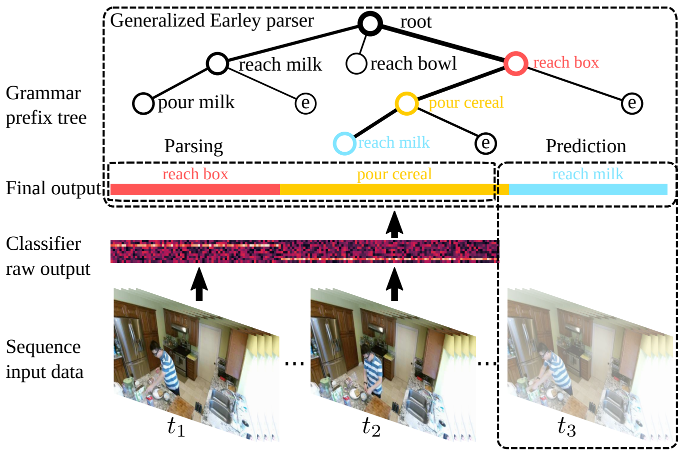


Fig. 1: The generalized Earley parser segments and labels the sequence data into a label sentence in the language of a given grammar. The input of the parser is a matrix of probabilities of each label for each frame, given by an arbitrary classifier. Based on the classifier output, we compute prefix probabilities for different label prefices. The parsing process is achieved by expanding a grammar prefix tree and searching heuristically in this tree according to the prefix probabilities. Future predictions can also be made based on the grammar prefix tree. In the figure, thicker edges indicate higher probabilities and e denotes the end of a sentence.

However, it has not been possible to directly use symbolic grammars to parse and label sequence data (*e.g.*, videos). Traditional grammar parsers take symbolic sentences as inputs instead of noisy sequence data. The data has to be i) segmented and ii) labeled to be parsed by existing

- *Siyuan Qi, Baoxiong Jia, Siyuan Huang, Song-Chun Zhu are with the Department of Computer Sience and Statistics, University of California, Los Angeles, Los Angeles, CA 90095. E-mail: syqi@cs.ucla.edu*
- *Ping Wei is with Xi'an Jiaotong University, Xi'an, 710049.*
- *Corresponding author: Ping Wei. E-mail: pingwei.pw@gmail.com.*

grammar parsers. One naive solution is to first segment and label the data using a detector and thus generating a label sentence. Then grammar parsers can be applied on top of it for parsing prediction. But this is apparently non-optimal, since the grammar rules are not considered in the detection/classification process. It may not even be possible to parse this label sentence, because the output from detectors are very often grammatically incorrect.

In this paper, we design a grammar-based parsing algorithm that directly operates on input sequence data, which goes beyond the scope of symbolic string inputs for classic parsing algorithms. Specifically, we propose a generalized Earley parser to take *probabilistic sequence inputs* instead of deterministic symbolic inputs, based on the classic Earley parser [1]. The algorithm finds the optimal segmentation and label sentence according to both a symbolic grammar and a classifier output of probabilities of labels for each frame as shown in Figure 1. Optimality here means maximizing the joint probability of the label sentence according to the grammar prior and classifier output while being *grammatically correct*.

The difficulty of achieving this optimality lies in the joint optimization of both the grammatical structure and the parsing likelihood of the output label sentence. For example, an expectation-maximization-type of algorithm will not work well since i) there is no guarantee for optimality, and ii) any grammatically incorrect sentence has a grammar prior of probability 0. The algorithm can easily get stuck in local minimums and fail to find the optimal solution that is grammatically correct.

The core idea of our algorithm is to directly and efficiently search for the optimal label sentence in the language defined by the grammar. The constraint of the search space ensures that the sentence is grammatically correct. Specifically, a heuristic search is performed on the prefix tree expanded according to the grammar, where the path from the root to a node represents a partial sentence (prefix). We search through the prefices to find the best sentence according to a heuristic. By carefully defining the heuristic as a prefix probability computed based on the grammar prior and classifier output, we can efficiently search through the tree to find the optimal label sentence. Here we draw some analogies to help understand the relationships between different components: the probabilistic output from the classifier is like the observation of hidden states in hidden Markov Models (HMM), the prefix search is like the dynamic/incremental construction of an HMM, and the proposed grammar parser is like the Viterbi parsing algorithm.

The generalized Earley parser has **four major advantages**. **i)** The inference process highly integrates a high-level grammar with an underlying classifier; the grammar gives guidance for segmenting and labeling the sequence data and future predictions. **ii)** The only requirement for the underlying classifier is that the classifier should give probabilisitc outputs. This makes the algorithm widely applicable, since almost all statistical learning classifiers are probabilistic. **iii)** It generates semantically meaningful results (a grammar parse tree) for data sequence, and the process is highly explainable. **iv)** It is principled and generic, as it applies to most sequence data parsing and prediction problems (the data does not have to be videos).

We evaluate the proposed approach on three datasets of human activities in the computer vision domain. The first dataset CAD-120 [2] consists of daily activities and most activity prediction methods are based on this dataset. Comparisons show that our method significantly outperforms state-of-the-art methods on future activity prediction. The second dataset Watch-n-Patch [3] is designed for "action patching", which includes daily activities that have action forgotten by people. Experiments on the second dataset show the robustness of our method on noisy data. The third dataset Breakfast [4] consists of long videos of daily cooking activities. Results on this dataset show comparisons between our method and other structured modeling and language-inspired modeling methods. All experiments show that the generalized Earley parser performs well on both activity parsing and prediction tasks.

This paper makes **three major contributions**.

• It designs a parsing algorithm for symbolic context-free grammars that directly operates on sequence data. It can obtain the optimal segmentation and labels according to the grammar prior and classifier outputs.

• It proposes a prediction algorithm that naturally integrates with this parsing algorithm.

• It formulates an objective for future prediction for both grammar induction and classifier training. The generalized Earley parser serves as a concrete example for combining symbolic reasoning methods and connectionist approaches.

2 Related Work

This paper is an extension of previous ICCV and ICML papers [5, 6]. The extension includes two major aspects. 1) For the method, we have extended the algorithm to incorporate a non-trivial grammar prior into the generalized Earley parser. This makes the algorithm applicable to not only context-free grammars (CFGs) but also probabilistic context-free grammars (PCFGs). 2) In the experiments, we tested the model on more datasets with more comparisons and in-depth analyses.

Activity parsing refers to recognition and segmentation of long-term and complicated activities from videos, whereas action recognition corresponds to short-term actions. They are two extensively-studied topics in computer vision and we refer the readers to a survey [7] for a more comprehensive treatment. The main stream of work on activity recognition is to extend mid-level representations to high-level representations.

These extensions are designed in several different ways to model the complex activity structures. A number of methods have been proposed to model the high-level temporal structure of low-level features extracted from video [8, 9, 10, 11, 12, 13]. Some other approaches represent complex activities as collections of attributes [14, 15, 16, 17]. Another important type of methods builds compositional/hierarchical models on actions [15, 18, 19, 20, 21, 22, 23]. Koppula et al. [2] proposed a model incorporating object affordances that detects and predicts human activities. Wei et al. [24] proposed a 4D human-object interaction model for event recognition. In some recent works, structural models are implicitly learned by neural networks [25, 26, 27, 28, 29].

Grammar models fall into the category of compositional models for temporal structures. Ivanov and Bobick [30] proposed to first generate a discrete symbol stream from continuous low-level detectors, and then applied stochastic context-free parsing to incorporate prior knowledge of the temporal structure. Pei et al. [31] detected atomic actions and used a stochastic context sensitive grammar for video parsing and intent prediction. Similar to the generalized Earley parser, it parses the video in an online fashion and enables prediction. However, the algorithm uses manually defined thresholds to detect action transitions. Kuehne et al. [4] modeled action units by hiddem Markov models (HMMs), and models the higher-level action sequence by context-free grammars. Pirsiavash and Ramanan [32] proposed segmental grammar for video parsing, which extends regular grammars to allow non-terminals to generate a segment of terminals of certain lengths. Vo and Bobick [33] generated a Bayes network, termed Sequential Interval Network (SIN), where the variable nodes correspond to the start and end times of component actions. This network then makes inference about start and end times for detected action primitives. Qi et al. [5] proposed to integrate spatial-temporal attributes to terminal nodes of a context-free grammar. Based on Earley parser, an activity parsing and prediction algorithm is proposed. Overall, grammar-based methods have shown effectiveness on tasks that have compositional structures.

However, the above grammar-based algorithms (except [32]) take symbolic inputs like the traditional language parsers. They require the action primitives/atomic actions to be first detected, then a grammar is used for high-level parsing. This limits the applicability of these algorithms. Additionally, the parser does not provide guidance for either training the classifiers or segmenting the sequences. They also lack a good approach to handle grammatically incorrect label sentences. For example, Qi et al. [5] found in the training corpus the closest sentence to the recognized sentence and applies the language parser afterward. Pirsiavash and Ramanan [32] ensures the results are grammatically correct, but it makes the grammar unnecessarily redundant (each possible segment length will make a new copy for each original grammar rule).

In our case, the proposed parsing algorithm takes sequence data of raw signals and a typical context-free grammar as input. It then generates the label sentence as well as the parse tree. All parsed label sentences are grammatically correct, and a learning objective is formulated for the classifier. Our work also serves as a bridge between connectionist and symbolic approaches, and it does not have any constraint on the low-level classifier.

Future activity prediction is a relatively new domain in computer vision. [6, 23, 24, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47] predict human trajectories/actions in various settings including complex indoor/outdoor scenes and crowded spaces. Li and Fu [43] built a probabilistic suffix tree to model the Markov dependencies between action units and thus predict future events using a compositional model. Walker et al. [41] predicted not only the future motions in the scene but also the visual appearances. In some recent work, Koppula and Saxena [48] used an anticipatory temporal conditional random field to model the spatial-temporal relations through object affordances. Jain et al. [49] proposed structural-RNN as a generic method to combine high-level spatial-temporal graphs and recurrent neural networks, which is a typical example that takes advantage of both graphical models and deep learning. Qi et al. [5] proposed a spatial-temporal And-Or graph (ST-AOG) for activity prediction. In this work, we present the generalized Earley parser, in which the recognition and prediction are naturally and tightly integrated.

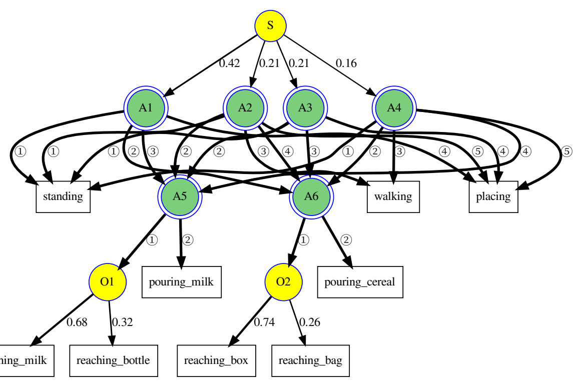


Fig. 2: An example of a temporal grammar representing the activity "making cereal". The green and yellow nodes are And-nodes (*i.e.*, production rules that represents combinations) and Or-nodes (*i.e.*, productions rules that represents alternatives), respectively. The numbers on branching edges of Or-nodes represent the branching probability. The circled numbers on edges indicate the temporal order of expansion.

3 Representation: Probabilistic Context-Free Grammars

We model complex activities by grammars, where low-level actions are terminal symbols, *i.e.*, like words in a language. In formal language theory, a *context-free grammar* (CFG) is a type of formal grammar, which contains a set of production rules that describe all possible sentences in a given formal language. In Chomsky Normal Form, a context-free grammar G is defined by a 4-tuple $G = (V, \Sigma, R, \Gamma)$ where

- V is a finite set of non-terminal symbols that can be replaced by/expanded to a sequence of symbols.
- Σ is a finite set of terminal symbols that represent actual words in a language, which cannot be further expanded.
- R is a finite set of production rules describing the replacement of symbols, typically of the form $A \to BC$ or $A \to \alpha$ for $A, B, C \in V$ and $\alpha \in \Sigma$. A production rule replaces the left-hand side non-terminal symbol by the right-hand side expression. For example, $A \to BC|\alpha$ means that A can be replaced by either BC or α.
- $\Gamma \in V$ is the start symbol (root of the grammar).

Probabilistic Context-Free Grammars (PCFGs) augment CFGs by associating each production rule with a probability. Formally, it is defined by a 5-tuple $G = (V, \Sigma, R, \Gamma, P)$, where P is the set of probabilities on production rules. Figure 2 shows an example probabilistic temporal grammar of the activity "making cereal".

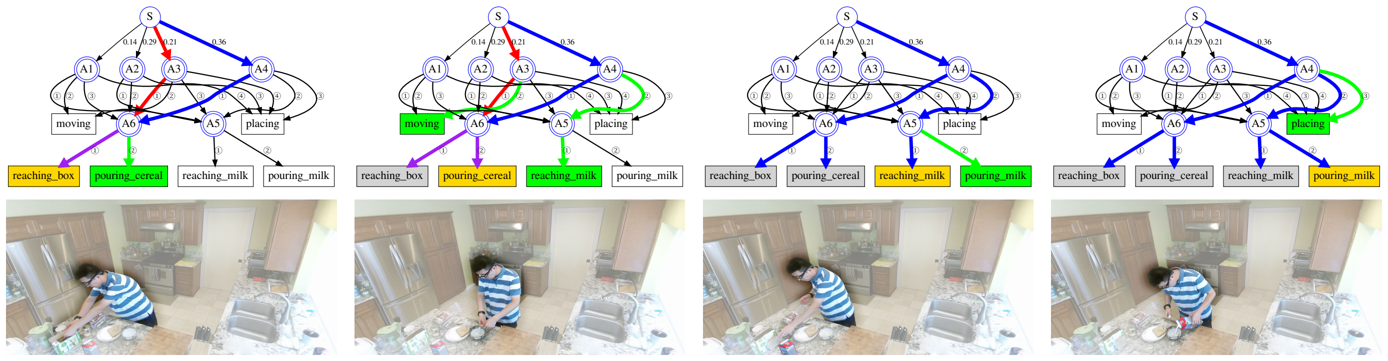


Fig. 3: An example illustrating the symbolic parsing and prediction process based on the Earley parser and detected actions. In the first two figures, the red edges and blue edges indicates two different parse graphs for the past observations. The purple edges indicate the overlap of the two possible explanations. The red parse graph is eliminated from the third figure. For the terminal nodes, yellow indicates the current observation and green indicates the next possible state(s).

Given a formal grammar, parsing is the process of analyzing a string of symbols, conforming to the production rules and generating a parse tree. A parse tree represents the syntactic structure of a string according to some context-free grammar. The root node of the tree is the grammar root. Other non-leaf nodes correspond to non-terminals in the grammar, expanded according to grammar production rules (could be expanding a combination or choosing alternatives). The leaf nodes are terminal symbols. All the leaf nodes together form a sentence in the language space described by the grammar.

The overall goal of parsing sequence data with a PCFG is to find the label sentence to best explain the input with the maximum posterior probability:

$$l^* = \operatorname*{argmax}_l p(l|\boldsymbol{x}, G). \quad (1)$$

Here $\boldsymbol{x}$ is a input sequence data of length T (*e.g.*, videos or audios) and l is a sentence of labels (*e.g.*, actions or words) of length $|l| \leq T$, where each label $k \in \{0, 1, \cdots, K\}$ corresponds to a segment of a sequence. We can clearly see the fundamental differences between traditional grammar parsers and our proposed algorithm: traditional grammar parsers take the label sequence l as the input to infer the parse tree, while our algorithm takes the original data sequence x as the input to infer the most likely parse tree.

In the following sections, we explain how to extend a traditional parsing algorithm to achieve this goal. In Section 4, we briefly review the original Earley parser and related concepts. In Section 5, we elaborate on the proposed generalized Earley parser and related algorithms based on the general concepts in Section 3 and Section 4.

4 Earley Parser

In this section, we briefly review the original Earley parser [1], a classic grammar parsing algorithm with useful concepts that will be extended in the generalized Earley parser. An illustrative example is shown in Figure 4 to run through the algorithm. We then discuss how the original Earley parser can be applied to event parsing and its drawbacks.

Earley parser is an algorithm for parsing sentences of a given context-free language. In the following descriptions, α, β, and γ represent any string of terminals/nonterminals (including the empty string ϵ), A and B represent single nonterminals, and a represents a terminal symbol. We adopt Earley's dot notation: for production rule of form $A \to \alpha\beta$, the notation $A \to \alpha \cdot \beta$ means α has been parsed and β is expected.

Input position n is defined as the position after accepting the nth token, and input position 0 is the position prior to input. At each input position m, the parser generates a state set $S(m)$. Each state is a tuple $(A \to \alpha \cdot \beta, i)$, consisting of

- The production currently being matched $(A \to \alpha\beta)$.
- The dot: the current position in that production.
- The position i in the input at which the matching of this production began: the position of origin.

Seeded with $S(0)$ containing only the top-level rule, the parser then repeatedly executes three operations: prediction, scanning and completion:

- **Prediction**: for every state in $S(m)$ of the form $(A \to \alpha \cdot B\beta, i)$, where i is the origin position as above, add $(B \to \cdot\gamma, m)$ to $S(m)$ for every production in the grammar with B on the left-hand side (*i.e.*, $B \to \gamma$).
- **Scanning**: if a is the next symbol in the input stream, for every state in $S(m)$ of the form $(A \to \alpha \cdot a\beta, i)$, add $(A \to \alpha a \cdot \beta, i)$ to $S(m+1)$.
- **Completion**: for every state in $S(m)$ of the form $(A \to \gamma\cdot, j)$, find states in $S(j)$ of the form $(B \to \alpha \cdot A\beta, i)$ and add $(B \to \alpha A \cdot \beta, i)$ to $S(m)$.

In this process, duplicate states are not added to the state set. These three operations are repeated until no new states can be added to the set. The Earley parser executes in $O(n^2)$ for unambiguous grammars regarding the string length n, and $O(n)$ for almost all $LR(k)$ grammars.

The original Earley parser inspires a way to do event parsing and prediction from videos [5]. The video can be first processed by a classifier to be segmented and labeled by actions, thus generating a label sentence. We can apply the Earley parser to parse the sentence to get a partial parse tree. The tree can be partial, since the sentence representing the activity might not be complete. Then action prediction can naturally be accomplished by looking that the open Earley

$\Gamma \to R$	1.0		$N \to$ "0"	0.3
$R \to N$	0.4		$N \to$ "1"	0.7
$R \to N$"+"N	0.6			

(a) The input grammar. It contains a root symbol Γ, two non-terminal symbols R and N, three terminal symbols $0, 1$ and $+$. The number to the right of each production rule is the corresponding probability.

state	rule	comment
$S(0)$		
(0)	$\Gamma \to \cdot R$	start rule
(1)	$R \to \cdot N$	predict: (0)
(2)	$R \to \cdot N + N$	predict: (0)
(3)	$N \to \cdot 0$	predict: (1), (2)
(4)	$N \to \cdot 1$	predict: (1), (2)
$S(1)$		
(0)	$N \to 0 \cdot$	scan: $S(0)(3)$
(1)	$R \to N \cdot$	complete: (0) and $S(0)(1)$
(2)	$R \to N \cdot +N$	complete: (0) and $S(0)(2)$
(3)	$\Gamma \to R \cdot$	complete: (1) and $S(0)(0)$
$S(2)$		
(0)	$R \to N + \cdot N$	scan: $S(1)(2)$
(1)	$N \to \cdot 0$	predict: (0)
(2)	$N \to \cdot 1$	predict: (0)
$S(3)$		
(0)	$N \to 1 \cdot$	scan: $S(2)(2)$
(1)	$R \to N + N \cdot$	complete: (0) and $S(2)(0)$
(2)	$\Gamma \to R \cdot$	complete: (1) and $S(0)(0)$

(b) A run-through for input string "$0 + 1$".

Fig. 4: An illustrative example of the original Earley parser.

states generated by the "prediction" operation. An example is shown in Figure 3.

However, this process can be problematic. Since the Earley parser takes symbols as input, it has little guidance to help the segmentation process that happens in the frame level. A more severe problem is that the segmentation and labeling process often generates sentences that are grammatically incorrect, *i.e.*, not in the language space described by the grammar. Thus the sentence cannot be parsed by the parser. In such cases, extra efforts are needed to modify the label sentence. One way to address that is sampling sentences from the language and find the closest alternatives [5]. There also exist work in computational linguistics [50, 51, 52] that address the problem of parsing grammatically incorrect sentences. However, these methods still operate in the symbolic space and does not provide much guidance for frame-level inference. To solve these problems, we propose the generalized Earley parser (detailed in Section 5) that directly takes sequence data as input and generates symbolic parse trees and predictions.

5 Generalized Earley Parser

In this section, we introduce the proposed generalized Earley parser. Instead of taking symbolic sentences as input, we aim to design an algorithm that can parse raw sequence data $\boldsymbol{x}$ of length T into a sentence l of labels of length $|l| \leq T$, where each label $k \in \{0, 1, \cdots, K\}$ corresponds to a segment of a sequence.

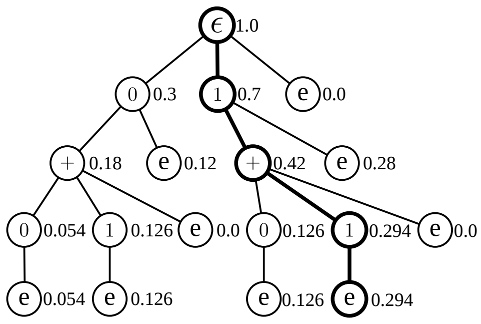


Fig. 5: Grammar prefix probabilities computed according to the grammar in Figure 4. The numbers next to the tree nodes are prefix probabilities according to the grammar. The transition probabilities can be easily computed from this tree, *e.g.*, $p(\text{"1"}|\text{"1+"}, G) = p(\text{"1+1"}...|G)/p(\text{"1+"}...|G) = 0.294/0.42 = 0.7$.

To achieve that, a classifier (*e.g.*, a neural network) is first applied to each sequence $\boldsymbol{x}$ to get a $T \times K$ probability matrix $\boldsymbol{y}$ (*e.g.*, softmax activations of the neural network), with y_t^k representing the probability of frame t being labeled as k. The proposed generalized Earley parser takes $\boldsymbol{y}$ as input and outputs the sentence l^* that best explains the data according to a grammar G of Chomsky normal form.

Now we discuss how we generalize the Earley parser to run on the output of a classifier, *i.e.*, the probability matrix. The core idea is to use the original Earley parser to help construct a prefix tree according to the grammar. The best solution is found by performing a heuristic search in this tree, where the heuristic is computed based on the probability matrix given by the classifier.

Figure 5 and Figure 6c shows example prefix trees for the grammar in Figure 4. A prefix tree is composed of three types of nodes. 1) The root node of the "empty" symbol ϵ represents the start of a sentence. 2) The non-leaf nodes (except the root node) correspond to terminal symbols in the grammar. A path from the root node to any non-leaf node represents a partial sentence (prefix). 3) The leaf nodes e are terminations that represent ends of sentences.

To find the best label sentence for a probability matrix, we perform a heuristic search in the prefix expanded according to the grammar: each node in the tree is associated with a probability, and the probabilities prioritize the nodes to be expanded in the prefix tree. The parser finds the best solution when it expands a termination node in the tree. It then returns the current prefix string as the best solution.

We compute two different heuristic probabilities for non-leaf nodes and leaf nodes. For non-leaf nodes, the heuristic is a prefix probability $p(l...|x_{0:T})$: the probability that the current path is the prefix for the label sentence. In other words, it measures the probability that $\exists t \in [0, T]$, the current path l is the label for frame $x_{0:t}$. For leaf nodes e, the heuristic $p(l|x_{0:T})$ is a parsing probability: the probability that the current path l is the label sentence for $x_{0:T}$. The computation for $p(l|x_{0:T})$ and $p(l...|x_{0:T})$ are based on the input probability matrix $\boldsymbol{y}$. The formulation is derived in details in Section 5.2.

This heuristic search generalizes the Earley parser to

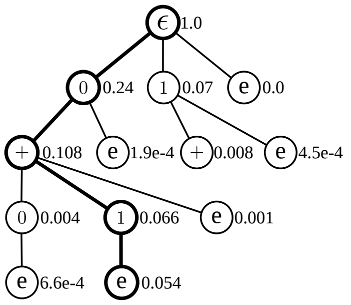
$\Gamma \rightarrow R$	1.0
$R \rightarrow N$	0.4
$R \rightarrow N$"+"N	0.6
$N \rightarrow$ "0"	0.3
$N \rightarrow$ "1"	0.7

frame	"0"	"1"	"+"
0	0.8	0.1	0.1
1	0.8	0.1	0.1
2	0.1	0.1	0.8
3	0.1	0.8	0.1
4	0.1	0.8	0.1

(a) Left: input grammar. Right: input probability matrix.

Frame	ϵ	0	1	0 +	1 +	0 + 0	0 + 1
0	0.000	0.240	0.070	0.000	0.000	0.000	0.000
1	0.000	0.192	0.007	0.014	0.004	0.000	0.000
2	0.000	0.019	7.0e-04	0.104	0.007	4.3e-04	0.001
3	0.000	0.002	5.6e-04	0.012	7.1e-04	0.003	0.059
4	0.000	1.9e-04	4.5e-04	0.001	1.1e-04	6.6e-04	0.054
prefix	1.000	0.240	0.070	0.108	0.008	0.004	0.066

(b) Cached probabilities



(c) Prefix tree

state #	rule	μ	ν	prefix	comment
$S(0,0) : l =$ "ϵ"$, p(l\|G) = 1.000, p(l\|x,G) = 0.000, p(l...\|x,G) = 1.000$					
(0)	$\Gamma \rightarrow \cdot R$	1.000	1.000	"ϵ"	start rule
(1)	$R \rightarrow \cdot N$	0.400	0.400	"ϵ"	predict: (0)
(2)	$R \rightarrow \cdot N + N$	0.600	0.600	"ϵ"	predict: (0)
(3)	$N \rightarrow \cdot 0$	0.300	0.300	"ϵ"	predict: (1),(2)
(4)	$N \rightarrow \cdot 1$	0.700	0.700	"ϵ"	predict: (1),(2)
$S(1,0) : l =$ "0"$, p(l\|G) = 0.300, p(l\|x,G) = 1.9e-04, p(l...\|x,G) = 0.240$					
(0)	$N \rightarrow 0\cdot$	0.300	0.300	"0"	scan: S(0, 0)(3)
(1)	$R \rightarrow N\cdot$	0.120	0.120	"0"	complete: (0) and S(0, 0)(1)
(2)	$R \rightarrow N \cdot +N$	0.180	0.180	"0"	complete: (0) and S(0, 0)(2)
(3)	$\Gamma \rightarrow R\cdot$	0.120	0.120	"0"	complete: (1) and S(0, 0)(0)
$S(1,1) : l =$ "1"$, p(l\|G) = 0.700, p(l\|x,G) = 4.5e-04, p(l...\|x,G) = 0.070$					
(0)	$N \rightarrow 1\cdot$	0.700	0.700	"1"	scan: S(0, 0)(4)
(1)	$R \rightarrow N\cdot$	0.280	0.280	"1"	complete: (0) and S(0, 0)(1)
(2)	$R \rightarrow N \cdot +N$	0.420	0.420	"1"	complete: (0) and S(0, 0)(2)
(3)	$\Gamma \rightarrow R\cdot$	0.280	0.280	"1"	complete: (1) and S(0, 0)(0)
$S(2,0) : l =$ "0 +"$, p(l\|G) = 0.180, p(l\|x,G) = 0.001, p(l...\|x,G) = 0.108$					
(0)	$R \rightarrow N + \cdot N$	0.180	0.180	"0+"	scan: S(1, 0)(2)
(1)	$N \rightarrow \cdot 0$	0.054	0.300	"0+"	predict: (0)
(2)	$N \rightarrow \cdot 1$	0.126	0.700	"0+"	predict: (0)
$S(2,1) : l =$ "1 +"$, p(l\|G) = 0.420, p(l\|x,G) = 1.1e-04, p(l...\|x,G) = 0.008$					
(0)	$R \rightarrow N + \cdot N$	0.420	0.420	"1+"	scan: S(1, 1)(2)
$S(3,0) : l =$ "0 + 0"$, p(l\|G) = 0.054, p(l\|x,G) = 6.6e-04, p(l...\|x,G) = 0.004$					
(0)	$N \rightarrow 0\cdot$	0.054	0.300	"0 + 0"	scan: S(2, 0)(1)
$S(3,1) : l =$ "0 + 1"$, p(l\|G) = 0.126, \boldsymbol{p(l\|x,G) = 0.054}, p(l...\|x,G) = 0.066$					
(0)	$N \rightarrow 1\cdot$	0.126	0.700	"0 + 1"	scan: S(2, 0)(2)
(1)	$R \rightarrow N + N\cdot$	0.126	0.126	"0 + 1"	complete: (0) and S(2, 0)(0)
(2)	$\Gamma \rightarrow R\cdot$	0.126	0.126	"0 + 1"	complete: (1) and S(0, 0)(0)

Final output: $l^* =$ "0 + 1" with probability 0.054

(d) A run-through of the algorithm

Fig. 6: An example of the generalized Earley parser. A classifier is applied to a 5-frame signal and outputs a probability matrix (a) as the input to our algorithm. The proposed algorithm expands a grammar prefix tree (c), where e represents termination. It finally outputs the best label "$0 + 1$" with probability 0.054. The probabilities of children nodes do not sum to 1 since grammatically incorrect nodes are eliminated from the search. The search process is illustrated in Figure 7.

parse the probability matrix. Specifically, the scan operation in the Earley parser essentially expands a new node in the grammar prefix tree. We organize the states into state sets by the partial sentence (prefix) each state represents. Instead of matching the sentence to the symbolic input, we now process state sets according to their prefix probabilities.

5.1 Parsing Operations

We now describe the details of the parsing operations. Each scan operation will create a new state set $S(m,n) \in S(m)$, where m is the length of the scanned string, n is the total number of the terminals that have been scanned at position m. This can be thought of as creating a new node in the prefix tree, and $S(m)$ is the set of all created nodes at level m. A priority queue q is kept for state sets for prefix search. Scan operations will push the newly created set into the queue with priority $p(l...)$, where l is the parsed string of the state being scanned. For brevity, we use $p(l...)$ as a shorthand for $p(l...|x_{0:t})$ when describing the algorithm.

Each state is a tuple $(A \rightarrow \alpha \cdot \beta, i, j, l, p(l...))$ augmented from the original Earley parser by adding $j, l, p(l...)$. Here l is the parsed string of the state, and i, j are the indices of the set that this rule originated. The parser then repeatedly executes three operations: prediction, scanning, and completion modified from Earley parser:

- **Prediction**: for every state in $S(m,n)$ of the form $(A \rightarrow \alpha \cdot B\beta, i, j, l, p(l...))$, add $(B \rightarrow \cdot\Gamma, m, n, l, p(l...))$ to $S(m,n)$ for every production in the grammar with B on the left-hand side.
- **Scanning**: for every state in $S(m,n)$ of the form $(A \rightarrow \alpha \cdot a\beta, i, j, l, p(l...))$, append the new terminal a to l and compute the probability $p((l+a)...)$. Create a new set $S(m+1, n')$ where n' is the current size of $S(m+1)$. Add $(A \rightarrow \alpha a \cdot \beta, i, j, l+a, p((l+a)...))$ to $S(m+1, n')$. Push $S(m+1, n')$ into q with priority $p((l+a)...)$.
- **Completion**: for every state in $S(m,n)$ of the form $(A \rightarrow \Gamma\cdot, i, j, l, p(l...))$, find states in $S(i,j)$ of the form $(B \rightarrow \alpha \cdot A\beta, i', j', l', p(l'...))$ and add $(B \rightarrow \alpha A \cdot \beta, i', j', l, p(l...))$ to $S(m,n)$.

This parsing process is efficient since we do not need to search through the entire tree. As shown in Figure 6 and Algorithm 1, the best label sentence l is returned when the probability of termination is larger than any other prefix probabilities. As long as the parsing and prefix probabilities are computed correctly, it is guaranteed to return the best solution.

The original Earley parser is a special case of the generalized Earley parser. Intuitively, for any input sentence to Earley parser, we can always convert it to one-hot vectors and apply the proposed algorithm. On the other hand, the original Earley parser cannot be applied to segmented

Algorithm 1: Generalized Earley Parser

```
Input  : Grammar $G$, probability matrix $y$
Output: Best label string $l^*$
/* For brevity, we denote $p(\cdot; x_{0:t})$ as $p(\cdot)$ */
/* Initialization */
1  $S(0,0) = \{(\Gamma \to \cdot R, 0, 0, \epsilon, 1.0)\}$
2  $q = priorityQueue()$
3  $q.push(1.0, (0, 0, \epsilon, S(0,0)))$
4  while $(m, n, l^-, currentSet) = q.pop()$ do
5      for $s = (r, i, j, l, p(l...)) \in currentSet$ do
6          if $p(l) > p(l^*)$: $l^* = l$ then $l^* = l$
7          if $r$ is $(A \to \alpha \cdot B\beta)$ then          // predict
8              for each $(B \to \Gamma)$ in $G$ do
9                  $r' = (B \to \cdot\Gamma)$
10                 $s' = (r', m, n, l, p(l...))$
11                 $S(m,n).add(s')$
12             end
13         end
14         else if $r$ is $(A \to \alpha \cdot a\beta)$ then          // scan
15             $r' = (A \to \alpha a \cdot \beta)$
16             $m' = m + 1, n' = |S(m+1)|$
17             $s' = (r', i, j, l + a, p((l + a)...))$
18             $S(m', n').add(s')$
19             $q.push(p((l + a)...), (m', n', S(m', n')))$
20         end
21         else if $r$ is $(B \to \Gamma\cdot)$ then          // complete
22             for each $((A \to \alpha \cdot B\beta), i', j')$ in $S(i, j)$ do
23                 $r' = (A \to \alpha B \cdot \beta)$
24                 $s' = (r', i', j', l, p(l...))$
25                 $S(m,n).add(s')$
26             end
27         end
28         if $p(l^-) > p(l), \forall$ un-expanded $l$ then return $l^*$
29     end
30 end
31 return $l^*$
```

one-hot vectors since the labels are often grammatically incorrect. Hence we have the following proposition.

Proposition 1. *Earley parser is a special case of the generalized Earley parser.*

Proof. Let $L(G)$ denote the language of grammar G, $h(\cdot)$ denote a one-to-one mapping from a label to a one-hot vector. $L(G)$ is the input space for Earley parser. $\forall\ l \in L(G)$, the generalized Earley parser accepts $h(l)$ as input. Therefore the proposition follows. □

Here we emphasize two important distinctions of our method to traditional probabilistic parsers with prefix probabilities. i) In traditional parsers, the prefix probability is the probability of a string being a prefix of some strings generated by a grammar (top-down grammar prior). In our case, the parser computes the bottom-up data likelihood. We further extend this to a posterior that integrates these two in Section 5.3. ii) Traditional parsers only maintain a parse tree, while our algorithm maintains both a parse tree and a prefix tree. The introduction of the prefix tree into the parser enables us to efficiently search in the grammar according to a desired heuristic.

TABLE 1: Summary of notations used for parsing & prefix probability formulation.

$x_{0:t}$	input frames from time 0 to t
l	a label sentence
k	the last label in l
l^-	the label sentence obtained by removing the last label k from the label sentence l
y_t^k	the probability for frame t to be labelled as k
$p(l\|x_{0:t})$	parsing probability of l for $x_{0:t}$
$p(l...\|x_{0:t})$	prefix probability of l for $x_{0:t}$

5.2 Parsing & Prefix Probability Formulation

Table 1 summarizes the notations we use in this section. The parsing probability $p(l|x_{0:T})$ is computed in a dynamic programming fashion. Let k be the last label in l. For $t = 0$, the probability is initialized by:

$$p(l|x_0) = \begin{cases} y_0^k & l \text{ contains only } k, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Let l^- be the label sentence obtained by removing the last label k from the label sentence l. For $t > 0$, the last frame t must be classified as k. The previous frames can be labeled as either l or l^-. Then we have:

$$p(l|x_{0:t}) = y_t^k[p(l|x_{0:t-1}) + p(l^-|x_{0:t-1})], \tag{3}$$

where $p(l|x_{0:t-1})$ corresponds to the possibility that frame $t-1$ is also labelled as k, and $p(l^-|x_{0:t-1})$ accounts for the possibility that label k starts from frame t. It is worth mentioning that when y_t^k is wrongly given as 0, the dynamic programming process will have trouble correcting the mistake. Even if $p(l^-|x_{0:t-1})$ is high, the probability $p(l|x_{0:t})$ will be 0. Fortunately, since the softmax function is usually adopted to compute y, y_t^k will not be 0 and the solution will be kept for further consideration.

Then we compute the prefix probability $p(l...|x_{0:T})$ based on $p(l^-|x_{0:t})$. For l to be the prefix, the transition from l^- to l can happen at any frame $t \in \{0, \cdots, T\}$. Once the label k is observed (the transition happens), l becomes the prefix and the rest frames can be labeled arbitrarily. Hence the probability of l being the prefix is:

$$p(l...|x_{0:T}) = p(l|x_0) + \sum_{t=1}^{T} y_t^k p(l^-|x_{0:t-1}). \tag{4}$$

In practice, the probability $p(l|x_{0:t})$ decreases exponentially as t increases and will soon lead to numeric underflow. To avoid this, the probabilities need to be computed in log space:

$$\begin{aligned}\log p(l|x_{0:t}) = {}& \log(y_t^k) + d + \\ & \log\{\exp[\log p(l|x_{0:t-1}) - d] + \exp[\log p(l^-|x_{0:t-1}) - d]\},\end{aligned} \tag{5}$$

where d is a constant number and is usually set to be $\max\{\log(y_t^k), \log p(l|x_{0:t-1}), \log p(l^-|x_{0:t-1})\}$. The time complexity of computing the probabilities is $O(T)$ for each sentence l because $p(l^-|x_{0:t})$ are cached. The worst case complexity of the entire parsing is $O(T|G|)$.

5.3 Incorporating Grammar Prior

For PCFGs, we can integrate the grammar prior of the sentence l into the above formulation to obtain a posterior parsing probability. The basic idea is that we can compute a "transition probability" of appending a new symbol to the current sentence. This probability will be multiplied to the parsing probability when we append a new symbol.

To compute a transition probability $p(k|l^-, G)$, we can first compute the prefix probabilities $p(l^-_{...}|G)$ and $p(l...|G)$ according to the grammar. Then the transition probability is given by:

$$p(k|l^-, G) = \frac{p(l...|G)}{p(l^-_{...}|G)}. \tag{6}$$

An example is shown in Figure 5 for a better intuition. The computation of this grammar prefix probability will be detailed in Section 5.3.1. There are two important remarks to make here. 1) This prior prefix probability is different from the prefix probability based on the likelihood. The prior is the probability that a string is the prefix of a sentence in the language defined by the grammar, without seeing any data; the likelihood is the probability that a string is the prefix of a video's label. 2) This grammar-based transition probability is non-Markovian, since the new symbol is conditioned on the entire history string that has a variable length.

Now, incorporating the grammar transition probability, for $t = 0$, the probability is initialized by:

$$p(l|x_0, G) \propto \begin{cases} p(k|\epsilon, G)\, y_0^k & l \text{ contains only } k, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $p(k|\epsilon, G)$ is the probability of appending k to the empty string ϵ, which is equivalent to $p(k...|G)$ or $p(l...|G)$. Notice that the equal sign is replaced by $\propto$ since the right hand side should be normalized by the prior $p(x_0)$ to get the correct posterior.

Whenever we append a new symbol to our sentence, we multiply the probability by the transition probability. Hence for $t > 0$ we have:

$$p(l|x_{0:t}, G) \propto y_t^k [p(l|x_{0:t-1}, G) + p(k|l^-, G) p(l^-|x_{0:t-1}, G)]. \tag{8}$$

Comparing to Eq. 3, we multiply the second term by $p(k|l^-, G)$ to account for the transition to symbol k.

Finally the posterior probability of l being the prefix of the label sentence for data x is:

$$p(l...|x_{0:T}, G) = p(l|x_0, G) + \sum_{t=1}^{T} p(k|l^-, G) y_t^k p(l^-|x_{0:t-1}, G). \tag{9}$$

5.3.1 *Grammar Prefix Probability*

The derivation of the grammar prefix probability with Earley parser [53] can be achieved by augmenting the Earley states with two additional variables: forward probability μ and inner probability ν. For a state S, the forward probability μ is the probability of all parses that lead to S, the inner probability ν is the probability of all parses expanded from S. In other words, μ is the probability of the prefix before S, and ν is the probability of the partial string parsed by S. Assuming that the grammar is not left-recursive, these two terms can be computed effiently during the Earley parsing process:

- **Prediction**. For $(A \rightarrow \alpha \cdot B\beta, i, [\mu, \nu]) \Rightarrow (B \rightarrow \cdot\gamma, m, [\mu', \nu'])$, the new probabilities are given by
$$\mu' += \alpha \cdot P(B \rightarrow \gamma), \nu' = P(B \rightarrow \gamma).$$
- **Scanning**. For $(A \rightarrow \alpha \cdot a\beta, i, [\mu, \nu]) \Rightarrow (A \rightarrow \alpha a \cdot \beta, i, [\mu', \nu'])$, we have
$$\mu' = \mu, \nu' = \nu.$$
- **Completion**. For $(A \rightarrow \gamma\cdot, j, [\mu'', \nu''])$ and $(B \rightarrow \alpha \cdot A\beta, i, [\mu, \nu]) \Rightarrow (B \rightarrow \alpha A \cdot \beta, i, [\mu', \nu'])$, we have
$$\mu' += \mu \cdot \nu'', \nu' = \nu \cdot \nu''.$$

Finally, the prefix probability of a string is given by the sum of forward probabilities over all scanned states. A run-through example of the generalized Earley parser with grammar prior is shown in Figure 6 abd Figure 7.

5.4 Segmentation and Labeling

The generalized Earley parser gives us the best grammatically correct label sentence l to explain the sequence data, which takes all possible segmentations into consideration. Therefore the probability $p(l|x_{0:T})$ is the summation of probabilities of all possible segmentations. Let $p(l|y_{0:e})$ be the probability of the best segmentation based on the classifier output y for sentence l. We perform a maximization over different segmentations by dynamic programming to find the best segmentation:

$$p(l|y_{0:e}) = \max_{b<e} p(l^-|y_{0:b}) \prod_{t=b}^{e} y_t^k, \tag{10}$$

where e is the time frame that l ends and b is the time frame that l^- ends. The best segmentation can be obtained by backtracing the above probability. Similar to the previous probabilities, this probability needs to be computed in log space as well. The time complexity of the segmentation and labeling is $O(T^2)$.

5.5 Future Label Prediction

We consider two types of future label predictions: 1) segment-wise prediction that predicts the next segment label at each time t, and 2) frame-wise prediction that predicts the labels for the future δt frames.

5.5.1 *Segment-wise Prediction*

Given the parsing result l, we can make grammar-based top-down predictions for the next label z to be observed. The predictions are naturally obtained by the predict operation in the generalized Earley parser, and it is inherently an online prediction algorithm.

To predict the next possible symbols at current position (m, n), we search through the states $S(m, n)$ of the form $(X \rightarrow \alpha \cdot z\beta, i, j, l, p(l...))$, where the first symbol z after the current position is a terminal node. The predictions Σ are then given by the set of all possible z:

$$\Sigma = \{z : \exists s \in S(m, n), s = (X \rightarrow \alpha \cdot z\beta, i, j, l, p(l...))\}. \tag{11}$$

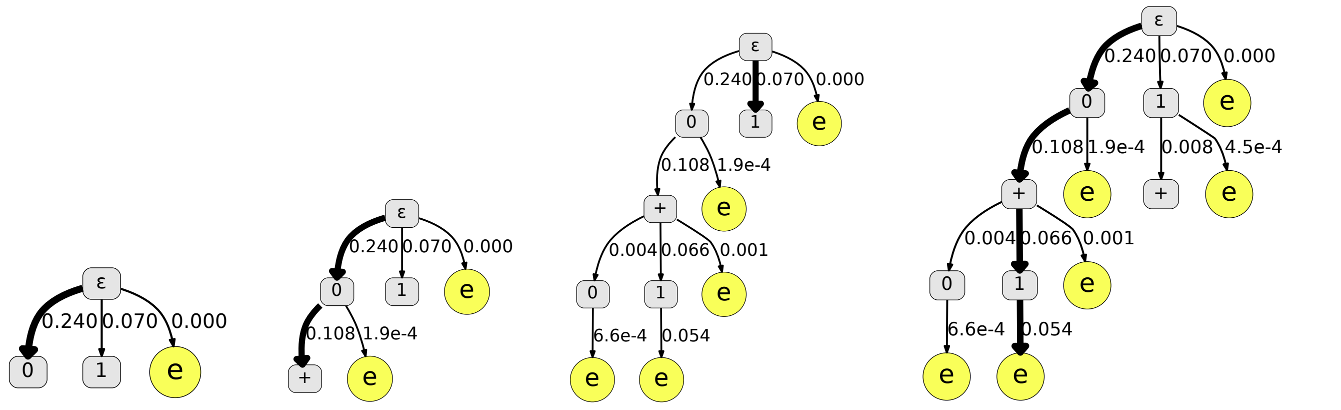


Fig. 7: An illustration of the parsing process of the example in Figure 6. It performs a heuristic search in the prefix tree according to the prefix/parsing probability. It iteratively expands the tree and computes the probabilities as it expands the tree. The search ends when it hits a parsing terminal e. The paths in bold indicate the best candidates at each search step.

The probability of each prediction is then given by the parsing likelihood of the sentence constructed by appending the predicted label z to the current sentence l. Assuming that the best prediction corresponds to the best parsing result, the goal is to find the best prediction z^* that maximizes the following conditional probability as parsing likelihood:

$$z^* = \operatorname*{argmax}_{z \in \Sigma} p(z, l|G). \tag{12}$$

For a grammatically complete sentence u, the parsing likelihood is simply the Viterbi likelihood [54] given by the probabilistic context-free grammar. For an incomplete sentence l of length $|l|$, the parsing likelihood is given by the grammar prefix probability. Hence they are both the forward probability computed in Section 5.3.1. We can also integrate top-down and bottom-up inference for segment-wise prediction. A classifier can be trained to predict the next segment label, and it can be combined with the grammar prior probability for better predictions.

5.5.2 Frame-wise Prediction

Frame-wise future label prediction is rather straightforward using the generalized Earley parser. We first run activity detection on the input videos, and we sample the duration of the current action. Based on the segment-wise prediction, we can further sample the duration for future segments, thus obtaining frame-wise future predictions according to the prediction range.

5.5.3 Maximum Likelihood Estimation for Prediction

We are interested in finding the best grammar and classifier that give us the most accurate segment-wise predictions based on the generalized Earley parser. Let G be the grammar, f be the classifier, and D be the set of training examples. The training set consists of pairs of complete or partial data sequence $\boldsymbol{x}$ and the corresponding label sequence for all the frames in $\boldsymbol{x}$. By merging consecutive labels that are the same, we can obtain partial label sentences l and predicted labels z. Hence we have $D = \{(\boldsymbol{x}, l, z)\}$. The best grammar G^* and the best classifier f^* together minimizes the prediction loss:

$$G^*, f^* = \operatorname*{argmin}_{G,f} \mathcal{L}_{pred}(G, f), \tag{13}$$

where the prediction loss is given by the negative log likelihood of the predictions over the entire training set:

$$\begin{aligned} \mathcal{L}_{pred}(G, f) &= - \sum_{(\boldsymbol{x},l,z)\in D} \log p(z|\boldsymbol{x}) \\ &= - \sum_{(\boldsymbol{x},l,z)\in D} \{\underbrace{\log p(z|l, G)}_{\text{grammar}} + \underbrace{\log p(l|\boldsymbol{x})}_{\text{classifier}}\}. \end{aligned} \tag{14}$$

Given the intermediate variable l, the loss is decomposed into two parts that correspond to the induced grammar and the trained classifier, respectively. Let $u \in \{l\}$ be the complete label sentences in the training set (*i.e.*, the label sentence for a complete sequence $\boldsymbol{x}$). The best grammar maximizes the following probability:

$$\prod_{(z,l)\in D} p(z|l, G) = \prod_{(z,l)\in D} \frac{p(z, l|G)}{p(l|G)} = \prod_{u \in D} p(u|G), \tag{15}$$

where denominators $p(l|G)$ are canceled by the previous numerator $p(z, l^-|G)$, and only the likelihood of the complete sentences remain. Therefore inducing the best grammar that gives us the most accurate future prediction is equivalent to the maximum likelihood estimation (MLE) of the grammar for complete sentences in the dataset. This finding lets us turn the problem (induce the grammar that gives the best future prediction) into a standard grammar induction problem, which can be solved by existing algorithms, *e.g.*, [55] and [56].

The best classifier minimizes the second term of Eq. 14:

$$\begin{aligned} f^* &= \operatorname*{argmin}_{f} - \sum_{(\boldsymbol{x},l,z)\in D} \log p(l|\boldsymbol{x}) \\ &\approx \operatorname*{argmin}_{f} - \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \sum_{k} y_k \log(\hat{y_k}), \end{aligned} \tag{16}$$

where $p(l|\boldsymbol{x})$ can be maximized by the CTC loss [57]. In practice, it can be substituted by the commonly adopted cross entropy loss for efficiency. Therefore we can directly apply generalized Earley parser to outputs of general detectors/classifiers for parsing and prediction.

6 Human Activity Parsing and Prediction

We evaluate our method on the task of human activity detection and prediction. We present and discuss our experiment results on three datasets, CAD-120 [2], Watch-n-Patch [3], and Breakfast [4], for comparisons with state-of-the-art methods and evaluation of the robustness of our approach. CAD-120 is the dataset that most existing prediction algorithms are evaluated on. It contains videos of daily activities that are long sequences of sub-activities. Watch-n-Patch is a daily activity dataset that features forgotten actions. Breakfast is a dataset that contains long videos of daily cooking activities. Results show that our method performs well on both activity detection and activity prediction.

6.1 Grammar Induction

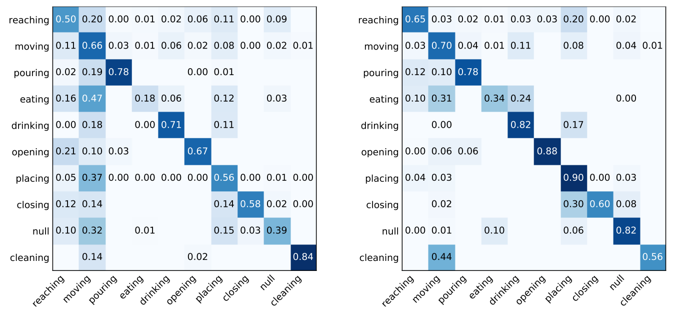
In both experiments, we used a modified version of the ADIOS (automatic distillation of structure) [55] grammar induction algorithm to learn the event grammar. The algorithm learns the production rules by generating significant patterns and equivalent classes. The significant patterns are selected according to a context-sensitive criterion defined regarding local flow quantities in the graph: two probabilities are defined over a search path. One is the right-moving ratio of fan-through (through-going flux of path) to fan-in (incoming flux of paths). The other one, similarly, is the left-going ratio of fan-through to fan-in. The criterion is described in detail in [55].

The algorithm starts by loading the corpus of activity onto a graph whose vertices are sub-activities, augmented by two special symbols, begin and end. Each event sample is represented by a separate path over the graph. Then it generates candidate patterns by traversing a different search path. At each iteration, it tests the statistical significance of each subpath to find significant patterns. The algorithm then finds the equivalent classes that are interchangeable. At the end of the iteration, the significant pattern is added to the graph as a new node, replacing the subpaths it subsumes. We favor shorter patterns in our implementation.

6.2 Experiment on CAD-120 Dataset

Dataset. The CAD-120 dataset [2] is a standard dataset for human activity prediction. It contains 120 RGB-D videos of four different subjects performing 10 high-level activities, where each high-level activity was performed three times with different objects. It includes a total of 61,585 total video frames. Each video is a sequence of sub-activities involving 10 different sub-activity labels. The videos vary from subject to subject regarding the lengths and orders of the sub-activities as well as the way they executed the task.

Evaluation metrics. We use the following metrics to evaluate and compare the algorithms. 1) Frame-wise detection accuracy of sub-activity labels for all frames. 2) Frame-wise (future 3s) online prediction accuracy. We compute the frame-wise accuracy of prediction of the sub-activity labels of the future 3s (using the frame rate of 14Hz as reported in [2]). The predictions are made online at each frame t, *i.e.*, the algorithms only see frame 0 to t and predicts the labels of frame $t+1$ to $t+\delta t$. 3) Segment-wise online prediction accuracy. At each frame t, the algorithm predicts the sub-activity label of the next video segment.



(a) Future 3s (b) Next segment

Fig. 8: Confusion matrices for predictions on CAD-120.

We consider the overall micro accuracy (P/R), macro precision, macro recall and macro F1 score for all evaluation metrics. Micro accuracy is the percentage of correctly classified labels. Macro precision and recall are the average of precision and recall respectively for all classes.

Comparative methods. We compare the results for the following methods:

1) KGS [2]: a Markov random field model where the nodes represent objects and sub-activities, and the edges represent the spatial-temporal relationships. Future frames are predicted based on the transition probabilities given the inferred label of the last frame.

2) Anticipatory temporal CRF (ATCRF) [48]: one of the state-of-the-art prediction methods. It is an anticipatory temporal conditional random field that models the spatial-temporal relations through object affordances. Future frames are predicting by sampling a spatial-temporal graph.

3) ST-AOG [5]: a spatial-temporal And-Or graph (ST-AOG) that uses a symbolic context-free grammar to model activity sequences. This sets up a comparison between our proposed method and methods that use traditional probabilistic parsers. Since traditional parsers operate on symbolic data, extra efforts need to be done first to extract symbols from sequence data. In this comparative method, the videos are first segmented and labeled by classifiers; the predictions are then made by the original Earley parser.

4) Multilayer Perceptron (MLP): a simple frame-wise detection classifier based on multilayer perceptron. A sub-activity label for every input frame feature is generated for every input frame feature.

5) MLP + GEP: the proposed generalized Earley parser (GEP) applied to the classifier output generated by a multi-layer perceptron for detection.

6) Bidirectional LSTM (Bi-LSTM): a simple frame-wise detection classifier based on LSTM. It outputs a sub-activity label for every input frame feature.

7) LSTM: a simple prediction classifier. We do not use the Bi-LSTM for prediction since the model should not see the future frames. For the future 3s (frame-wise) prediction, the LSTM is trained to output the label for frame $t + 3$s for

TABLE 2: Detection results on CAD-120.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
KGS [2]	68.2	71.1	62.2	66.4
ATCRF [48]	70.3	74.8	66.2	70.2
ST-AOG + Earley [5]	76.5	77.0	75.2	76.1
MLP	67.2	58.7	51.5	51.1
MLP + GEP	73.8	72.8	61.1	61.0
Bi-LSTM	76.2	78.5	74.5	74.9
Bi-LSTM + GEP	**79.4**	**87.4**	**77.0**	**79.7**

TABLE 3: Future 3s prediction results on CAD-120.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
KGS [2]	28.6	–	–	11.1
ATCRF [48]	49.6	–	–	40.6
ST-AOG + Earley [5]	55.2	**56.5**	**56.6**	**56.6**
LSTM	49.4	40.9	37.3	37.8
LSTM + GEP	**57.1**	52.3	54.1	52.3

TABLE 4: Segment prediction results on CAD-120.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
ST-AOG + Earley [5]	54.3	61.4	39.2	45.4
LSTM	52.8	52.5	52.8	47.6
LSTM + GEP	**70.6**	**72.1**	**70.6**	**70.1**

an input frame at time t. For future segment prediction, it outputs the label of the next segment for an input frame.

8) Bi-LSTM/LSTM + GEP: the proposed generalized Earley parser (GEP) applied to the classifier output for detection and prediction. The predictions for the next segments are made according to Section 5.5. The lengths of unobserved segments are sampled from a log-normal distribution for the future 3s prediction.

Implementation details. We use the same Bi-LSTM as the base classifier for detection task and LSTM as the base classifier for prediction tasks on all three datasets. For both models, we used a 2 layer LSTM backbone with hidden size 256 and added bidirectional propagation for the Bi-LSTM model. For training, we use a Adam optimizer with learning rate 1×10^{-3} and set weight decay as 0.8 for every 20 epochs.

Feature extraction. All methods in the experiment use the same publicly available features from KGS [2]. These features include the human skeleton features and human-object interaction features for each frame. The human skeleton features are location and distance features (relative to the subjects head location) for all upper-skeleton joints of a subject. The human-object features are spatial-temporal, containing the distances between object centroids and skeleton joint locations as well as the temporal changes.

Experiment results. We follow the convention in KGS [2] to train on three subjects and test on a new subject with a 4-fold validation. The results for the three evaluation metrics are summarized in Table 2, Table 3 and Table 4, respectively. Figure 8 shows the confusion matrices for the two prediction tasks.

TABLE 5: Detection results on Watch-n-Patch.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
ST-AOG + Earley [5]	79.3	71.5	73.5	71.9
MLP	55.6	48.7	46.7	46.4
MLP + GEP	78.1	73.0	68.4	69.7
Bi-LSTM	84.0	79.7	82.2	80.3
Bi-LSTM + GEP	**84.8**	**80.7**	**83.4**	**81.5**

TABLE 6: Future 3s prediction results on Watch-n-Patch.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
ST-AOG + Earley [5]	48.9	43.1	39.3	39.3
LSTM	43.9	28.3	26.6	24.9
LSTM + GEP	**58.7**	**50.5**	**49.9**	**49.4**

TABLE 7: Segment prediction results on Watch-n-Patch.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
ST-AOG + Earley [5]	29.4	28.5	18.9	19.9
LSTM	44.6	43.6	44.6	40.4
LSTM + GEP	**49.5**	**50.1**	**49.4**	**45.5**

Our method outperforms the comparative methods on all three tasks. Specifically, the generalized Earley parser on top of a Bi-LSTM performs better than ST-AOG, while ST-AOG outperforms the Bi-LSTM. More discussions are highlighted in Section 6.5.

6.3 Experiment on Watch-n-Patch Dataset

Dataset. Watch-n-Patch [3] is an RGB-D dataset that features forgotten actions. For example, a subject might fetch milk from a fridge, pour milk, and leave. The typical action "putting the milk back into the fridge" is forgotten. The dataset contains 458 videos with a total length of about 230 minutes, in which people forgot actions in 222 videos. Each video in the dataset contains 2-7 actions interacted with different objects. 7 subjects are asked to perform daily activities in 8 offices and 5 kitchens with complex backgrounds. It consists of 21 types of fully annotated actions (10 in the office, 11 in the kitchen) interacted with 23 types of objects.

Feature extraction. We extract the same features as described in [3] for all methods. Similar to the previous experiment, the features are composed of skeleton features and human-object interaction features extracted from RGB-D images. The skeleton features include angles between connected parts, the change of joint positions and angles from previous frames. Each frame is segmented into super-pixels, and foreground masks are detected. We extract features from the image segments with more than 50% in the foreground mask and within a distance to the human hand joints in both 3D points and 2D pixels. Six kernel descriptors [58] are extracted from these image segments: gradient, color, local binary pattern, depth gradient, spin, surface normals, and KPCA/self-similarity.

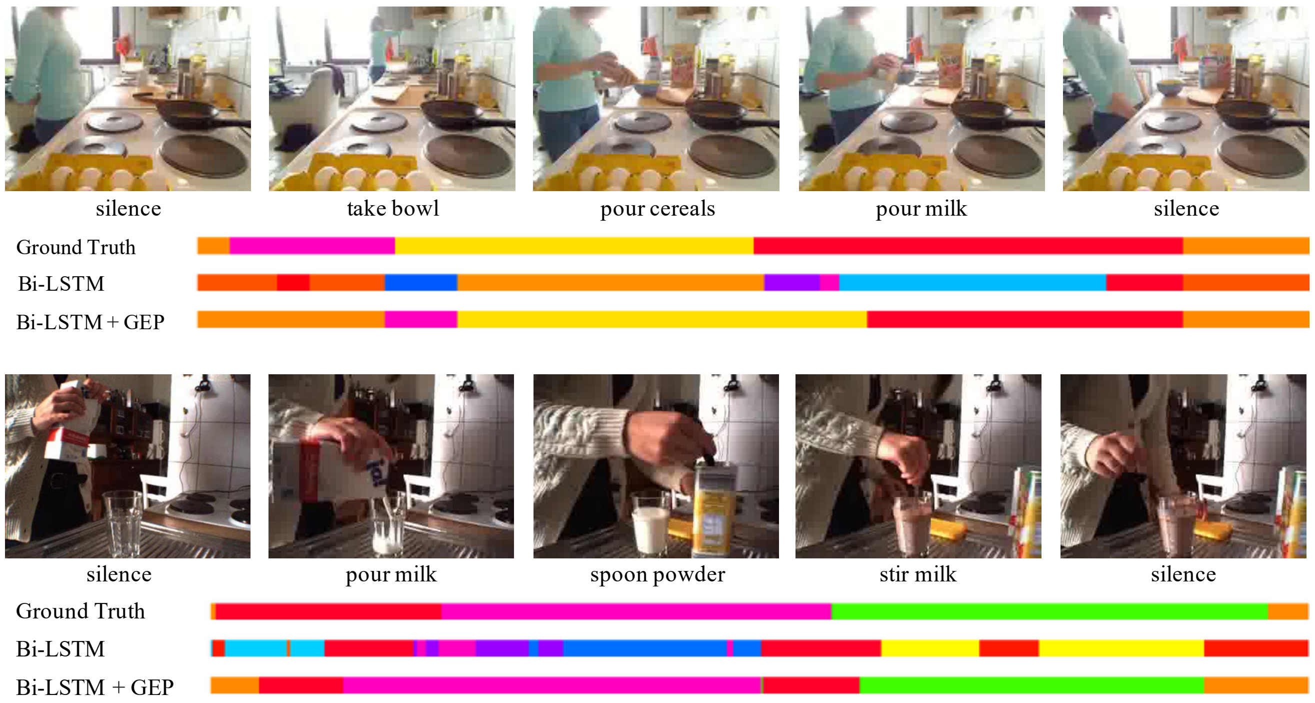


Fig. 9: Qualitative results on the Breakfast dataset. The top row pictures show the typical frames and labels of the groud-truth segments. The bottom rows show the ground-truth segmentation, Bi-LSTM, and Bi-LSTM + GEP results. Best viewed in color.

Experiment results. We use the same evaluation metrics as the previous experiment and compare our method to ST-AOG [5] and Bi-LSTM. For detection, we also use the base classifier Multilayer Perceptron (MLP) and MLP with our generalized Earley parser (GEP), i.e MLP + GEP. We use the train/test split in [3]. The results for the three evaluation metrics are summarized in Table 5, Table 6 and Table 7, respectively. Our method slightly improves the detection results over the Bi-LSTM outputs, and outperforms the other methods on both prediction tasks. In general, the algorithms make better predictions on CAD-120, since Watch-n-Patch features forgotten actions and the behaviors are more unpredictable. Figure 11 shows some qualitative results, and more details are discussed in Section 6.5.

6.4 Experiment on Breakfast Dataset

Dataset. Breakfast [4] is a dataset of daily cooking activities. The dataset includes 52 unique participants, each conducting 10 distinct cooking activities captured in 18 different kitchens. It has ∼77 hours of video footage containing different camera views (3 to 5 depending on the location). For data annotations, 48 different coarse action units are identified with 11,267 samples (segments) in total including ∼3,600 silence samples.

Comparative methods. Besides Bi-LSTM, we compare Bi-LSTM + generalized Earley parser (GEP) with state-of-art methods for activity detection on the Breakfast dataset. The base classifier Multilayer Perceptron (MLP) and MLP + GEP are also tested. The other comparative methods include:

1) HOGHOF+HTK [4]: a grammar-based method. The authors proposed to use the hidden Markov model (HMM) for modeling individual action units in the sequence recognition problem. These action units then form the building blocks to model complex human activities as sentences using an action grammar. A speech recognition engine (the HTK toolkit [59]) is used for recognition on top of the extracted HOGHOF features [9].

2) ED-TCN [60]: an end-to-end method. Encoder-Decoder Temporal Convolutional Network is proposed to tackle the action classification problem. Under the model's setting, predictions at each frame are a function of a fixed-length period of time, which is referred to by the authors as the receptive field.

3) TCFPN [61]: one of the end-to-end state-of-the-art methods. The Temporal Convolutional Feature Pyramid retains the encoder-decoder architecture and adapt the lateral connection mechanism proposed in Feature Pyramid networks to the task of action segmentation.

4) Fisher+HTK [62]: one of the grammar-based state-of-the-art methods. Similar to [4], the action units are modeled by HMM, and high-level activities are modeled by action grammars. The main difference is the feature (Fisher kernels [63]) proposed for action recognition.

Experiment results. To eliminate the factors of feature extraction for fair comparison, we use the pre-computed feature provided by [62] to train the underlying Bi-LSTM classifier. Figure 9 shows the qualitative results of activity detection on the Breakfast dataset. The quantitative results (Table 8) show that a simple Bi-LSTM is far from state-of-the-art methods (an absolute difference of 10.7%). Our full algorithm Bi-LSTM + Genearlized Earley improves the absolute performance by 14.1%, and outperforms the state-of-the-art by 3.6%. This shows that our explicit grammar

TABLE 8: Detection results on Breakfast.

Method	Micro	Macro		
	P/R	Prec.	Recall	F1-score
HOGHOF+HTK [4]	28.8	–	–	–
ED-TCN [60]*	43.3	–	–	–
TCFPN [61]	52.0	–	–	–
Fisher+HTK [62]	56.3	38.1	–	–
MLP	15.4	7.2	7.5	5.9
MLP + GEP	32.5	35.9	15.6	18.5
Bi-LSTM	45.6	29.2	25.4	25.6
Bi-LSTM + GEP	**59.7**	**45.8**	**36.3**	**38.5**

*The results for [60] is obtained from [61].

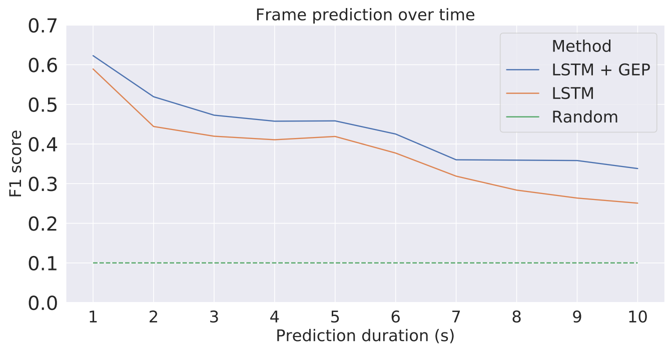
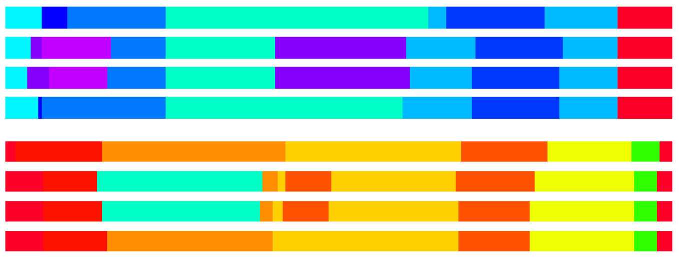
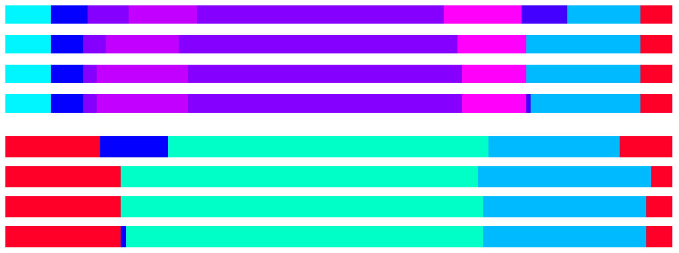


Fig. 10: Frame-wise prediction accuracy vs. prediction duration on the CAD-120 dataset. GEP stands for the generalized Earley parser.



(a) Correction



(b) Insertion

Fig. 11: Qualitative results of segmentation results on Watch-and-Patch. In each group of four segmentations, the rows from the top to the bottom show the results of: 1) ground-truth, 2) ST-AOG + Earley, 3) Bi-LSTM, and 4) Bi-LSTM + generalized Earley parser. The results show (a) corrections and (b) insertions by our algorithm on the initial segment-wise labels given by the classifier (Bi-LSTM).

regularization is effective in correcting the mistakes of the underlying classifier. Although the underlying classifier is simple, it is able to perform well in the activity detection task well.

6.5 Discussion

How does the performance change with respect to the prediction duration? Figure 10 shows how the frame-wise prediction accuracy changes when the prediction duration varies on the CAD-120 dataset. In the CAD-120 dataset, the average duration of a sub-activity is around 3.6 seconds. Hence a prediction duration of 10 seconds is over two to three sub-activities. When the prediction duration increases, the performance of both approaches drop, but we can see that the generalized Earley parser still improves the performance of the underlying classifier.

Can we reduce the running time for the generalized Earley parser? When running the generalized Earley parser for detection, the most time-consuming step is the segmentation by dynamic programming. As discussed in Section 5.4, the complexity for segmentation and labeling is $O(T^2)$ if we run the complete algorithm. One approximate solution is to run the algorithm on a sub-sampled video. This would reduce the running time quadratically. The potential effect of sub-sampling is two-fold: 1) certain information is dropped and could hurt the performance, whereas 2) given the length of the activities are generally long, sub-sampling can serve as a weak duration prior that might improve performance. We study the effect of sub-sampling the video, and Figure 12 shows the performance under different sample rates for both the Bi-LSTM and Bi-LSTM + generalized Earley parser.

As we can see from Figure 12, our proposed model outperforms Bi-LSTM models under all sample rates. The results show a slight increase for both models when the sample rate increase. We argue that this matches with the second hypothesis where the added sub-sampling procedure can be seen as adding basic duration constraints. We also conjecture that the sub-sampled features become more distinguishable which further contributes to the overall performance improvement.

How different are the classifier outputs and the final outputs for detection? Figure 11 shows some qualitative examples of the ground truth segmentations and results given by different methods. The segmentation results show that the refined outputs are overall similar to the classifier outputs since the confidence given by the classifiers are often very high, but some segments are modified to ensure the grammatical correctness.

How does the generalized Earley parser refine the classifier detection outputs? When the classifier outputs violate the grammar, two types of refinements occur: i) correction and deletion of wrong labels as shown in Figure 11a; ii) insertion of new labels as shown in Figure 11b. The inserted segments are usually very short to accommodate both the grammar and the classifier outputs. Most boundaries of the refined results are well aligned with the classifier outputs.

Why do we use two metrics for future prediction? The future 3s prediction is a standard evaluation criterion set up by KGS and ATCRF. However, this criterion does not tell how well the algorithm predicts the next segment

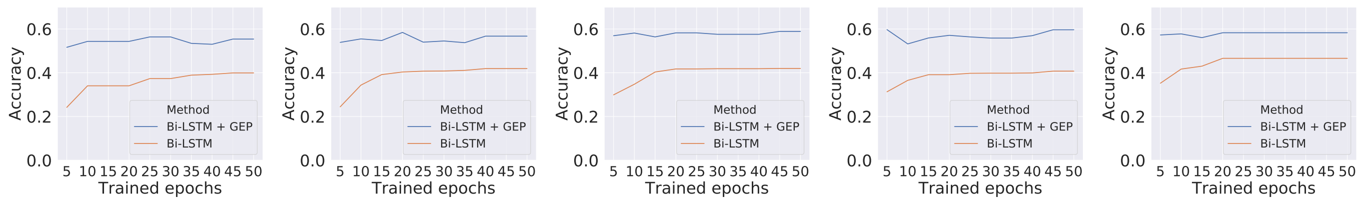


Fig. 12: Detection accuracy vs. trained epochs on the Breakfast dataset. From left to right, the videos are sub-sampled by a sample rate of 1, 2, 5, 10, and 20 frames, respectively. The running time for Bi-LSTM + Generalized Earley on the testing splits of Breakfast (on average 27,000 frames before sub-sampling) are on average 150, 40, 10, 2, 1, 0.5 (min) on a single Titan X GPU desktop with an Intel 8700K CPU.

label. i) At any time frame, part of the future 3s involves the current sub-activity for most of the times. ii) If the predicted length of the current sub-activity is inaccurate, the frame-wise inaccuracy drops proportionally, even when the future segment label prediction is accurate. Therefore we also compare against the future segment label prediction because it is invariant to variations in activity lengths.

How well does the generalized Earley parser perform for activity detection and prediction? In some cases it slightly improves over the classifier outputs for detection (Table 5), but still significantly outperforms the classifier for predictions (Table 6 and Table 7). The modifications on classifier outputs (corrections and insertions in Figure 11) are minor but important to make the sentences grammatically correct, thus high-quality predictions can be made.

How useful is the grammar for activity modeling? From Table 3, Table 4, Table 6 and Table 7 we can see that both ST-AOG and generalized Earley parser outperforms Bi-LSTM for prediction. Prediction algorithms need to give different outputs for similar inputs based on the observation history. Hence the non-Markovian property of grammars is useful for activity modeling, especially for future prediction.

How effective is the generalized Earley parser with different base classifiers? On the CAD-120 dataset, the Watch-n-Patch dataset, and the Breakfast dataset, we compare the detection results of different base classifiers Multilayer Perceptron (MLP), Bidirectional LSTM (Bi-LSTM), and the base classifiers with our generalized Earley parser (GEP), i.e. MLP + GEP and Bi-LSTM + GEP. Table 2, Table 5, and Table 8 show the comparison results on the three datasets, respectively. We have two observations from the results in the three tables. First, our generalized Earley parser improves the detection results of the base classifiers. Second, the generalized Earley parser algorithm helps the weaker classifiers more. Generally, the base classifier MLP is weaker than Bi-LSTM for detection. Combining the generalized Earley parser algorithm, the performance improvement for MLP is much larger than for Bi-LSTM. These two observations are consistent with the motivation of our proposed method and also prove its effectiveness and strength.

How useful is the grammar prior? In this work, we incorporate a non-trivial grammar prior into the generalized Earley parser, which is a novel contribution compared with our previous conference paper [6]. The usefulness and novelty of incorporating the grammar prior into the parser lie in two aspects: methodology and performance.

From the perspective of methodology, incorporating the grammar prior into the parser makes the model theoretically comprehensive and complete. The grammar prior is supposed to provide guidance for the parser when the input sequence is not easily distinguishable, i.e. the input probability sequence is nearly uniform. Meanwhile, balancing the use of prior knowledge and current data likelihood is difficult: a wrong grammar prior could potentially lead to wrong parsing states that are not reversible. To avoid this problem, we add the grammar prior transition probability between action segments as shown in Eq. 8. Action segments with short length will be more influenced by the grammar prior due to the inadequate data evidence. As the segments become longer, the data likelihood will become dominant. Incorporating the grammar prior into the parser provides a possibility in probabilistic formulation for addressing the above challenges, and therefore makes the model theoretically comprehensive and complete.

From the perspective of performance, incorporating the grammar prior into the parser is helpful to improve algorithm performance. To analyze the effects of the grammar prior, we carried out ablation studies from two aspects. In the first experiment, we control the uniformity of the input probability sequence and compare the performance of models with and without the grammar prior. This experiment is designed to show the effect of the prior grammar in the scenario of uniform input probabilities. In the second experiment, we change the length of each action segment to compare the performance. This experiment studies the situations that the data likelihood becomes dominant.

We use the ground truth action label sequences in the CAD-120 dataset to simulate and generate the input probability sequences to the parser. We extract the label sequence from each video and repeat each action label for different numbers of times to get new sequences with different segment lengths. Next, we convert the one-hot vector representing the ground truth action label for each frame into logits which are passed into a softmax function to generate probabilities. The probability sequence of a video is used as the input to the parser. To control the uniformity of the generated probabilities, we divide the logits by a temperature factor parameter before passing them into the softmax function. The action segment length and the temperature factor are used as the control variables for the ablation studies. The higher (larger) the temperature, the more uniform the input probabilities; the longer the segments, the larger the cumulative effect of the likelihood.

With different values of the control variables, we compare the parsing precision of the generalized Earley parser

TABLE 9: Parsing precision of the ablation study of the grammar prior. The pairing results follow the format 'with prior / without prior'. Comparatively better results are highlighted in bold.

Segment Length	Temperature Factor				
	1 (One-Hot)	10^5	7×10^6	10^7	10^{10} (Uniform)
1	87.1%/**89.2%**	63.4%/**66.8%**	62.7%/**65.8%**	54.9%/**58.3%**	**53.2%**/46.7%
2	98.0%/**100.0%**	94.6%/**100.0%**	89.8%/**92.2%**	38.1%/**40.2%**	**30.1%**/28.1%
3	99.5%/**99.8%**	**91.4%**/89.2%	**85.1%**/84.2%	**34.5%**/34.1%	**23.5%**/**23.5%**
5	**98.9%**/98.5%	**91.5%**/91.1%	**85.2%**/84.9%	**31.7%**/31.3%	18.2%/**19.0%**
10	**99.1%**/97.7%	**93.9%**/93.6%	**85.7%**/85.6%	**26.9%**/**26.9%**	**15.6%**/**15.6%**
20	**99.5%**/99.4%	**94.8%**/**94.8%**	**85.7%**/**85.7%**	**25.8%**/**25.8%**	**15.9%**/**15.9%**

with and without the grammar prior, as shown in Table 9. From the results we have two main observations:

1) The grammar prior is helpful to improve the parsing precision, especially when the input probability sequence is going to be uniform (last columns of Table 9). In this case, the input probability provides little information to distinguish each action label and the grammar prior probability aggregates prior information into the parser to help discriminate different possible parses of the input.
2) The grammar prior is also helpful when the action segments are long and the input probability sequence is informative (the last four rows and the first four columns of Table 9). When the segment length is small (the first two rows and four columns of Table 9), the grammar prior affects the parsing process by introducing strong prior obtained from previous observations. Even with the non-uniform data likelihood, the discrepancy between the current data and the grammar prior still led to performance drop. These results are consistent with our previous discussions of the compounding error problem for adding grammar prior. Following Eq. 7 and Eq. 8, data likelihood dominates the parsing probability when the input sequence is long. This encourages the parser to fit the data sequences when they are distinguishable and minimize the side effect caused by an inaccurate grammar prior.

These results and observations demonstrate that incorporating the prior grammar is positive both in methodology and experiment performance.

How robust is the generalized Earley parser? Comparing Table 4 and Table 7 we can see that there is a performance drop when the action sequences are more unpredictable (in the Watch-n-Patch dataset). But it is capable of improving over the noisy classifier inputs and significantly outperforms the other methods. It is also robust in the sense that it can always find the best sentence in a given language that best explains the classifier outputs.

7 Conclusion

We proposed a generalized Earley parser for parsing sequence data according to symbolic grammars. Detections and predictions are made efficiently by the parser given the probabilistic outputs from a general base classifier. Experiments show that the generalized Earley parser improves the performance for a base classifier for both detection and prediction task in general. We are optimistic about and interested in further applications of the generalized Earley parser. In general, we believe this is a step towards the goal of integrating the connectionist and symbolic approaches.

Acknowledgments

The authors would like to thank Professor Ying Nian Wu from UCLA Statistics Department for helpful comments on this work. The work reported herein is supported by DARPA XAI N66001-17-2-4029, ONR MURI N00014-16-1-2007, N66001-17-2-3602, and NSFC 61876149.

References

[1] J. Earley, "An efficient context-free parsing algorithm," *Communications of the ACM*, 1970.
[2] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *International Journal of Robotics Research (IJRR)*, 2013.
[3] C. Wu, J. Zhang, S. Savarese, and A. Saxena, "Watch-n-patch: Unsupervised understanding of actions and relations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
[4] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
[5] S. Qi, S. Huang, P. Wei, and S.-C. Zhu, "Predicting human activities using stochastic grammar," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.
[6] S. Qi, B. Jia, and S.-C. Zhu, "Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction," in *IEEE International Conference on Machine Learning (ICML)*, 2018.
[7] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.
[8] B. Laxton, J. Lim, and D. Kriegman, "Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
[9] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
[10] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *European Conference on Computer Vision (ECCV)*, 2010.
[11] A. Gaidon, Z. Harchaoui, and C. Schmid, "Actom sequence models for efficient action detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
[12] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
[13] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis, "Representing videos using mid-level discriminative patches," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
[14] J. Liu, B. Kuipers, and S. Savarese, "Recognizing human actions by attributes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
[15] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele, "Script data for attribute-based recognition of composite activities," in *European Conference on Computer Vision (ECCV)*, 2012.

[17] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Attribute learning for understanding unstructured social activity," in *European Conference on Computer Vision (ECCV)*, 2012.

[18] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis, "Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[19] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010.

[20] Y. Song, L.-P. Morency, and R. Davis, "Action recognition by hierarchical sequence summarization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[21] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu, "Action recognition with actons," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[22] T. Lan, Y. Zhu, A. Roshan Zamir, and S. Savarese, "Action recognition by hierarchical mid-level action elements," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] S. Holtzen, Y. Zhao, T. Gao, J. B. Tenenbaum, and S.-C. Zhu, "Inferring human intent from video by sampling hierarchical plans," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[24] P. Wei, Y. Zhao, N. Zheng, and S.-C. Zhu, "Modeling 4d human-object interactions for joint event segmentation, recognition, and object localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[25] X. Wang, A. Farhadi, and A. Gupta, "Actions˜ transformations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] M. S. Ibrahim and G. Mori, "Hierarchical relational networks for group activity recognition and retrieval," in *European Conference on Computer Vision (ECCV)*, 2018.

[28] A. Cherian, S. Sra, S. Gould, and R. Hartley, "Non-linear temporal subspace representations for activity recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] Y. Zhang, P. Tokmakov, C. Schmid, and M. Hebert, "A structured model for action detection," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2000.

[31] M. Pei, Y. Jia, and S.-C. Zhu, "Parsing video events with goal inference and intent prediction," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[32] H. Pirsiavash and D. Ramanan, "Parsing videos of actions with segmental grammars," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[33] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[34] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[35] J. Yuen and A. Torralba, "A data-driven approach for event prediction," in *European Conference on Computer Vision (ECCV)*, 2010.

[36] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[37] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision (ECCV)*, 2012.

[38] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation." in *Robotics: Science and Systems (RSS)*, 2012.

[39] Z. Wang, M. P. Deisenroth, H. B. Amor, D. Vogt, B. Schölkopf, and J. Peters, "Probabilistic modeling of human movements for intention inference," *Robotics: Science and Systems (RSS)*, 2012.

[40] M. Pei, Z. Si, B. Yao, and S.-C. Zhu, "Video event parsing and learning with goal and intent prediction," *Computer Vision and Image Understanding (CVIU)*, 2013.

[41] J. Walker, A. Gupta, and M. Hebert, "Patch to the future: Unsupervised visual prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[42] T.-H. Vu, C. Olsson, I. Laptev, A. Oliva, and J. Sivic, "Predicting actions from static scenes," in *European Conference on Computer Vision (ECCV)*, 2014.

[43] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.

[44] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] D. Xie, T. Shu, S. Todorovic, and S.-C. Zhu, "Modeling and inferring human intents and latent functional objects for trajectory prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[46] N. Rhinehart and K. M. Kitani, "First-person activity forecasting with online inverse reinforcement learning," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[47] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani, "Forecasting interactive dynamics of pedestrians with fictitious play," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[48] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[49] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[50] D. Parsing, "A grammar correction algorithm," in *Formal Grammar: 14th International Conference*, 2011.

[51] J. Wagner, "Detecting grammatical errors with treebank-induced, probabilistic parsers," Ph.D. dissertation, Dublin City University, 2012.

[52] J. Wagner and J. Foster, "The effect of correcting grammatical errors on parse probabilities," in *Proceedings of the 11th International Conference on Parsing Technologies*. Association for Computational Linguistics, 2009.

[53] A. Stolcke, "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities," *Computational linguistics*, 1995.

[54] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, 1967.

[55] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised learning of natural languages," *Proceedings of the National Academy of Sciences (PNAS)*, 2005.

[56] K. Tu, M. Pavlovskaia, and S.-C. Zhu, "Unsupervised structure learning of stochastic and-or grammars," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2013.

[57] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *IEEE International Conference on Machine Learning (ICML)*, 2006.

[58] C. Wu, I. Lenz, and A. Saxena, "Hierarchical semantic labeling for task-relevant rgb-d perception," in *Robotics: Science and Systems (RSS)*, 2014.

[59] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey *et al.*, "The htk book," *Cambridge university engineering department*, 2002.

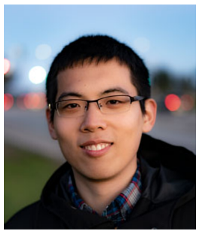
[60] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] L. Ding and C. Xu, "Weakly-supervised action segmentation with iterative soft boundary assignment," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[62] H. Kuehne, J. Gall, and T. Serre, "An end-to-end generative framework for video segmentation and recognition," in *Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[63] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Conference on Neural Information Processing Systems (NeurIPS)*, 1999.

Siyuan Qi received his B.Eng. degree in Computer Engineering from the University of Hong Kong in 2013. He received his M.S. and Ph.D degree in Computer Science from University of California, Los Angeles in 2015 and 2019, respectively. His research interests include pattern recognition, machine learning and computer vision, with a focus on human activity recognition, scene understanding with compositional representations.



Baoxiong Jia received his B.S. degree in Computer Science from Peking University and M.S. degree in Computer Science from University of California, Los Angeles in 2018 and 2019 respectively. He is currently a Ph.D. student in Computer Science at University of California, Los Angeles. His research focus on learning and inference problems that involve spatial-temporal reasoning. His interests include activity prediction and hierarchical task planning.

Siyuan Huang is a Ph.D. student in Statistics at the University of California, Los Angeles. Before that, he received a bachelor's degree in the Department of Automation from Tsinghua University in 2016. His current research focuses on learning 3D representations from images, parsing and reconstruction 3D scene from a single image, and reasoning the human activities in 3D scenes. He has related publications in CVPR, ECCV, ICCV, NeurIPS, and IJCV.

Ping Wei received the BE and PhD degrees from Xi'an Jiaotong University, China. He is currently an associate professor with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University. From 2016 to 2017, he was a postdoctoral researcher at the VCLA Center of UCLA. His research interests include computer vision, machine learning, and cognition modeling. He served as a coorganizer of the Workshop on Vision Meets Cognition: Functionality, Physics, Intents and Causality at CVPR 2017 and 2018, respectively. He is a member of the IEEE.

Song-Chun Zhu received his Ph.D. degree from Harvard University in 1996. He is currently professor of Statistics and Computer Science at UCLA, and director of the Center for Vision, Cognition, Learning and Autonomy (VCLA). He has published over 200 papers in computer vision, statistical modeling, learning, cognition, and visual arts. In recent years, his interest has also extended to cognitive robotics, robot autonomy, situated dialogues, and commonsense reasoning. He received a number of honors, including the Helmholtz Test-of-time award in ICCV 2013, the Aggarwal prize from the Intl Association of Pattern Recognition in 2008, the David Marr Prize in 2003 with Z. Tu et al., twice Marr Prize honorary nominations with Y. Wu et al. in 1999 for texture modeling and 2007 for object modeling respectively. He received the Sloan Fellowship in 2001, a US NSF Career Award in 2001, and an US ONR Young Investigator Award in 2001. He is a Fellow of IEEE since 2011, and served as the general co-chair for CVPR 2012 and CVPR 2019.