Learning Behavior Profiles from Noisy Sequences

SHARADA VALIVETI

Contents

Introduction

- Capturing abstract pattern of temporal evolution process
- Profiles modeled as Finite State Stochastic Automata
  - Using Hidden Markov Models
- Agent profiling is used in IDS and FDS
  - Is an Abstract characterization of agent activity
  - Can be used to check for normal or anomalous behaviors
- Learning by induction from logs of agent behavior
- Agent behavior is short sequence of actions
  - Execution
  - Interleaving with phases
    - Where activities cannot be modeled because it is non-repetitive

Introduction

- Attributes are used to set constraints on atomic events
  - Problem of discovering structure of profile turns to the problem of learning probabilistic regular expressions from sequences containing gaps and noises
- Inference of regular expressions from data is done using
  - Computational learning theory
  - Neural networks
  - Syntactic pattern recognition
  - Probabilistic automata

Introduction

- Abstraction mechanism
  - Allows a process behavior to be seen at different levels of granularity
    - Exploited by learning algorithm
    - Method for detection and learning recurrent structures inside an event is novel in presence of noise
- A real agent profiling task is designed
  - Challenge is to characterize behavior of a user typing on keyboard
- Algorithm was successful in discovering profiles, which identify an user from another

Learning by Abstraction

- Difficulty in discovering and modeling profiles
  - Presence of long gaps, filled by irrelevant facts
  - Statistical correlations are difficult to detect
  - Complexity of mining algorithm increases with length of portion of sequence to be searched to detect such correlations
- Strategy is to cope with such kind of problems
- By replacing a sub-expression with a new symbol, an abstract expression is obtained
- Profiles can be described by means of regular expressions extended with attributes
- It can be abstracted or de-abstracted

Learning by Abstraction

- Scheme of algorithm used for discovering profiles hidden in a set $\mathscr{LS}$ (Learning Sequences)
  - Constructs an abstraction hierarchy, layer after layer (Bottom-up)
  - Identify episodes (characterized by $\mathscr{R}$) occurring with a relevant frequency in $\mathscr{LS}$
  - Detected episodes are named by a new symbol
  - These names become alphabet for describing $\mathscr{LS}$ at next abstraction level
  - Every sequence in $\mathscr{LS}$ is rewritten by replacing every episode instance occurring in it with corresponding episode name
  - Subsequences of consecutive atomic events which have not been included in any episode are replaced with symbol denoting gap

Learning by Abstraction

- Gaps between episodes are considered special kind of episodes
- Subsequences of irrelevant facts may become consecutive
- Important aspects to consider for statistical correlation between consecutive atomic events
  - Event duration and distance from one another
  - Correlation between two event A and B
- Every atomic event E is
  - Denoted by a name (Symbol)
  - An attribute l_E reporting the length (duration) of E on unabstracted sequence

Regular Expressions

- Regular language syntax contains
  - Metasymbols
    - To denote disjunction (|) and iteration
    - ε – Null Symbol
    - Repetition (denoted by superscript on a symbol)
    - Constraints on event / gap length may be set by annotating symbols in regular expressions

String Alignment and Flexible Matching

- Key role in abstraction process is played by approximate matching of strings and of regular expressions
- Global Alignment
  - For s_1 and s_2, let s_1' and s_2' be two strings obtained from s_1 and s_2 , inserting arbitrary number of spaces such that atomic events in both can be put in one to one correspondence
  - Local and multi-alignment must be defined
- Local Alignment
  - Any global alignment between a pair of substrings r_1 and r_2 extracted from 2 strings s_1 and s_2 respectively is local alignment $LA(s_1, s_2)$

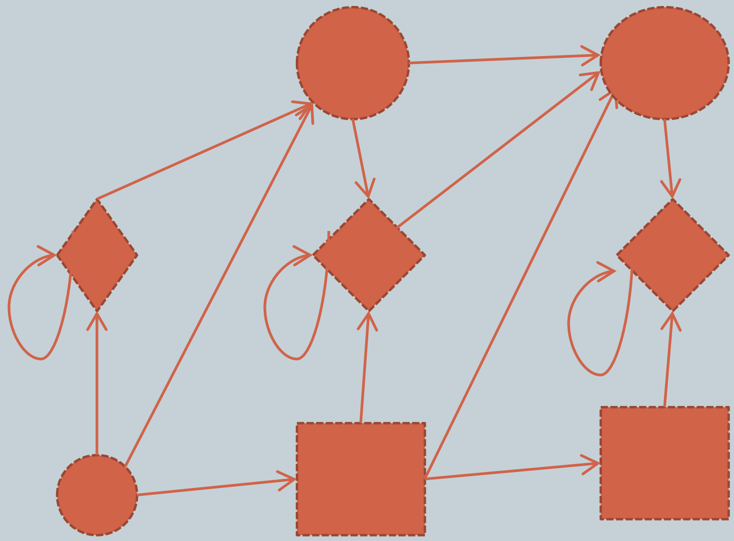
String Alignment and Flexible Matching

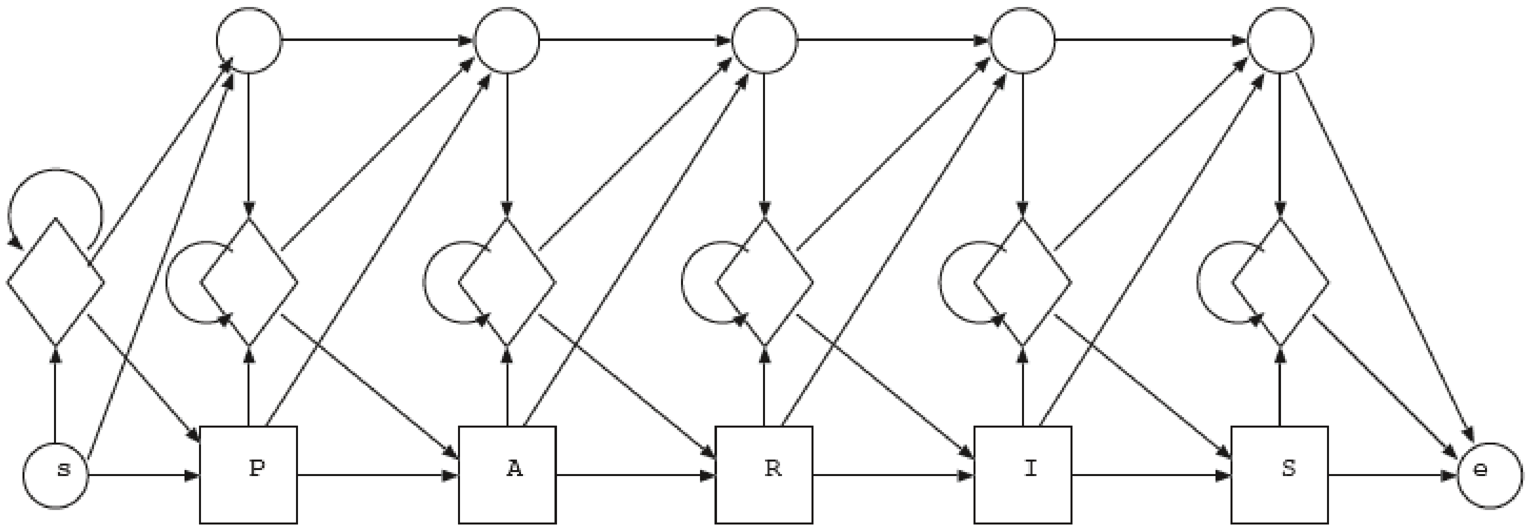
- Multi-alignment
  - A multi-alignment MA(S) on S is a set S' of strings, where every string s ϵ S generates a corresponding string s' ϵ S' by inserting a proper number of spaces, and every pair of strings (s_1',s_2') is a global alignment A(s_1,s_2) of the corresponding strings s_1, s_2 in set S
- Scoring Function
  - n is length of the alignment

$$\int(s_1, s_2) = \sum_{i=1}^{n} \int(s_1'(i), s_2'(i))$$

String Alignment and Flexible Matching

- Hidden Markov Model (Profile HMM)
  - Has three types of states
    - Match states (emission corresponds to expected nominal symbol)
    - Null emission states (modeling deletion errors)
    - Insertion states (Modeling insertion errors)





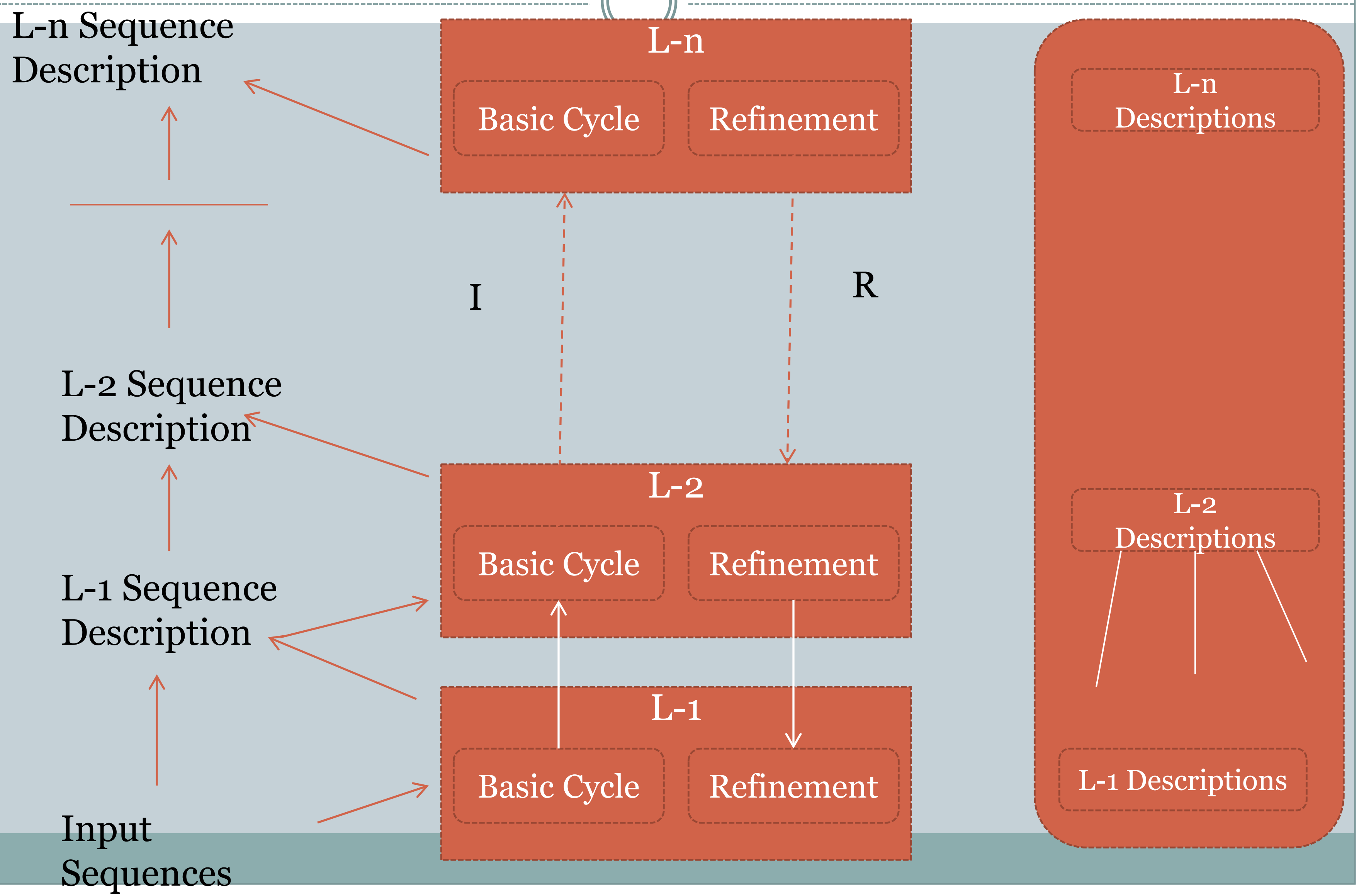
Profile HMM obtained from the string ”PARIS”.

- Square nodes represent *match states*
- *C*ircles represent *null emission states* and diamonds represents *insertion states*
- Transitions, from one state to another, and
- Emissions are governed by probability distributions not shown in the figure
- States labeled by *s* and *e* are the initial and final state, respectively

String Alignment and Flexible Matching

- In dynamic Programming, it can be solved in O(nm)
- Regular expressions can be translated into HMMs by following augmentations:
  - Extra states added to deal with insertion and deletion errors
  - Cycles in regular expressions need to be unrolled into a feed forward graph to model probability distributions
- Impact on final alignment will depend on the specific scoring function
- Iterations in excess will be considered as (insertion or deletion) errors

The Learning Algorithm



The Learning Algorithm

- ω_s and ω_I used for Basic Cycle and Refinement Cycle
- ω_s constructs regular expressions non containing iterative constructs
- ω_I aims at discovering and abstracting iterative constructs
- Abstraction hierarchy is obtained by interleaving two operators

The Learning Algorithm

- ω_s Operator
  - Takes set S of similar substrings, detected using a local alignment algorithm and constructs an abstract atomic event defined as a pair <$\mathscr{R}$, E>
    - $\mathscr{R}$ is regular expression generalizing the episode instances contained in S
    - E is abstract event associated to $\mathscr{R}$
- Algorithm ω_s
  - Construct the multi-alignment MA(S) table for strings in S
    - Columns contain symbols put in correspondence by alignment algorithm
  - Construct the match graph MG(S)
    - Removal of noise from MA(S)
  - Transform MG(S) into an equivalent regular expression
    - If there is at least one row in MA(S) where a match symbol x follows a match symbol y, immediately or after one or more spaces; a link from x to y is set in MG(S)

The Learning Algorithm

Paris

Party

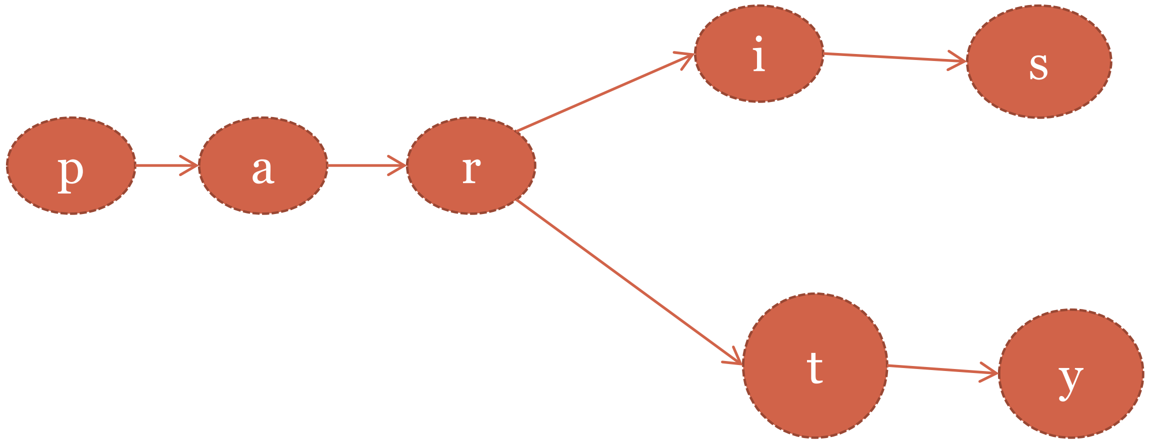
Part -

Parys

Pray -

Pay - -

Example of non-iterative expression obtained from string set



Retained Alternatives

Par(is|ty)

Final Regular Expression

The Learning Algorithm

- ω_I operator
  - Explicitly searches for contiguous repetitions of a same substring inside a given string s
  - By computing self-correlation of string similarity function
  - W_i = reference window
  - w_j = sliding window
  - s = Size of sliding window and reference window
  - n=length of s minus length of W_i
  - SC is a triangular matrix of size $n^2/2$
  - SC(i, j) indicates i, j element of SC

The Learning Algorithm

- Self correlation Algorithm
  1. Set i =1
  2. For j=i to n, evaluate SC(i, j) = ∫(W_i , W_j) between substrings selected by W_i and w_j, respectively
  3. Set i=i+1
  4. If i is smaller than n, goto step 2, otherwise continue
  5. Detect chains of maxima on SC, where maximum value is close to maximum possible similarity value between two substrings W_i,w_j . A substring r of s, laying in between two consecutive maxima, is an iterated substring
  6. For every different iterated substring r construct a new hypothesis for an iterated episode
- Complexity of algorithm is $O(n^2/2)$

The Learning Algorithm

- Basic Learning cycle
  - Non – iterative episode detection (Using operator ω_s)
  - Iterative episode detection (Using operator ω_I)
  - Model construction (An HMM is constructed for every abstracted episode when necessary)
  - Sequence abstraction (input sequences are rewritten using as new alphabet the names of abstract episodes)
    - Every sequence s is scanned left-to-right searching for instances of episodes detected and abstracted in previous steps
    - Presence of episode E is decided by matching corresponding regular expression $\mathscr{R}_E$ to s.

The Learning Algorithm (Example)

<table>
<tr><td>. . . . accacbbbbsthsturlm..
. . . zacacbbbbbbststuhbn
. . . . acacbbbbbstfstubkku..</td><td>$B::=(b)^{4,7}$
. . . . accacBsthsturlm..
..zacacBststuhbn..
. . . . acacBstfstubkku. .</td></tr>
<tr><td colspan="2">Iterated Symbols are detected and replaced with name of corresponding regular expression</td></tr>
<tr><td>. . . . accacBsthsturlm . .
. . . Zac_acBst_stuhbn..
. . . zacacBst_stuhbn. .
. . . . acacBstfstubkku . .</td><td>accacBsthstu
acacBststu
acacBstfstu</td></tr>
<tr><td colspan="2">Local alignments are detected and similar substrings are clustered together</td></tr>
<tr><td>accacBsthstu
ac_acBst_stu
ac_acBstfstu</td><td>ac(c|ε)acBst(h|f|ε)stu</td></tr>
<tr><td colspan="2">From multiple alignment of elements in a same cluster a regular expression is obtained</td></tr>
</table>

The Learning Algorithm

- Refinement Cycle
  - May be activated at abstraction layer L_i every time new episodes are detected and modeled at a level higher than i
    - When an episode E is hypothesized and characterized at an abstraction level L_i , the context i.e. presence of other episodes before or after E, is not considered
    - The context is considered later when the episodes of Layer L_i are linked together into an episode at level l_{i+1}
    - Regular expression describing E are re-learned using instances that have been retained

User Profiling

- Widely used to detect intrusions in computer or in telephony networks
- Possibility of automatically building profile for users or for network services reflecting temporal behavior would offer a significant help to the deployment of adaptive IDS
- Assumption is that every user has a different way of typing
- 2 experiments were performed

User Profiling

- Key Phrase Typing Model
  - Goal was to construct a model for a user typing a key phrase, discriminant enough to recognize user among others
  - Selected sentence of 22 syllables typed many times on same keyboard
  - An algorithm recorded, for every typed key, the duration of each stroke and the delay between two consecutive strokes
  - Every repetition of sentence generated a sequence, where every key stroke corresponded to an atomic event; delay between two strokes was represented as a gap, whose length was set to the corresponding duration
  - User profile based on a key phrase is too restricted

User Profiling

- Text Typing Model
  - Modeling a user during a text editing activity
  - An entire set of data to enter was selected where user would type from the set of Newspapers in different session and at different times
  - Results have been obtained without requiring any tuning of algorithm
  - Model is robust and easy to apply to such problems

Conclusions

- Agent behavior modeled by means of probabilistic regular expressions
- Agent model is an abstract characterization of agent activity used to check for normal behavior or anomalous behaviors
- Cascade of regular expressions generated by abstraction mechanism leads to a Hierarchical Hidden Markov Model that offers a framework which is powerful enough to model many real world problems
- Has affordable computational complexity