DETECTION AND SIMULATION OF SCENARIOS WITH HIDDEN MARKOV MODELS AND EVENT DEPENDENCY GRAPHS*

W. M. Campbell, S. Barrett, J. Acevedo-Aviles, B. Delaney, C. Weinstein

MIT Lincoln Laboratory, Information Systems Technology Group, Lexington, MA, USA
{wcampbell,joel}@ll.mit.edu

ABSTRACT

The wide availability of signal processing and language tools to extract structured data from raw content has created a new opportunity for the processing of structured signals. In this work, we explore models for the simulation and recognition of scenarios—i.e., time sequences of structured data. For simulation, we construct two models—hidden Markov models (HMMs) and event dependency graphs. Combined, these two simulation methods allow the specification of dependencies in event ordering, simultaneous execution of multiple scenarios, and evolving networks of data. For scenario recognition, we consider the application of multi-grained HMMs. We explore, in detail, mismatch between training scenarios and simulated test scenarios. The methods are applied to terrorist scenario detection with a simulation coded by a subject matter expert.

Index Terms— Hidden Markov Models, Goal Recognition

1. INTRODUCTION

Many signal processing application have focused on the process of converting raw data to structured form. For speech applications, the lexical content as well as the speaker, dialect, and language can be extracted [1]. For language processing, technologies like information extraction can take raw text and convert it to structured data that encodes entities, relations, and events [2].

In many applications, extracting structure from content is only part of the problem. With large unstructured masses of multimedia data available on the world wide web, converting raw data to another form creates a new flood of structured data. Therefore, methods which look at processing across multiple documents, signals, images, etc. become critical. For this paper, we consider the detection and simulation of time-sequenced structured data in the form of scenarios. This area has many interesting applications including network intrusion detection, terrorist scenario recognition, multi-sensor fusion, etc. In this paper, we will focus on terrorist scenario recognition from multimedia content to give concrete examples and address a relevant real-world application. Prior related work in this area includes research in keyhole goal recognition [3, 4, 5], partial-order planning [6], and Petri nets [7].

Our first area of exploration is simulation of scenarios. Simulation is a necessary component of research, since in many cases we may want to model unseen scenarios. In a broad sense, one major use of simulation is for a subject matter expert (SME) to code a hypothetical scenario which could then be used to query (detect) the scenario in a multimedia database. Other complimentary uses of simulation are to perform parameter tradeoff studies (such as the ones in this paper) and provide formal descriptions. We describe discrete HMMs and event dependency graphs (EDGs) for simulation.

Our second area of exploration is recognition of sequences of structured data. In prior work [8], we used support vector machines to recognize scenarios from structured data. A drawback of this approach is that it does not provide the time evolution of scenario execution. Therefore, we explore the use of discrete HMMs for recognition. We propose a *multi-grained* model for recognition—i.e., a model which is trained to recognize the coarse structure of a scenario. We explore different types of mismatch which could occur and explore their effect on recognition.

The outline of the paper is as follows. In Section 2, we outline a basic framework for scenario simulation and processing. In Section 3, we describe methods for scenario simulation with HMMs and EDGs. Section 4 discusses multi-grained HMM recognition and mismatch. Finally, in Section 5, we present results for scenario detection under various mismatch conditions with a terrorist scenario constructed by a SME.

2. SCENARIO PROCESSING FRAMEWORK

2.1. Information Extraction and Graphical Data

We represent the extraction of structured content from raw data using standard knowledge representation methods [6]. For every input document (signal, image, etc.), we produce a set of objects, attributes, and predicates conforming to an ontology that describe structured information in the document. The ontology is based on standards for information extraction; i.e., primarily the ACE protocol [2]. We limit ourselves to first-order binary predicates. We also use a typed (many-sorted) logic. A typical example extraction from a document might be *Travels(Bob, NewYorkCity)* where *Bob* is an object of type *per* (a person). We group multiple predicate statements together from a document and call these *transactions*.

We represent four basic items in our ontology. First, typed objects encode people, events, and groups. Second, objects can have attributes. For instance, we might extract *ATT_age(Bob,25)*. Third, predicates representing network relations establish a graph between objects; a typical network relation might be *Knows(Bob,John)*. Fourth, binary predicates are used to represent other types of knowledge; e.g., *GivesMoneyTo(John,Bob)*.

2.2. Processing Structured Data

Our framework for simulation and recognition is shown in Figure 1. A SME encodes a representation of a scenario for simulation that describes, e.g., a bombing. The simulation is stochastic and can be used to general multiple distinct instances that are used for model training. As a separate process, information extraction has been applied to a data store to produce structured information which is then

*This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

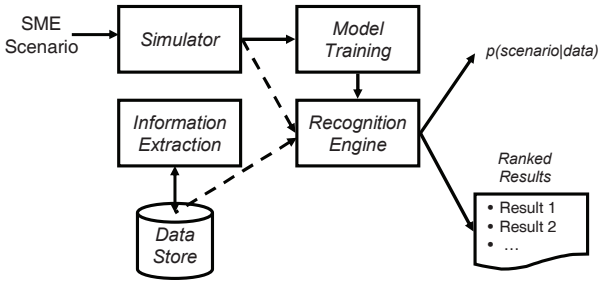
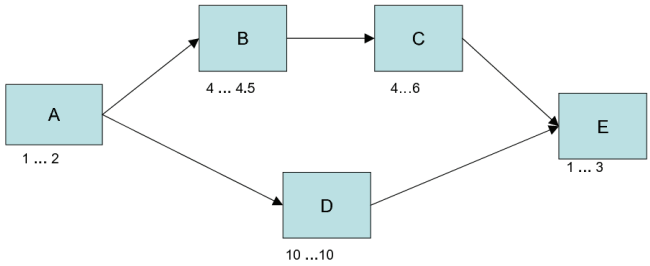


Fig. 1. Framework for Scenario Simulation and Recognition



Output: $a_1a_2...a_{LA}\ d_1b_1d_2b_2...b_{LB}c_1d_5...d_{LD}\ ...\ c_{Lc}\ e_1e_2\ ...\ e_{LE}$

Fig. 2. Event Dependency Graph

put back in the data store. After these two steps have been completed, we can then perform detection using a recognition engine either on simulated data or on the data store. Results can be returned as a list of documents ranked by match to the scenario. Alternatively, a detection score could be produced and a corresponding decision, scenario present/not-present could be returned.

3. SIMULATING SCENARIOS

3.1. Hidden Markov Model Approach

3.1.1. Overview

Our simulation framework consists of three main components: an HMM, a knowledge base (KB) and an ontology. The HMM [9] determines the order in which transactions can be observed and the likelihood of each possible ordering. It also provides a way to concisely represent the multiple courses of action that can be taken at different points during the development of a scenario. The knowledge base (KB) stores the objects and predicate instances that exist in the simulated world at each point in time. Its contents can be modified at any point during simulation to reflect changes in the scenario's actors, their attributes and the relationships between them. The last component, the ontology, has been described in Section 2.1.

3.1.2. Hidden Markov Model

HMMs are at the core of our modeling and simulation scheme for generic scenarios. The structure of the HMM determines the overall flow of events that constitute the scenario. A scenario is therefore divided into states; the distinct phases a scenario *might* go through during its development.

We assume a discrete HMM with state transition matrix, $A = \{a_{ij}\}$. Each state of the HMM, contains a set of transactions that are representative of that state. The observation symbol probability distribution, $B = \{b_j(x_k)\}$, describes the probability of observing transaction x_k when the current state of the HMM is s_j. In addition to predicates, transaction definitions can include one or more directives (e.g., "Generic Network Predicate"). Directives are special constructs to randomly select predicates from the knowledge base and make them part of a transaction.

3.1.3. Simulation with an HMM

In this section we outline the overall simulation process:

1. Execute initial KB modifications to introduce initial actors, their attributes and the initial state of the network.
2. Set current state, q_t = initial state of HMM
3. **If** first time in q_t,
   Execute all KB modifications specified in q_t
4. Randomly choose a transaction, x, from all the transactions in q_t, according to distribution B. If the transaction includes directives, execute them and append resulting predicates to x.
5. Randomly choose the next state, q_{t+1}, based on the transition probability distribution A
6. Output x
7. Set $q_t = q_{t+1}$
8. **If** q_t = final state of HMM, exit. Otherwise, go to Step 3.

3.2. Event Dependency Graphs (EDGs)

One drawback of using HMMs to model scenarios is that it is difficult to represent phases that occur in parallel. EDGs leverage our previous approach by providing an effective way of representing concurrency in scenarios. Like HMMs, an EDG is a stochastic model that encodes the complete execution of a scenario. It is used in conjunction with one or more HMMs in a hierarchical 2-level model. An EDG represents the evolution of a scenario through coarse-grained components called *stages*. These stages represent high-level phases; all of which have to occur for a scenario to be complete. Each stage in turn is modeled with an HMM of the kind exposed in the previous sections. Notice the difference in the two approaches. In the first one, an HMM stands for a complete scenario. In the second, scenarios are composed of one or more HMMs, each of which represents an aspect of the scenario at a fine-grained level. In addition, all the stages of the EDG are executed in a simulation run whereas in HMMs only states in a single path are executed.

EDGs are essentially constraint graphs (see Figure 2). Each node corresponds to a stage and therefore to an HMM. Edges correspond to ordering constraints between stages. An edge from stage A to stage B represents a constraint of the form $A \prec B$: A has to be executed from start to finish sometime before B but not necessarily immediately before B (analogous to partial-order planning [6]).

We have mentioned that stages execute in parallel but we have not defined what this means. Since the output of our simulation and of each stage is a sequence of transactions, when n stages execute in parallel the result should be a new sequence that interleaves the output of each stage (see Figure 2). To generate the output sequence, we use an algorithm that allows us to interleave transaction sequences from concurrent stages on-the-fly, i.e. without having to execute all concurrent HMMs from start to end. To understand how we achieve this, we introduce the notion of time.

In many domains, it might be difficult to assess the time duration of each transaction or predicate. For this reason, we opted for specifying time durations at the stage level. For each stage, a range of time is specified representing a lower and upper bound on the duration of the stage. This time duration can be absolute or relative to the duration of the other stages. Before starting execution of the EDG, a specific time value T_i is chosen, within the specified interval, for each stage i. This becomes the duration of each stage for that simulation run. On-the-fly interleaving is based on determining the time

instant when the next transaction of each concurrent stage will take place and executing the HMM which next transaction time is closest to the current simulation time.

The time stamp $\phi_{P,t+1}$ of the next transaction $x_{P,t+1}$ generated by stage P is determined by the time gap $\delta_{P,t+1}$ between $x_{P,t}$ and $x_{P,t+1}$. The time interval $\delta_{P,t+1}$ is determined stochastically based on the structure of the HMM and duration of P. For the sake of explanation, let us assume that the duration of P has been randomly chosen and that a time stamp is given by an integer number representing a time displacement from the beginning of the simulation ($t = 0$). Let us define the maximum time interval between consecutive transactions of stage P, τ_P, as follows: the stage duration T_P divided by the expected value of the length of P's HMM output sequence, L_P. Formally, $\tau_P = T_p/E[L_P]$. With the necessary definitions in place, we assign a value to $\phi_{P,t+1}$ performing the following two steps: 1) $\delta_{P,t+1}$ = random value in the interval $[0, \tau_P]$, 2) $\phi_{P,t+1}$ =(current simulation time)+$\delta_{P,t+1}$.

3.2.1. Simulation with EDG

In this section we outline the EDG execution process.

1. $G =$ event dependency graph
2. Add all unconstrained nodes of G to set U. According to the semantics of the EDGs an unconstrained node has no in-edges.
3. **Foreach** node, u, in U
   Determine the time stamp $\phi_{u,t+1}$ of the next transaction that will be generated by HMM in u if it has not been calculated before.
4. Choose node in U with minimal $\phi_{u,t+1}$ based on the calculations from the previous step. Let us call this node N.
5. Execute HMM in N to get transaction $x_{N,t+1}$. (Refer to section 3.1.3)
6. Output transaction $x_{N,t+1}$
7. **If** the current state of the HMM in N is its final state
   (a) Remove all the edges involving N from G.
   (b) Remove node N from set U and graph G
8. **If** $U == \{\emptyset\}$
   Exit
   Else
   Go to Step 2

3.3. Graphical Clutter Simulation

Clutter generation consists of three parts. For the first part, we create a clutter graph of individuals using the R-mat algorithm [10]. R-mat creates a random graph with a power-law degree distribution—a commonly observed distribution in social networks [11]. Connections between nodes for the R-mat algorithm are kept if at least one individual is not in the scenario; this strategy preserves the structure of the scenario social network connections during simulation. The second part of clutter generation is random transactions on or between nodes in the clutter graph. A prior over transaction types is pre-specified, e.g., *p(travel(per,location))*, and the distribution is assumed to be stationary during the simulation. The final clutter generation parameter varies the duty cycle $(0 - 100\%)$ between scenario transactions and clutter transactions. The duty cycle indicates the asymptotic ratio of the number of scenario to clutter transactions output by the simulator.

4. SCENARIO RECOGNITION WITH HMMS

4.1. Multi-Grain Recognition Structure

HMMs are an appropriate tool for scenario recognition due to their ability to correlate observations with hidden states. Observations correspond to the actions and connections that can be seen in a social network. The hidden states are the fine details of the scenario, which cannot be directly detected by observers, such as subtasks in recruitment and reconnaissance. Using the most likely path, HMMs can be used to determine the progress of a scenario, enabling human operators to determine if it is necessary to intervene.

When designing an HMM for recognition, it is important to consider the granularity of the HMM. Having more states makes it easier to recognize a scenario if the recognition model *exactly* matches the generation model. Unfortunately, in the real world, it is likely that actual scenario will not match the recognition model, so using the most specific model may not be optimal. Also, human operators that are trying to detect scenarios may not care about the finer-grained parts of the scenario; often identifying the larger actions such as recruitment or reconnaissance is enough.

To classify unlabeled examples, the system chooses the class, c^*, most likely to have generated the episode using $c^* = \underset{c}{\operatorname{argmax}}\, P(c|\hat{x})$ where $\hat{x} = x_1, x_2, \ldots, x_n$ is the sequence of time-ordered observations. $P(\hat{x}|c)$ was calculated using the forward algorithm and Bayes rule was applied to obtain the posterior.

4.2. Mismatch and Recognition

Note that the models created by subject experts may not exactly match future terrorist attacks. Furthermore, even if future terrorist attacks happen in a similar fashion, some of their actions may go unobserved and noise in the information extraction may add observations of irrelevant actions. Therefore, the question is: will recognition models learned on the training data provided by the experts' models detect new terrorist attacks? We explore this problem by considerings several mismatches—permutation of action ordering, dropouts of actions, and duration mismatch.

5. EXPERIMENTS

5.1. Experimental Setup

For the experiments, subject matter experts created a model of the Valentine's Day Attack, which occurred on Feb 14, 2005 in the Philippines by the terrorist organization Abu Sayyaf. This model was used to generate positive training examples. A mismatched model was created based on the experts' model, and was used to generate the positive testing examples. Randomly created clutter models were used to generate the negative training and testing examples. The recognition model with differing amounts of granularity was trained using simulated data based on the experts' model. To create these models, some states represented separately in the experts' model were represented as single states in the recognition model.

In each run, the training and testing data sets each consisted of 50 episodes of scenario mixed with clutter and 50 episodes of pure clutter, and for each test 100 runs were used. In the positive episodes, the scenario had a 20% duty cycle, and the clutter networks were generated with 64 nodes and 100 edges. Unless otherwise stated, the clutter transactions were generated uniformly with respect to the transaction type, and the testing scenarios were truncated from state 11 on. This truncation leaves approximately 58.8% of the states and 66.7% of the expected scenario length. This truncation was used because it simulates the need to recognize a possible attack in the preparation stages, rather than after a bombing has occurred.

The scenario and clutter recognition HMMs' output and transition probabilities were trained on the positive and negative training examples, respectively. Then, testing examples were assigned to the most probable recognition HMM. The specific mismatch tests run were— **1)** no mismatch between the training and testing models and

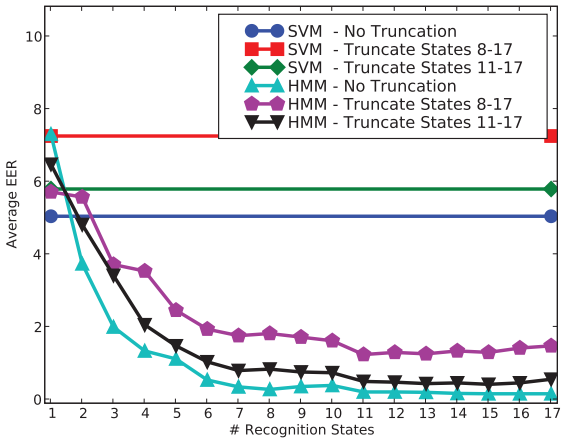


Fig. 3. Effects of scenario truncation on recognition

a truncation of states 11-17, leaving on average 66.7% of the scenario actions, **2)** dropping 5 randomly chosen states from the testing data generation model, **3)** setting the probabilities of transitioning to the next state to a random value in 1-15%, while the original model had transition probabilities of 5%, **4)** reversing states 1-5, on average 50% of the observed actions, **5)** reversing states 1-10, 100% of the observed actions, **6)** truncating states 8-17 leaving on average 46.7% of the scenario actions, and **7)** noise with the distribution of actions matched to that of the scenario.

5.2. Experimental Results

Results are shown in Figure 3 and Table 1. Performance is given in terms of average equal error rate (EER) across multiple Monte Carlo runs. An interesting result shown is that having finer granularity in the recognition model improves the recognition EER. Even in the cases where the finer granularity made the recognition model structurally match the testing data generation model, the finer grained models outperform the coarser grained models. One likely cause for this is that the output probabilities of the finer grained models are more accurately tuned to that of the scenario, creating better separation between the noise and the scenario. It is interesting to note that there is little difference between having 7 and 17 states in the recognition model. Most of the improvement comes from having small number of states rather than just a single state, like the SVM.

Investigating the effects of truncation on the ability to recognize scenarios is important because it is desirable to detect attacks in the planning stages, before a bombing takes place. Depending on the scenario and the time needed to respond to an attack, it may be necessary to detect the scenario in different places. Figure 3 and Table 1 show the ability of the system to detect the scenario with no truncation, states 8-17 truncated (leaving 41.2% of the states and 46.7% of the expected scenario length), and states 11-17 truncated (leaving 58.8% of the states and 66.7% of the expected scenario length). The results indicate that attempting to detect a scenario earlier in its execution is harder for the system, but the system still has very good results. Furthermore, the HMM detection outperforms the SVM detection consistently, and both methods are similarly affected by the scenario truncation.

Table 1 shows the effects of mismatch between the training data generation model and the testing data generation model. The results show that the recognition model is effective despite the mismatch. The worst performance comes from a full reversal of the observable states, which is expected. Full reversal of a scenario is

Table 1. Effects of mismatch, truncation, and noise on EER

Testing Data / Generation Model	# Recognition States 3	7	17	SVM
No Mismatch	3.42%	0.80%	0.56%	5.80%
Drop 5 States	6.78%	2.92%	2.96%	10.16%
Transition Prob 1%-15%	4.66%	1.74%	1.16%	7.08%
Reverse States 1-5	3.46%	2.28%	2.24%	5.80%
Reverse States 1-10	9.46%	9.28%	8.86%	5.80%
Truncate States 8-17	3.72%	1.76%	1.48%	7.26%
No Truncation	2.00%	0.35%	0.16%	5.05%
Clutter Matched to Scenario	13.90%	3.46%	2.26%	40.74%

the worst case scenario, but it is also not logically possible in most scenarios. Reversing a scenario means that a bombing might happen before any planning or recruitment. With smaller reversals, the HMM performs well, only slightly affected by mismatch. Note that the SVM recognition is unaffected by reversing the scenario as the SVM recognition ignores the order of events. With clutter matched to the scenario, performance with few numbers of states is poor due to each state's output probabilities only slightly differing from the noise output probabilities. However, as the number of recognition states increases, the output probabilities differ more from the noise, so the EER improves significantly.

6. CONCLUSIONS

We have demonstrated a framework for simulation and recognition of scenarios with applications to terrorist activities. Models robust to mismatch were proposed for simulation and recognition. Future work includes applying these techniques to open source corpora and creating advanced feature sets for representing transactions.

7. REFERENCES

[1] W.M. Campbell, J.P. Campbell, D.A. Reynolds, E. Singer, and P.A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech & Language*, vol. 20, no. 2-3, pp. 210–229, 2006.

[2] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel, "The automatic content extraction (ACE) program—tasks, data, and evaluation," in *LREC 2004: Fourth International Conference on Language Resources and Evaluation*, 2004.

[3] N. Blaylock and James Allen, *Generating artificial corpora for plan recognition in User Modeling 2005*, pp. 179–188, Number 3538 in Lecture Notes in Artificial Intelligence. Springer, Edinburgh, 2005.

[4] S. Singh, W. Donat, J. Lu, K. Pattipati, and P. Willett, "An advanced system for modeling asymmetric threats," in *IEEE Conference on Systems, Man, and Cybernetics*, 2006, pp. 3943–3948.

[5] D. W. Albrecht, I. Zukerman, and A. E. Nicholson, "Bayesian models for keyhole plan recognition in an adventure game," *User Modeling and User-Adapted Interaction*, vol. 8, no. 1-2, pp. 5–47, 1998.

[6] Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 2003.

[7] Tadao Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541–580, Apr. 1989.

[8] C. J. Weinstein, W. M. Campbell, B. Delaney, and G. O'Leary, "Modeling and detection techniques for counter-terror social network analysis and recognition," in *IEEE Aerospace Conference*, 2009, pp. 1–16.

[9] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[10] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining," in *SIAM Int. Conf. on Data Mining*, 2004.

[11] A.-L. Barabsi and E. Bonabeau, "Scale-free networks," *Scientific American*, vol. 288, pp. 50–59, May 2003.