**Title**
Human Activity Understanding and Prediction with Stochastic Grammar



**Author**
Jia, Baoxiong

**Publication Date**
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

# Human Activity Understanding and Prediction with Stochastic Grammar

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Computer Science

by

Baoxiong Jia

2019

ABSTRACT OF THE THESIS

Human Activity Understanding and Prediction with Stochastic Grammar

by

Baoxiong Jia
Master of Science in Computer Science
University of California, Los Angeles, 2019
Professor Song-Chun Zhu, Chair

Video understanding is a booming research problem in computer vision. With its innate feature where spatial and temporal information entangles with each other, video understanding has been challenging mainly because of the difficulty for having a unified framework where these two aspects can be modeled jointly. Among the tasks in video understanding, human activity understanding and prediction serve as a good starting point where the spatial-temporal reasoning capability of learning modules can be tested. Most of the current approaches towards solving the human activity understanding and prediction problems use deep neural networks for spatial-temporal reasoning. However, this type of approach lacks the ability to reason beyond the local frames and conduct long-term temporal reasoning. On the other hand, stochastic grammar models are used to model observed sequences on a symbolic level with all history information considered, but they perform poorly on handling noisy input sequences. Given these insights and problems of current approaches, we propose the generalized Earley parser for bridging the gap between sequence inputs and symbolic grammars. By combining the advantages of these two types of methods, we show that the proposed model achieves a better performance on both human activity recognition and future prediction.

The thesis of Baoxiong Jia is approved.

Kai-wei Chang

Ying Nian Wu

Song-Chun Zhu, Committee Chair

University of California, Los Angeles

2019

*To my family and friends who have*
*supported me along the journey*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude towards my advisor Dr. Song-chun Zhu for his continuous support and guidance on my study and research. I would also like to all the colleagues in VCLA, in particular, Siyuan Qi for setting an great example for being a good researcher and also the help he provided in months of discussions on new ideas, Yixin Zhu and Feng Gao for showing how me how research should be conducted and providing general guidance on life beyond research, Siyuan Huang, Yixin Chen and Luyao Yuan for their insightful comments on my research, Hangxin Liu and Mark Edmonds for showing me how ideas should be thoroughly developed and presented, Tao Yuan and Xu Xie for being exemplars on code writing.

Last but not the least, I would like to thank my family members for the support they provided me along this journey. I would not have finished this thesis without their love and encouragement.

This work is mainly based on the paper "Generalized Earley Parser: Bridging Symbolic Grammars and Sequence Data for Future Prediction" published in ICML 2018. The list of co-authors is Siyuan Qi and Song-chun Zhu. Siyuan Qi led the project and Song-chun Zhu provided guidance and support.

# CHAPTER 1

# Introduction

In the past years, we have witnessed a tremendous progress in the capabilities of computer systems where amazing results are achieved on image analysis tasks. With models performing comparably well as human in the scope of image classification, the focus of computer vision field has gradually shifted to harder vision tasks. Among them, video understanding, especially human activity recognition and prediction, have received great attention because of its crucial role for future applications *e.g.* surveillance systems and collaborative robots. The major challenge of these two tasks compared to image classification or pose estimation is the fact that these inference processes requires reasoning not only on the current frame, but also previous frames that provide definitive hints of what activities the subjects are doing.

Consider the scenario shown in Figure 1.1. With the current observation that the subject is walking at this frame, we human can make the prediction that the subject is probably going to get water or use microwave. This simple reasoning process actually requires machines to have the capability of making inference with different level of spatial information combined (*e.g.* pose estimation, object detection, *etc.*). If we take this example further, spatial information alone is not enough for making prediction since we could have similar observations when people are doing tasks completely different from each other. For example, if the person just finished drinking, he is probably heading for the water cooler for more water, but if the person just arrived at the office, it is highly likely that he is pouring the old tea from yesterday. These two actions can not be distinguished from the information contained in the current frame along. So in such cases, reasoning jointly on the spatial-temporal domain is necessary for machines to truly understand what is going on.

With the boom of deep learning research, deep neural networks (DNNs) have been applied

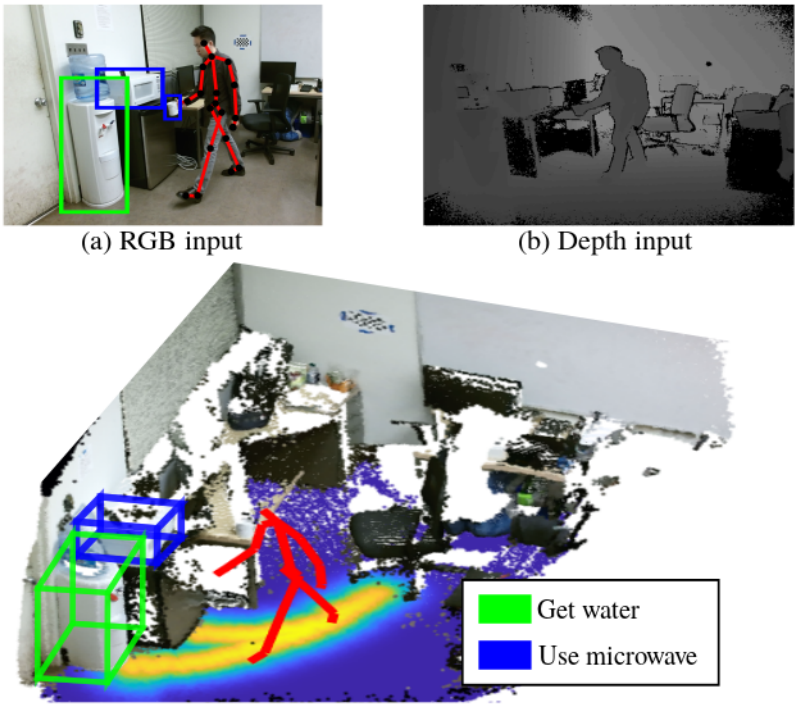



Figure 1.1: An illustrative example for human activity recognition and prediction. (a)(b) Input RGB-D video frames. (c) The prediction of possible activities with the current observation.

to various tasks and achieved good performance. With its dense representations generated along back propagation, it serve as backbone models for most of the representation learning tasks. In video understanding, various design of DNN backbones have been used for learning the spatial-temporal representations [KTS14, SZ14, TBF15, CZ17]. This type of approach generally works well for learning in a local time frame, but the long-term information is often addressed weakly [QHW17, QJZ18]. In addition, the representation obtained from back propagation is usually not meaningful enough for us to design explicit reasoning algorithms on it. With these two major challenges, the video understanding problem is hard to be solved with merely deep neural networks.

To address the importance of retrieving meaningful representations and take the long-term information into consideration at the same time, insights can be drawn from logical reasoning and grammar models. Probabilistic logic models have been used for explicitly

modeling reasoning problems before deep learning took off. Similarly, stochastic grammar models have also been used for modeling and reasoning on the natural language long before deep learning methods took over. These two types of models share the common feature that the internal structure of the modeled data are explicitly learned and used for inference. We argue that this type of structure modeling is crucial for video understanding, especially for long-term temporal inference problems. However, with the domain constrained to symbolic space, finding a good and expressive mapping from the observation space (often continuous and dense) to the symbolic space is generally hard for grammar models.

With the advantages and disadvantages of these two models in mind, it is natural to think about combining DNNs with models that have a explicit representation for the internal structure of the problems. Taking the compositional nature of human activities into consideration, we propose to aggregate the advantages of both neural network models and grammar models, bridge the gap between the two approaches and reason on the spatial-temporal domain jointly through Generalized Earley Parser (GEP). The parser tries to relax the hard symbolic constraint of traditional grammar and operates on the probability space for inference. In our formulation, the parsing or reasoning process takes soft labels (probability distribution over activity categories) for each frame, instead of hard symbolic labels as inputs. As we show further, this greatly improves the capability of grammar models and also enhance the performance of neural network models.

# CHAPTER 2

# Literature Review

The essence of human activity recognition and prediction problem is to find the most meaningful representation for human activities. Such a representation should be able to capture the spatial-temporal relations inside video sequences. In this section, we take a brief overview on the problems in human activity understanding.

## 2.1 Latent Features in Images

One starting point of activity recognition is to extract meaningful features from still images. Among the information lay inside images, human poses are the most informative hints on people's activities. Pose models [WBR07, TH08, CLS15] has been used for extracting low-level features for human activity recognition. In this type of modeling, features like spatial location of joints, angles between different parts, offsets of features between frames are all commonly used for representing human actions. Other features like velocity of the joints, momentum of the joints are also frequently used when sequence of images are given. Next, as activities are highly correlated with objects and scene contexts, some approaches propose to explicitly address the focus on the scene contexts and human-object interactions. Koppula *et al.* [KGS13] modeled object affordance and predict human activity using spatial-temporal conditional random field. Further, Jain *et al.* [JZS16] extended the model with deep neural networks and gained huge improvements on affordance prediction using the structural-RNN architecture. Qi *et al.* [QWJ18] modeled human-object interactions (HOI) explicitly using HOI graph and generate HOI predictions through graph parsing.

## 2.2 Video Representation

Compared to image representation, generating good video representation are difficult in nature for its temporal relations across frames. With the success of deep neural networks in image representation learning, various methods have been proposed for learning good video representation. Karpathy *et al.* [KTS14] first proposed to use fusion schemes for passing information across frames into 2D convolutions. Then, to address the change across frames, Simonyan *et al.* [SZ14] first proposed the two-stream neural network which takes both RGB frame input and optical flow to capture movements. Tran *et al.* [TBF15] built on the two-stream idea and proposed 3D convolution to have convolutional filters that operates on 2D spatial information across a set of frames. Carreira *et al.* [CZ17] further improved the model and proposed I3D architecture which leverage the pretrained results on image classification tasks to help video analysis.

## 2.3 Temporal Structure of Activities

When assigned to predict what a person is doing and will be doing, we human often rely on the internal compositional structure of activities learned from previous experiences. Similarly, many argues that the retrieval of such structures is crucial for the task of activity understanding and proposes to address this temporal structure of activities for better activity recognition. Niebles *et al.* [NCF10] retrieved the temporal structure by training different motion segment classifiers and extracting sequential patterns. Tang *et al.* [TFK12] used the variable-duration hidden Markov model to represent the duration of segments and the transitional probabilities between states. Wei *et al.* [WZZ16] proposed a 4D human-object interaction model for event recognition which integrates different low-level features using a stochastic hierarchical graph similar to And-Or Graph. Qi *et al.* [QHW17, QJZ18] also learned And-Or activity grammar unsupervisedly from videos for activity understanding.

## 2.4 Structured Modeling and Inference

One of the biggest challenges for predicting human activities is that we need to reason on the latent hierarchical structure of activities for future forecasting. As we discussed earlier, one activity might not immediately affect the upcoming activity, but activities that happen in relatively far future. Thus, the problem comes to reasoning on the structured representation learned and making inference.

Probabilistic graphical models are commonly used for sequential prediction problems. The general idea is to find the most probable next state based on the probabilistic graph built from training videos. Laxton *et al.* [LLK07] used a dynamic Bayesian network to represent the underlying temporal structure between activities and predicts the most probable next state. Wu *et al.* [WZS15] used a modified LDA topic model for activity prediction and model the latent topic of activities.Inspired by how human finish this task using grammar and language, grammar models are proposed to improve the probabilistic graphical models [PJZ11, HZG16, QHW17]. Pei *et al.* [PJZ11] used an unsupervised learning method to learn the temporal grammar for video parsing . Holtzen *et al.* [HZG16] used an hierarchical task model for human intent prediction. For these grammar models, the inference step is essentially finding the best parse that satisfies the learned grammar and predictions are usually made based on the top-down parsing process.

As DNNs are already widely applied in practice, there are methods that attempt to model and solve the structured prediction problem with DNNs. Du *et al.* [DWW15] proposed the Hierarchical RNN for contextual information modeling and inference. Others proposed to add human knowledge in the learning process and have multi-step learning for activity understanding and prediction. Walker *et al.* [WMG17] generated frames by first predicting the human pose. Villegas *et al.* [VYZ17] also attempted to learn the latent structure using LSTMs for the task of pose prediction. The advantage of deep neural networks provide tasks specific representations that perform well on the given dataset but generalize poorly to other scenarios.

# CHAPTER 3

# Background

## 3.1 Stochastic Grammar

With its origin tracing back to the first millenium BCE, the modern formalization of grammar attributed to Chomsky [Cho02]. By definition, a stochastic grammar is defined into a 4-tuple $\mathcal{G} = (V_N, V_T, R, \Gamma)$, where $V_N$ is a finite set of non-terminal nodes, $V_T$ is a finite set of terminal symbols, $\Gamma \in V_N$ is a start symbol, and $R$ is a finite set of production rules. We constrain the production rules to be

$$R = \{\gamma : \alpha \to \beta\} \tag{3.1}$$

Here we require that $\alpha, \beta \in (V_N \cup V_T)^+$ are strings of grammar nodes and $\alpha$ includes at least a non-terminal symbol. Depending on the type of production rules, Chomsky classified language into four types.

For the clarity of illustration, we use the notation where $\alpha, \beta \in (V_N \cup V_T)^*$ denote strings of grammar nodes, $\eta \in (V_N \cup V_T)^+$ denotes an non-empty string of grammar nodes, lower-case letter $a \in V_T$ denotes a terminal node, $A, B \in V_N$ denote non-terminal nodes and $\omega$ denotes a terminal string. The Chomsky hierarchy of language is defined as follows:

**Type-0** grammars are unrestricted grammars with production rules of form

$$\alpha A \beta \to \beta$$

**Type-1** grammars are context-sensitive grammars with production rules of form

$$\alpha A \beta \to \alpha \eta \beta$$

**Type-2** grammars are context-free grammars with production rules of form

$$A \to \alpha$$

**Type-3** grammars are regular grammars with production rules of the form

$$A \to a \quad \text{and} \quad A \to aB$$

Walking from the bottom of the language hierarchy to the top, the constraints on grammar production rules are gradually relaxed. The expressive power of the grammar gradually decreases going from top to the bottom at the same time. Such expressive power is defined as the different sets of *language* that can be generated from the grammar. Here we borrow the notations from [ZM07]. A *language* of a grammar $\mathcal{G}$ is defined as the set of all possible strings of terminals $\omega$ that can be derived from the grammar. We denote grammar $\mathcal{G}$'s language as $\mathbf{L}(\mathcal{G})$:

$$\mathbf{L}(\mathcal{G}) = \{\omega : S \overset{R^*}{\Longrightarrow} \omega, \omega \in V_T^*\} \tag{3.2}$$

where $R^*$ denotes a sequence of production rules $\gamma_1, \gamma_2, ..., \gamma_{n(\omega)}$ that corresponds to the derivation of $\omega$ from $S$. If the grammar is of type 1, 2, or 3, a parse tree $\mathbf{pt}$ can be defined for $\omega$ where

$$\mathbf{pt}(\omega) = (\gamma_1, \gamma_2, ..., \gamma_{n(\omega)}) \tag{3.3}$$

Here we state the fact that type 0 and type 1 grammars are non-deterministic, meaning that no assumptions can be made based on simply the right hand side of the production. Although some certain classes of type 1 grammar are easier to parse since type 2 grammar is contained type 1, parsing type 1 grammar is difficult in general. Therefore, we will mainly focus on type 2 grammar (context-free grammar, CFG) in the following discussion.

As formal grammars only generate language deterministicly, to connect with the real-world scenario where the occurrences of strings comes with probabilities, we must augment the formal grammar with probability measures $\mathcal{P}$. A stochastic context free grammar

(SCFG) is then defined with a fifth component, $\mathcal{G}_p = (V_N, V_T, R, \Gamma, \mathcal{P}(R))$. For each production rule, starting from $A$ we define the probability of different branches as following

$$A \to \alpha_1 B_1 \beta_1 \mid \alpha_2 B_2 \beta_2 \mid \cdots \mid \alpha_{n(A)} B_{n(A)} \beta_{n(A)}$$

$$p(A \to \alpha_i B_i \beta_i) = p_i$$

$$\sum_{i=1}^{n(A)} p(A \to \alpha_i B_i \beta_i) = \sum_{i=1}^{n(A)} = 1$$

With probability measure defined on the production rules, we can define the language of a SCFG $\mathcal{G}_p$ as

$$\mathbf{L}(\mathcal{G}_p) = \{(\omega, p(\omega)) : S \overset{R^*}{\Longrightarrow} \omega, \omega \in V_T^*\}$$

where $p(\omega)$ is the proability that $\omega$ is generated by grammar $\mathcal{G}_p$. Assume $R^* = \gamma_1 \gamma_2 \cdots \gamma_{n(w)}$ where $\gamma_1$, we can first define the parse tree probability $p(\mathbf{pt}_i(\omega))$ as

$$p(\mathbf{pt}_i(\omega)) = \prod_{j=1}^{n(\omega)} p(\gamma_j)$$

and the sentence probability $p(\omega)$ is thus defined as

$$p(\omega) = \sum_{i}^{m(\omega)} p(\mathbf{pt}_i(\omega))$$

Here $m(\omega)$ is the number of parse trees for string $\omega$. The sentence probability $p(\omega)$ is also referred as parsing probability for sentence $\omega$.

## 3.2 Earley Parser

In this section, we review the original Earley parser [Ear70], which is an algorithm for parsing sentences of a given context-free language. The basic concepts are bases for the generalized Earley parser introduced in chapter 4. Following the previous annotations, we use $\alpha$, $\beta$, and $\gamma$ to represent string of terminals/nonterminals (including the empty string $\epsilon$), $A$ and $B$ to represent single nonterminals, and $a$ to represent a terminal symbol. We adopt Earley's dot notation: for production of form $A \to \alpha\beta$, the notation $A \to \alpha \cdot \beta$ means $\alpha$ has been parsed and $\beta$ is expected.

Input position $n$ is defined as the position after accepting the $n$th 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. A simple example is demonstrated in Figure 3.1.

As we can see from the process, the Earley parser operates on a symbolic space where uncertainty at the input level is considered. Although it is widely used in natural language processing domain, it still struggles at solving some problems where sequence data is provided.

Sample grammar: $\Gamma \to R$; $R \to R + R$; $R \to$ "0"|"1"

Input string: $0 + 1$

State: | state # | rule | origin | comment |

$S(0)$

| | | | |
|---|---|---|---|
| (1) | $\Gamma \to \cdot R$ | 0 | start rule |
| (2) | $R \to \cdot R + R$ | 0 | predict: (1) |
| (3) | $R \to \cdot 0$ | 0 | predict: (1) |
| (4) | $R \to \cdot 1$ | 0 | predict: (1) |

$S(1)$

| | | | |
|---|---|---|---|
| (1) | $R \to 0\cdot$ | 0 | scan: $S(0)(3)$ |
| (2) | $R \to R \cdot + R$ | 0 | complete: (1) and $S(0)(2)$ |
| (3) | $\Gamma \to R\cdot$ | 0 | complete: (2) and $S(0)(1)$ |

$S(2)$

| | | | |
|---|---|---|---|
| (1) | $R \to R + \cdot R$ | 0 | scan: $S(1)(2)$ |
| (2) | $R \to \cdot R + R$ | 2 | predict: (1) |
| (3) | $R \to \cdot 0$ | 2 | predict: (1) |
| (4) | $R \to \cdot 1$ | 2 | predict: (1) |

$S(3)$

| | | | |
|---|---|---|---|
| (1) | $R \to 1\cdot$ | 2 | scan: $S(2)(4)$ |
| (2) | $R \to R + R\cdot$ | 0 | complete: (1) and $S(2)(1)$ |
| (3) | $R \to R \cdot + R$ | 0 | complete: (1) and $S(2)(2)$ |
| (4) | $\Gamma \to R\cdot$ | 0 | complete: (2) and $S(0)(1)$ |

Figure 3.1: An example of the original Earley parser.

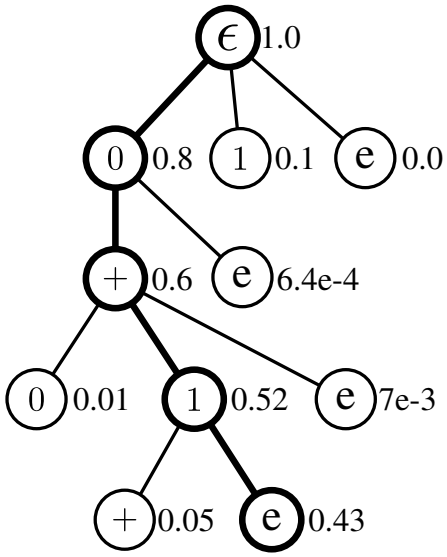
# CHAPTER 4

# Generalized Earley Parser

## 4.1 Overview

To leverage the capability of neural network for handling the uncertainty at input level and combine it into the grammar model, we introduce the generalized Earley parser algorithm [QWJ18]. Instead of taking symbolic sentences as input, we aim to design an algorithm that can parse raw sequence data $\boldsymbol{x}$ of length $T$ (*e.g.*, videos or audios) into a sentence $l$ 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. 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.*, from softmax activations), 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 stochastic context-free grammar $G$ .

The core idea is to use the original Earley parser to help construct a prefix tree according to the grammar as illustrated in Figure 4.1. A prefix tree is composed of terminal symbols and terminations that represent ends of sentences. The root node of the tree is the "empty" symbol. The path from the root to any node in the tree represents a partial sentence (prefix). For each prefix, we can compute the probability that the best label sentence starts with this prefix. This probability is used as a heuristic to search for the best label sentence in the prefix tree: the prefix 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.




Sample grammar:

$\gamma \rightarrow R$

$R \rightarrow R + R$

$R \rightarrow$ "0" | "1"

Input (classifier output):

| 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 |

Figure 4.1: Prefix search according to grammar. A classifier is applied to a 5-frame signal and outputs a probability matrix (bottom right) as the input to our algorithm. The proposed algorithm expands a grammar prefix tree (left), where "e" represents termination. It finally outputs the best label "$0 + 1$" with probability 0.43. The probabilities of children nodes do not sum to 1 since the grammatically incorrect nodes are eliminated from the search.

This heuristic search is realized by generalizing the Earley parser. Specifically, the scan operation in the Earley parser essentially expands a new node in the grammar prefix tree. For each prefix $l$, we can compute $p(l|x_{0:T})$ and $p(l_{\cdots}|x_{0:T})$ based on $\boldsymbol{y}$, where $p(l|x_{0:T})$ is the probability of $l$ being the best label, and $p(l_{\cdots}|x_{0:T})$ is the probability of $l$ being the prefix of the best label of $x_{0:T}$. The formulations for $p(l|x_{0:T})$ and $p(l_{\cdots}|x_{0:T})$ are derived in section 4.2.

We now describe the details for the algorithm. Each scan operation will create a new 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 leaf 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 $p(l_{\dots})$, where $l$ is the parsed string of the state being scanned.

Each state is a tuple $(A \to \alpha \cdot \beta, i, j, l, p(l_{\dots}))$ augmented from the original Earley parser by adding $j, l, p(l_{\dots})$. 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 $(A \to \alpha \cdot B\beta, i, j, l, p(l_{\dots}))$, add $(B \to \cdot\gamma, m, n, l, p(l_{\dots}))$ 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 $(A \to \alpha \cdot a\beta, i, j, l, p(l_{\dots}))$, append the new terminal $a$ to $l$ and compute the probability $p((l + a)_{\dots})$. Create a new set $S(m + 1, n')$ where $n'$ is the current size of $S(m + 1)$. Add $(A \to \alpha a \cdot \beta, i, j, l + a, p((l + a)_{\dots}))$ to $S(m + 1, n')$. Push $S(m + 1, n')$ into $q$ with priority $p((l + a)_{\dots})$.

- Completion: for every state in $S(m, n)$ of the form $(A \to \gamma \cdot, i, j, l, p(l_{\dots}))$, find states in $S(i, j)$ of the form $(B \to \alpha \cdot A\beta, i', j', l', p(l'_{\dots}))$ and add $(B \to \alpha A \cdot \beta, i', j', l, p(l_{\dots}))$ to $S(m, n)$.

This parsing process is efficient since we do not need to search through the entire tree. As shown in Figure 4.1 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 prefix probability is computed correctly, it is guaranteed to return the best solution. A step-by-step generalized Earley parser example is shown in Figure 4.2.

**Algorithm 1:** Generalized Earley Parser

**Input :** Grammar $G$, probability matrix $y$

**Output:** Best label string $l^*$

$S(0,0) = \{(\Gamma \rightarrow \cdot R, 0, 0, \epsilon, 1.0)\}$

$q = priorityQueue()$

$q.push(1.0, (0, 0, \epsilon, S(0,0)))$

**while** $(m, n, l^-, currentSet) = q.pop()$ **do**

  **for** $s = (r, i, j, l, p(l_{...})) \in currentSet$ **do**

    **if** $p(l) > p(l^*)$: $l^* = l$ **then** $l^* = l$

    **if** $r$ ***is*** $(A \rightarrow \alpha \cdot B\beta)$ **then** `// predict`

      **for** *each* $(B \rightarrow \Gamma)$ ***in*** $G$ **do**

        $r' = (B \rightarrow \cdot\Gamma)$

        $s' = (r', m, n, l, p(l_{...}))$

        $S(m,n).add(s')$

      **end**

    **end**

    **else if** $r$ ***is*** $(A \rightarrow \alpha \cdot a\beta)$ **then** `// scan`

      $r' = (A \rightarrow \alpha a \cdot \beta)$

      $m' = m + 1, n' = |S(m+1)|$

      $s' = (r', i, j, l + a, p((l+a)_{...}))$

      $S(m',n').add(s')$

      $q.push(p((l+a)_{...}), (m', n', S(m',n')))$

    **end**

    **else if** $r$ ***is*** $(B \rightarrow \Gamma \cdot)$ **then** `// complete`

      **for** *each* $((A \rightarrow \alpha \cdot B\beta), i', j')$ ***in*** $S(i,j)$ **do**

        $r' = (A \rightarrow \alpha B \cdot \beta)$

        $s' = (r', i', j', l, p(l_{...}))$

        $S(m,n).add(s')$

      **end**

    **end**

    **if** $p(l^-) > p(l)$, $\forall$ *un-expanded* $l$ **then return** $l^*$

  **end**

**end**

**return** $l^*$

## 4.2 Parsing Probability Formulation

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:

$$p(l|x_0) = \begin{cases} y_0^k & l \text{ contains only } k \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

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:

$$p(l|x_{0:t}) = y_t^k(p(l|x_{0:t-1}) + p(l^-|x_{0:t-1})) \tag{4.2}$$

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 consideration.

Then we compute the prefix probability $p(l_{\cdots}|x_{0:T})$ 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:

$$p(l_{\cdots}|x_{0:T}) = p(l|x_0) + \sum_{t=1}^{T} y_t^k p(l^-|x_{0:t-1}) \tag{4.3}$$

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. 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 $O(T|G|)$.

## 4.3 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:

$$p(l|y_{0:e}) = \max_{b<e} p(l^-|y_{0:b}) \prod_{t=b}^{e} y_t^k \tag{4.4}$$

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 $O(T^2)$.

## 4.4 Future Label Prediction

Given the parsing result $l$, we make grammar-based predictions for the next label $z$ to be observed. The predictions are naturally obtained by the predict operation in the generalized Earley parser.

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

$$\Sigma = \{z : \exists s \in S(m, n), s = (X \to \alpha \cdot z\beta, i, j, l, p(l_{...}))\} \tag{4.5}$$

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:

$$z^* = \arg\max_{z \in \Sigma} p(z, l|G) \tag{4.6}$$

For a grammatically complete sentence $u$, the parsing likelihood is simply the Viterbi likelihood [Vit67] given by the probabilistic context-free grammar. For an incomplete sentence $l$ of length $|l|$, the parsing likelihood is given by the sum of all the grammatically possible sentences:

$$p(l|G) = \sum_{u_{1:|l|}=l} p(u|G) \tag{4.7}$$

where $u_{1:|l|}$ denotes the first $|l|$ words of a complete sentence $u$, and $p(u|G)$ is the Viterbi likelihood of $u$.

## 4.5 Maximum Likelihood Estimation for Prediction

We are interested in finding the best grammar and classifier that give us the most accurate 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 $\boldsymbol{y}$ for all the frames in $\boldsymbol{x}$. By merging consecutive labels in $\boldsymbol{y}$ that are the same, we can obtain partial label sentences $l$ and predicted labels $z$. Hence we have $D = \{(\boldsymbol{x}, \boldsymbol{y}, l, z)\}$. The best grammar $G^*$ and the best classifier $f^*$ together minimizes the prediction loss:

$$G^*, f^* = \arg\min_{G,f} \mathcal{L}_{pred}(G, f) \tag{4.8}$$

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

$$\begin{aligned} \mathcal{L}_{pred}(G, f) &= -\sum_{(\boldsymbol{x},z)\in D} \log(p(z|\boldsymbol{x})) \\ &= -\sum_{(\boldsymbol{x},l,z)\in D} (\underbrace{\log(p(z|l, G))}_{\text{grammar}} + \underbrace{\log(p(l|\boldsymbol{x}))}_{\text{classifier}}) \end{aligned} \tag{4.9}$$

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:

$$\prod_{(z,l)\in D} p(z|l,G) = \prod_{(z,l)\in D} \frac{p(z,l|G)}{p(l|G)} = \prod_{u\in D} p(u|G) \tag{4.10}$$

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 to 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.*, [SHR05] and [TPZ13].

The best classifier minimizes the second term of Equation 4.9:

$$\begin{aligned} f^* &= \arg\min_f - \sum_{(\boldsymbol{x},l,z)\in D} \log(p(l|\boldsymbol{x}) \\ &\approx \arg\min_f - \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \sum_k y_k \log(\hat{y_k}) \end{aligned} \tag{4.11}$$

where $p(l|\boldsymbol{x})$ can be maximized by the CTC loss [GFG06]. 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.

Sample grammar: $\gamma \to R$; $R \to R + R$; $R \to 0|1$

Input (classifier output)

| 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 |

Cached probability

| Frame | $\epsilon$ | 0 | 1 | 0 + | 0 + 0 | 0 + 1 | 0 + 1 + |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 8e-1 | 0.1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 6.4e-1 | 1e-2 | 8e-2 | 0 | 0 | 0 |
| 2 | 0 | 6.4e-2 | 1e-3 | 0.58 | 8e-3 | 8e-3 | 0 |
| 3 | 0 | 6.4e-3 | 8e-4 | 6.4e-2 | 5.8e-2 | 0.47 | 8e-4 |
| 4 | 0 | 6.4e-4 | 6.4e-5 | 7e-4 | 1.2e-2 | 0.42 | 4.7e-2 |
| prefix | 1 | 8e-1 | 0.1 | 0.60 | 7.2e-2 | 0.52 | 4.8e-3 |

State: | state # | rule | origin | prefx | comment |

$S(0,0) : l = \epsilon, p(l) = 0.0, p(l\_) = 1.0$

| | | | | |
|---|---|---|---|---|
| (1) | $\gamma \to \cdot R$ | 0, 0 | $\epsilon$ | start rule |
| (2) | $R \to \cdot R + R$ | 0, 0 | $\epsilon$ | predict: (1) |
| (3) | $R \to \cdot 0$ | 0, 0 | $\epsilon$ | predict: (1) |
| (4) | $R \to \cdot 1$ | 0, 0 | $\epsilon$ | predict: (1) |

$S(1,0) : l =$ "0", $p(l) = 6.4e-4, p(l\_) = 0.8$

| | | | | |
|---|---|---|---|---|
| (1) | $R \to 0\cdot$ | 0, 0 | "0" | scan: $S(0,0)(3)$ |
| (2) | $R \to R \cdot + R$ | 0, 0 | "0" | complete: (1) and $S(0,0)(2)$ |
| (3) | $\gamma \to R\cdot$ | 0, 0 | "0" | complete: (2) and $S(0,0)(1)$ |

$S(1,1) : l =$ "1", $p(l) = 6.4e-4, p(l\_) = 0.1$

| | | | | |
|---|---|---|---|---|
| (1) | $R \to 0\cdot$ | 0, 0 | "1" | scan: $S(0,0)(4)$ |

$S(2,0) : l =$ "0 + ", $p(l) = 7.0e-3, p(l\_) = 0.599$

| | | | | |
|---|---|---|---|---|
| (1) | $R \to R + \cdot R$ | 0, 0 | "0 + " | scan: $S(1,0)(2)$ |
| (2) | $R \to \cdot R + R$ | 2, 0 | "0 + " | predict: (1) |
| (3) | $R \to \cdot 0$ | 2, 0 | "0 + " | predict: (1) |
| (4) | $R \to \cdot 1$ | 2, 0 | "0 + " | predict: (1) |

$S(3,0) : l =$ "0 + 0", $p(l) = 1.2e-2, p(l\_) = 7.2e-2$

| | | | | |
|---|---|---|---|---|
| (1) | $R \to 0\cdot$ | 2, 0 | "0 + 0" | scan: $S(2,0)(3)$ |

$S(3,1) : l =$ "0 + 1", **p(l)=0.43**, $p(l\_) = 0.52$

| | | | | |
|---|---|---|---|---|
| (1) | $R \to 1\cdot$ | 2, 0 | "0 + 1" | scan: $S(2,0)(4)$ |
| (2) | $R \to R + R\cdot$ | 0, 0 | "0 + 1" | complete: (1) and $S(2,0)(1)$ |
| (3) | $R \to R \cdot + R$ | 2, 0 | "0 + 1" | complete: (1) and $S(2,0)(2)$ |
| (4) | $\gamma \to R\cdot$ | 0, 0 | "0 + 1" | complete: (2) and $S(0,0)(1)$ |

$S(4,0) : l =$ "0 + 1 + ", $p(l) = 4.7e-2, p(l\_) = 4.8e-2$

| | | | | |
|---|---|---|---|---|
| (1) | $R \to 0\cdot$ | 2, 0 | "0 + 0" | scan: $S(3,1)(3)$ |

Final output: $l^*$ = "0 + 1" with probability 0.43

Figure 4.2: An example of the generalized Earley parser. This example corresponds to Figure 3 in the original paper.

# CHAPTER 5

# Experiments

## 5.1 Experiment Setup

We evaluate our method on the task of human activity recognition and prediction on two datasets, CAD-120 [KGS13] and Watch-n-Patch [WZS15].

The CAD-120 dataset 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. For this dataset, we use the precomputed features from KGS [KGS13] provided with the dataset and test the performance of our algorithm.

Watch-n-Patch is an RGB-D dataset that features action recognition and forgotten actions. In some of the videos, a action is forgotten compared to the standard action sequence. For example, a subject might fetch milk from a fridge, pour milk, and leave where the typical action "putting the milk back into the fridge" is forgotten. The dataset contains 458 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 interacted with 23 types of objects. We extract the same features described in [WZS15] for all methods. These features are composed of human-object interaction features extracted from RGB-D images and skeleton features. Some of the visualizations for the features extracted are shown in Figure 5.2.

In both experiments, we used a modified version of the ADIOS (automatic distillation of structure) [SHR05] grammar induction algorithm to learn the event grammar. The algorithm learns the production rules by generating significant patterns and equivalent classes. 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. The algorithm then finds the equivalent classes that are interchangeable. And therefore find the possible grammar production rules based on the significant patterns found. An example grammar generated by ADIOS is shown in Figure 5.1

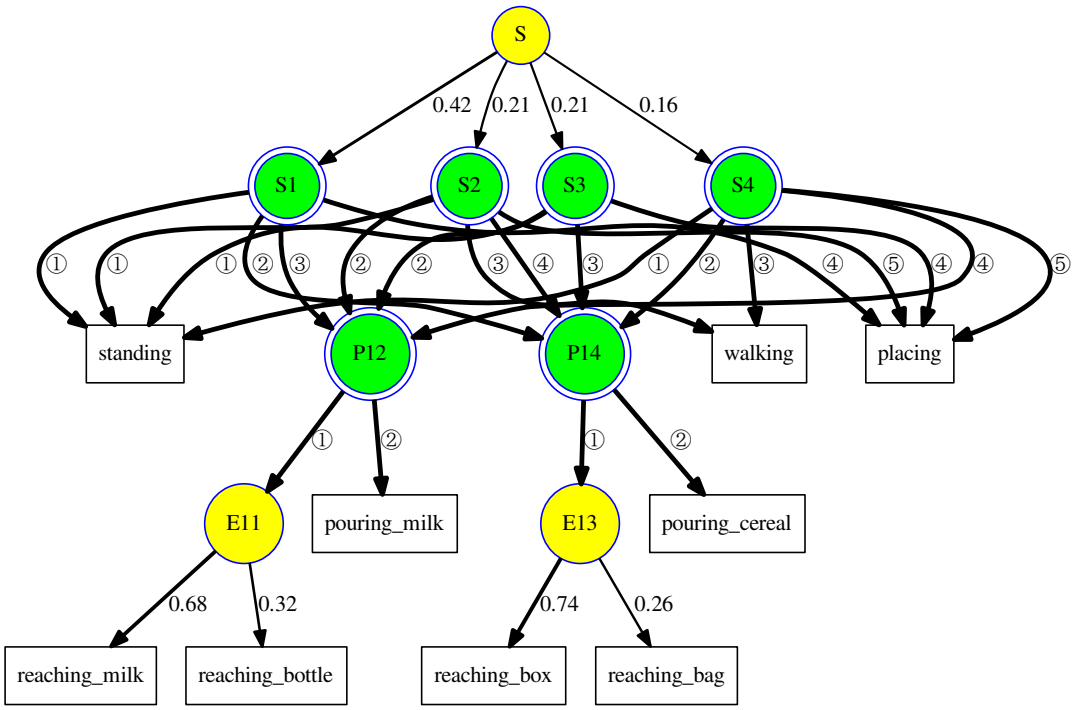



Figure 5.1: An example grammar learned by ADIOS for activity "making cereal".

## 5.2 Results

For the CAD-120 dataset, We follow the convention in KGS [KGS13] 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 5.1, Table 5.2 and Table 5.3, respectively. 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.

Table 5.1: Detection results on CAD-120.

| **Method** | Micro | Macro | | |
|---|---|---|---|---|
| | P/R | Prec. | Recall | F1-score |
| KGS [KGS13] | 68.2 | 71.1 | 62.2 | 66.4 |
| ATCRF [KS16] | 70.3 | 74.8 | 66.2 | 70.2 |
| Bi-LSTM | 76.2 | 78.5 | 74.5 | 74.9 |
| ST-AOG + Earley [QHW17] | 76.5 | 77.0 | 75.2 | 76.1 |
| Bi-LSTM + Generalized Earley | **79.4** | **87.4** | **77.0** | **79.7** |

Table 5.2: Future 3s prediction results on CAD-120.

| **Method** | Micro | Macro | | |
|---|---|---|---|---|
| | P/R | Prec. | Recall | F1-score |
| KGS [KGS13] | 28.6 | – | – | 11.1 |
| ATCRF [KS16] | 49.6 | – | – | 40.6 |
| Bi-LSTM | 54.2 | 61.6 | 39.9 | 34.1 |
| ST-AOG + Earley [QHW17] | 55.2 | 56.5 | 56.6 | 56.6 |
| Bi-LSTM + Generalized Earley | **61.5** | **63.7** | **58.7** | **59.9** |

Table 5.3: Segment prediction results on CAD-120.

| **Method** | Micro | Macro | | |
|---|---|---|---|---|
| | P/R | Prec. | Recall | F1-score |
| Bi-LSTM | 31.4 | 10.0 | 12.7 | 10.1 |
| ST-AOG + Earley [QHW17] | 54.3 | 61.4 | 39.2 | 45.4 |
| Bi-LSTM + Generalized Earley | **72.2** | **70.3** | **70.5** | **67.6** |

For the Watch-n-Patch dataset, we use the same evaluation metrics as the previous experiment and compare our method to ST-AOG [QHW17] and Bi-LSTM. We use the train/test split in [WZS15]. The results for the three evaluation metrics are summarized in Table 5.4, Table 5.5 and Table 5.6, 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.

Table 5.4: Detection results on Watch-n-Patch.

| **Method** | Micro | Macro | | |
|---|---|---|---|---|
| | P/R | Prec. | Recall | F1-score |
| ST-AOG + Earley [QHW17] | 79.3 | 71.5 | 73.5 | 71.9 |
| Bi-LSTM | 84.0 | 79.7 | 82.2 | 80.3 |
| Bi-LSTM + Generalized Earley | **84.8** | **80.7** | **83.4** | **81.5** |

Table 5.5: Future 3s prediction results on Watch-n-Patch.

| **Method** | Micro | Macro | | |
|---|---|---|---|---|
| | P/R | Prec. | Recall | F1-score |
| Bi-LSTM | 42.1 | **66.6** | **62.6** | **61.8** |
| ST-AOG + Earley [QHW17] | 48.9 | 43.1 | 39.3 | 39.3 |
| Bi-LSTM + Generalized Earley | **49.0** | 57.0 | 56.5 | 55.3 |

Table 5.6: Segment prediction results on Watch-n-Patch.

| **Method** | Micro | Macro | | |
|---|---|---|---|---|
| | P/R | Prec. | Recall | F1-score |
| Bi-LSTM | 21.7 | 11.8 | 23.3 | 14.0 |
| ST-AOG + Earley [QHW17] | 29.4 | 28.5 | 18.9 | 19.9 |
| Bi-LSTM + Generalized Earley | **35.6** | **59.2** | **59.3** | **53.5** |

Figure 5.2: Feature extraction for Watch-n-Patch dataset. The red lines are the poses of subjects and the green parts are interactive objects at current frames.

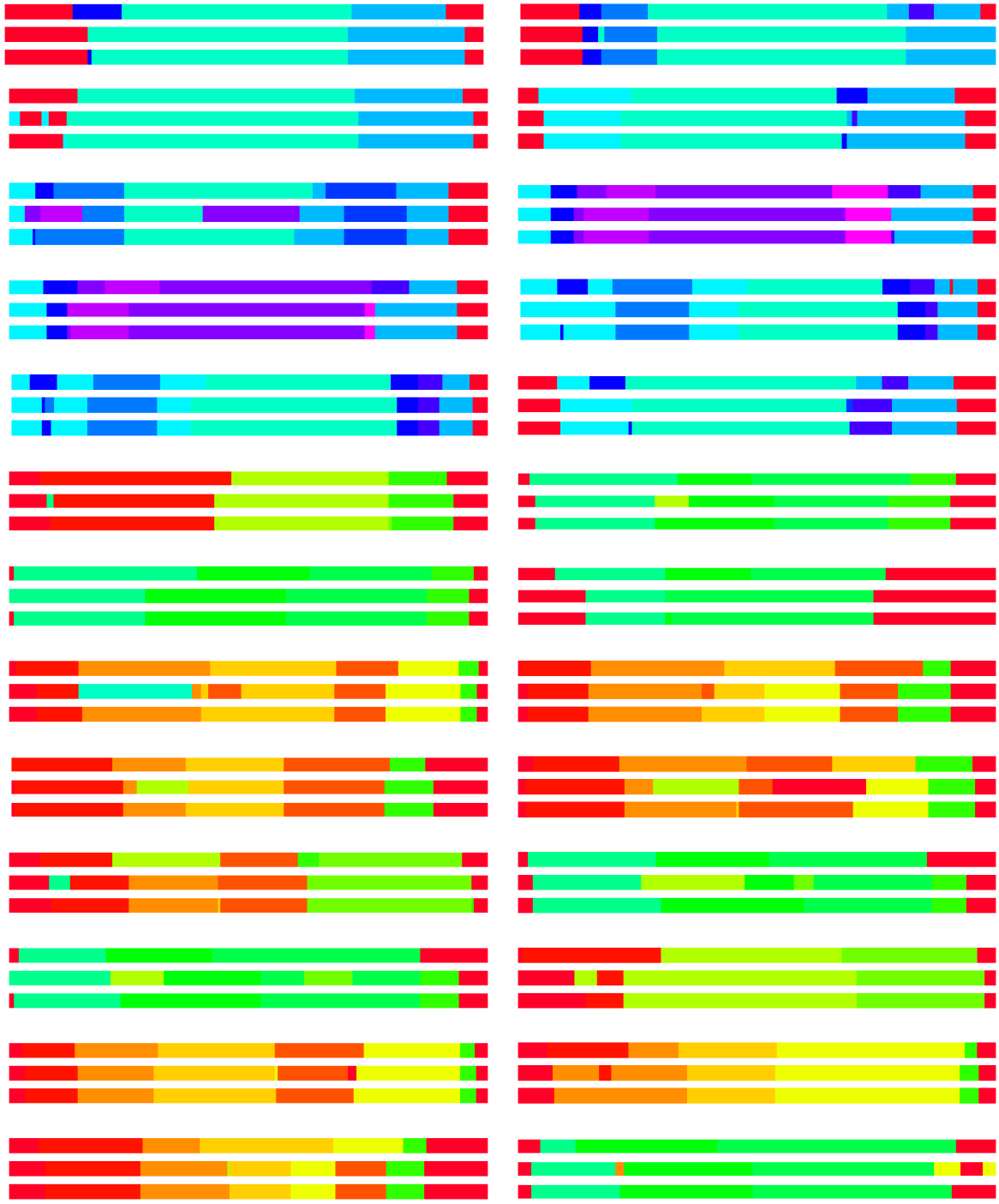


Figure 5.3: Qualitative results of segmentation results. In each group of four segmentations, the rows from the top to the bottom shows: 1) ground-truth, 2) results of ST-AOG, 3) Bi-LSTM, and 4) Bi-LSTM + generalized Earley parser.

# CHAPTER 6

# Conclusion

In this work, we present the generalized Earley parser for parsing sequence data according to symbolic grammars. Unlike previous methods, we tackle the spatial-temporal learning problem for human activity recognition and prediction with the combination of neural networks and grammar models. In our model, detections and predictions are formulated as a grammar constrained parsing problem on the probabilistic outputs of neural networks. Such a formulation offers the ability to handle frame-wise input with neural network and high-level parsing/prediction with the stochastic grammars. As the grammar learned for the temporal structure is symbolic, we argue that the learned symbolic grammars can be easily adapted to other scenarios where the temporal order of activities are the same but the scene layout is different. In such a way, we can use the generalized Earley parser as a add-on module for current classifiers to achieve better results when the tasks are similar in the temporal domain. 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 and a good starting point for better solving the spatial-temporal reasoning problem.

# REFERENCES


[Cho02] Noam Chomsky. *Syntactic structures*. 2002.

[CLS15] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. "P-cnn: Pose-based cnn features for action recognition." In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[CZ17] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[DWW15] Yong Du, Wei Wang, and Liang Wang. "Hierarchical recurrent neural network for skeleton based action recognition." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[Ear70] Jay Earley. "An efficient context-free parsing algorithm." *Communications of the ACM*, 1970.

[GFG06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In *IEEE International Conference on Machine Learning (ICML)*, 2006.

[HZG16] Steven Holtzen, Yibiao Zhao, Tao Gao, Joshua B Tenenbaum, and Song-Chun Zhu. "Inferring human intent from video by sampling hierarchical plans." In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[JZS16] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. "Structural-RNN: Deep learning on spatio-temporal graphs." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[KGS13] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. "Learning human activities and object affordances from rgb-d videos." *International Journal of Robotics Research (IJRR)*, 2013.

[KS16] Hema S Koppula and Ashutosh Saxena. "Anticipating human activities using object affordances for reactive robotic response." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[KTS14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-scale video classification with convolutional neural networks." In *CVPR*, 2014.

[LLK07] Benjamin Laxton, Jongwoo Lim, and David Kriegman. "Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[NCF10] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. "Modeling temporal structure of decomposable motion segments for activity classification." In *European Conference on Computer Vision (ECCV)*, 2010.

[PJZ11] Mingtao Pei, Yunde Jia, and Song-Chun Zhu. "Parsing video events with goal inference and intent prediction." In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[QHW17] Siyuan Qi, Siyuan Huang, Ping Wei, and Song-Chun Zhu. "Predicting Human Activities Using Stochastic Grammar." In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[QJZ18] Siyuan Qi, Baoxiong Jia, and Song-Chun Zhu. "Generalized Earley Parser: Bridging Symbolic Grammars and Sequence Data for Future Prediction." In *IEEE International Conference on Machine Learning (ICML)*, 2018.

[QWJ18] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. "Learning human-object interactions by graph parsing neural networks." In *European Conference on Computer Vision (ECCV)*, 2018.

[SHR05] Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. "Unsupervised learning of natural languages." *Proceedings of the National Academy of Sciences (PNAS)*, 2005.

[SZ14] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[TBF15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks." In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[TFK12] Kevin Tang, Li Fei-Fei, and Daphne Koller. "Learning latent temporal structure for complex event detection." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[TH08] Christian Thurau and Václav Hlavác. "Pose primitive based human action recognition in videos or still images." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[TPZ13] Kewei Tu, Maria Pavlovskaia, and Song-Chun Zhu. "Unsupervised structure learning of stochastic and-or grammars." In *Conference on Neural Information Processing Systems (NeurIPS)*, 2013.

[Vit67] Andrew Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE transactions on Information Theory*, 1967.

[VYZ17] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. "Learning to generate long-term future via hierarchical prediction." In *IEEE International Conference on Machine Learning (ICML)*, 2017.

[WBR07] Daniel Weinland, Edmond Boyer, and Remi Ronfard. "Action recognition from arbitrary views using 3d exemplars." In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[WMG17] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. "The pose knows: Video forecasting by generating pose futures." In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[WZS15] Chenxia Wu, Jiemi Zhang, Silvio Savarese, and Ashutosh Saxena. "Watch-n-Patch: Unsupervised Understanding of Actions and Relations." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[WZZ16] Ping Wei, Yibiao Zhao, Nanning Zheng, and Song-Chun Zhu. "Modeling 4d human-object interactions for joint event segmentation, recognition, and object localization." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[ZM07] Song-Chun Zhu, David Mumford, et al. "A stochastic grammar of images." *Foundations and Trends® in Computer Graphics and Vision*, 2007.