Generating Corpora of Activities of Daily Living and Towards Measuring the Corpora's Complexity

Rudolf Kadlec, Michal Čermák, Zdeněk Behan, and Cyril Brom

Faculty of Mathematics and Physics
Charles University in Prague
Czech Republic
rudolf.kadlec@gmail.com,mikajel@yahoo.com
rain@matfyz.cz,brom@ksvi.mff.cuni.cz
http://amis.ms.mff.cuni.cz

Abstract. Episodic memory modeling enjoys increasing interest in virtual agents, autonomous companions or computer games research communities. To evaluate memory models, it is often necessary to "fill" them with appropriate data - streams of activities corresponding to what would happen to a real human during a particular time period. However, such activity corpora, freely available, are, to our best knowledge, lacking. This paper has two goals. First, it shows two already implemented complementary approaches to generating activity corpora. While the first one uses HTN planning to create a corpus with relatively abstract activities, the second one utilizes a 3D simulation to generate a stream of more fine-grained actions. The key question is evaluation of the resulting corpora in terms of their resemblance to streams of activities of real humans. Thus, the second goal of this paper is to compare the generated corpora to several known datasets of human activity, based on their entropy, compressibility and statistics of transitions between actions. This can be conceived as a first step towards creating general complexity measures of streams of human actions and we see it as the main contribution of this paper.

Keywords: Virtual agents, Behavior complexity measurement, HTN planning, AND-OR trees, Compressibility, Entropy

1 Introduction

Episodic memory [30] modeling, that is, modeling a memory for personal history of an entity, enjoys increasing interest in agents and robotic research communities. Possible applications of autonomous entities with episodic memory range from videogames in large role playing games to artificial companions (reviewed in [7]).

Every day in human life brings a number of events that can be possibly remembered. When equipping an autonomous entity with episodic memory abilities, a researcher must be able to generate streams of such events to evaluate the memory model; otherwise, the model would exist in a vacuum. Two open questions are how to generate these *event streams* so that they are similar to event streams of real humans, and how to objectify the similarity. In this paper, we address these questions. We extend previous approaches to generation of event streams and present new complexity measures of these streams.

There are many types of events. Their list includes, but is not limited to, conversational events, physiological events, thought events, perceptual events concerning surrounding entities, and events about starts and ends of activities an agent is engaged in directly. Here, we specifically focus at streams of events of the last kind. Technically, these streams can be conceived as lists of activities that the agent is performing and that change the environmental state.

Such activities are of various complexities: moving a finger is a more fine-grained action than visiting grandparents in Rome. Our goal is to generate event streams with atomic level roughly corresponding to activities such as "sleep," "wake up," "brush teeth," "eat," etc. For example we are not interested in modeling every step in walking, nevertheless our approach enables adding this level of detail if needed. At the same time, our goal is to generate streams for months-long periods (think of a persistent world of a MMORPG). Thus, we also need to group these actions recursively into clusters of increasingly more abstract activities such as "eat," "dinner in restaurant" and "enjoying an evening," a hierarchical approach.

In this paper, we use two complementary methods for generating corporas of activities of daily living (henceforth ADL corpora or ADL stream). While one generates a preliminary ADL stream and then executes it in a 3D environment producing a final stream, the other employs only the first step. The 3D simulation used by the former approach can add an interesting detail to the stream, e.g. interruption of actions, but the cost is increased computational time. Consequently the former approach is suitable for generating shorter streams, e.g. for a month long period, and the latter is suitable for half year long periods. Both mechanisms output a hierarchically represented ADL corpora. These two techniques capitalize on already known approaches to action selection and corpora generation and do not present significant contribution *per se*.

The main contribution is defining new measures of behavior complexity and applying them on our corpora and also on behavior data obtained from real settings. We will demonstrate that the outputs of our two methods are similar to real world data according to these measures. From the methodological perspective, this work can be also conceived as evaluation of these measures *per se*. We discuss in the paper the possibilities of general application of the measures, their limitations and next steps. We see these measures as a first step towards a set of more expressive measures capturing also other aspects of the behavior. Relation of these measures to humans' judgment of the behavior also remains as a future work.

The rest of the paper continues as follows. In the next section we describe requirements that we have on our programs for generating the corporas. In Section 3 we detail our two programs and show how the requirements are implemented.

In Section 4 we define several measures, apply them on the generated streams and compare the results with real datasets. We close the paper with Discussion and Future Work.

2 Requirements

The goal is to generate a parameterizable corpus of ADL. Most notable requirements on the programs are:

1. Capture daily human activity regularities.
2. Create different behavior variants through nondeterminism.
3. Use hierarchical behavior representation.
4. Provide non-trivial number of alternative behavior decompositions.
5. Include objects as resources of some behaviors.
6. Assert plausible frequency of different behaviors.
7. Provide open-source implementation of the programs.

Both our programs address these requirements.

3 Corpora generation

In this section we detail our two programs. The former uses HTN planner for generating ADL and the latter uses hierarchical reactive planner and also includes 3D simulation in the production loop. For example, concerning movement, the output of both programs is one action $MOVE(FROM, TO)$; however, the latter simulates walking the agent along the respective navigation graph step by step, which may result in altering the final destination.

We first describe the HTN approach and then the hierarchical reactive planning approach.

3.1 Planning approach

Our planning algorithm is based on a randomized version of the HTN planner SHOP2 [23] developed by [5]. Besides simple randomization of possible alternative goal decompositions introduced in [5] we extended the planning algorithm with probabilities of alternative decompositions used to fulfill different goals. Previous version [5] used only uniform distribution in selection of alternatives. We decided to use HTN because our previous work [21] comparing several PDDL planners shows that they suffered serious performance degradation issues when applied to a domain of our size. The source code of the modified SHOP2 planner explained here and a code of our domain is available online[1] under GNU GPL license.

[1] Homepage of our planning program is at http://code.google.com/p/epis-planner/ [16.9.2011]

Technical details The SHOP2 planner does not operate directly with time and generally outputs a satisfactory plan. Thus to meet Req. 1 we create a hierarchy of tasks starting with one root task *DAY* that is decomposed to tasks corresponding to various parts of the day: *MORNING, BEFORE-NOON, NOON, AFTER-NOON, EVENING and NIGHT*. These tasks are not part of the real stream of an agent's actions. They serve as a tool that helps to simulate different parts of the day and ensure time flow continuity. Each task representing a real activity of an agent is then associated with some time interval, which ensures that tasks are performed in corresponding time.

Req. 2 on nondeterminism was already addressed by the previous work [5] that we are building upon. In [5] all alternative task decompositions have the same probability of being chosen, that is the probability of choosing decomposition T of parent Pa is: $P(T|Pa) = \frac{1}{|children(Pa)|}$, where $children(Pa)$ denotes a set of all applicable decompositions. We extended this simple schema and introduced *weight of the decomposition* $W_T \in (0, \infty\rangle$. Now the probability of selecting decomposition T is given by:

$$P(T|Pa) = \frac{W_T}{\sum_{t \in children(Pa)} W_t}$$

This way we can introduce some rare events like illness etc.

Req. 3 and 5 are naturally addressed by the SHOP planner. Hierarchical decomposition is an innate property of the HTN formalism and objects can be easily incorporated into the plan as parameters of actions in a STRIPS like notation.

High number of alternatives (Req. 4) for each goal is fulfilled by the domain, e.g. in our domain goal *satisfy hunger* has three different decompositions that have further variants. At the end there are 15 different ways the goal can be accomplished. Similarly, Req. 6 is addressed by the domain. More detailed description of the domain is available on the program's web page.

Example Code Here is an example of definition of our domain.

Listing 1.1: Example of method definition

```
(:method :weight 2 (ASatisfyHunger)
    eat-outside
    ((hungry)
     (affordance ?place to-eat)
     (affordance ?place menu ?food))
    ((GetTo ?place)
     (!order ?food) (!eat ?food)
     (!drink))
)
```

Listing 1.2: Example of operator definition

```
(:operator (!eat ?food)
    () ((hungry)) ()
)
```

Listing 1.1 shows an HTN method *ASatisfyHunger* defining one possible decomposition of satisfying hunger. Each definition of a method is equivalent to an OR-node in the AND-OR tree formalism. Each child in the decomposition is equivalent to an AND-node. In this case, we describe a way how to satisfy a hunger by finding a place where we can eat, traveling there, picking a random item from a menu and eating it. Operators are elements that can have a direct influence on a state of the world. The operator *!eat* defined in Listing 1.2 removes the fact *hungry* from the current world state. Listing 1.3 shows an example of output of our program: a stream of actions corresponding to a visit of the pub *PUB1.* Some of the actions are parameterized with objects or places needed for their proper execution.

Our domain consists of 56 operators, 8 top level methods, 19 methods in total plus another 16 auxiliary methods used to capture different parts of the day and finally 81 atoms representing places and objects in the world.

Listing 1.3: Example of generated stream of actions

```
GET-TO-TRAMSTOP STATION_WORK
WAIT-FOR-TRAM TRAM22
BOARD-TRAM TRAM22
RIDETRAM TRAM22 STATION_WORK STATION_PUB1
EXIT-TRAM TRAM22
GET-TO STATION_PUB1 PUB1
ENTER PUB1
DANCE
ORDER-BEER
DRINK
CHAT
LEAVE PUB1
GET-TO PUB1 STATION_PUB1
WAIT-FOR-TRAM TRAM14
BOARD-TRAM TRAM14
```

3.2 3D simulation approach

Our second program using hierarchical reactive planning produces a flat preliminary stream of ADL, which is then executed in a 3D virtual environment in an accelerated way. This helps with dealing with visibility of objects, action failures etc. An agent's decision making is based on a concept of AND-OR trees, see [8]. The scheduling algorithm chooses a top-level goal that will be executed by the agent and execution of this goal is then driven by the AND-OR tree representing possible behavior to accomplish the top-level goal. The chosen tree is then executed in the virtual environment. During the actual simulation the agent also collects objects required to perform those actions. The final output is a hierarchical ADL.

Technical details Requirements 1 and 2 on regularity and nondeterminism are addressed by a scheduling mechanism used to create the agent's schedule for each day. The schedule defines which goals can be executed at a particular simulation time.

To define the schedule we will introduce following terms:

1. G is a set of all top-level goals that can be performed by the agent. G_0 is this set plus *no_goal* element, that is $G_0 = G \cup \{no\_goal\}$.
2. Transition λ is a triplet (g_1, g_2, T) where $g_1 \in G_0, g_2 \in G$ and T is a time interval when this transition can be applied.
3. Schedule S is a set of transitions λ and a function p assigning each transition a probability of being applied $p(\lambda)$. A valid schedule must fulfill: $\forall t \forall g_1 \in G_0$: $\sum_{\lambda \in S, \lambda=(g_1,g_2,T), t \in T} p(\lambda) = 1$, where t denotes simulation time.

During the simulation the agent performs top-level goals from the schedule. When the execution of a top-level goal is finished, all transitions applicable to the current simulation time and the previous top-level goal are found and one of them is chosen according to their probabilities. In some cases (e.g. start of simulation, no applicable transition is found) the agent may not follow any goal. In this case the next goal will be chosen from the transitions that are applicable to the current simulation time where $g_1 = no\_goal$. For any time such transition has to be defined in a valid schedule.

Scheduling top-level goals is done with granularity of 1 hour, meaning the boundaries of a time interval in a transition will always be whole hours. Beside hours, day of week can be defined for each time interval, e.g. *[08:00, 11:00] on Monday.* It is also possible to define general intervals valid for any day of a week or change the granularity of 1 hour.

In contrast with the HTN approach where the generated plan was immediately outputted as an activity stream here the schedule serves only as an "advice" for the simulated agent, some goals in the schedule can be for example interrupted during the simulation by external forces. This also adds to the nondeterminism of the simulation.

Req. 3 is met by using the AND-OR tree formalism. For each atomic action executed in the environment we have a trace of possibly several higher level goals, e.g. $Work \rightarrow Commute \rightarrow Walk$.

High number of alternative decompositions for higher level goals (Req. 4) is given by design of the domain. For our simulation we have created a schedule representing top-level goals of a single worker living in a city. He is following a general plan based on a working week; he sleeps each night, goes to work on week days, enjoys free time during evenings and weekends. Our agent can follow

27 different top-level goals. These range from basic goals like sleeping to more complex ones like going to work, drinking in a pub or doing physical activities.

Complex goals are defined by more complex AND-OR trees. Therefore every execution of the top-level goal can generate a different stream of atomic actions. Currently the most complex goal defined in our program is the top-level goal *to work*. Courses of executions of this goal can differ in details like going to work by foot or taking the public transportation, or in more important points like working in the office building or working from home. Altogether our *to work* goal can be completed by 160 unique sequences of atomic actions.

Of course not all available top-level goals offer such variability and there are several goals, like sleeping, that offer no variability at all. Altogether more than half of our 27 different top-level goals offer at least 10 different ways how the goal can be achieved.

As in the HTN approach, objects or places are needed as preconditions for performing some actions (Req. 5). For example, in order to buy some groceries, the agent needs to localize a place with type *supermarket* and needs to acquire an object of type *money*.

Defining different frequencies for different behaviors (Req. 6) is enabled due to the scheduling mechanism described above. Frequency can range from several executions a day to less than one execution in a week.

Virtual environment Actual simulation is run in Pogamut platform [11] on Unreal Tournament 2004[2] engine. A specific map of a small town that contains all the objects and places necessary to follow an agent's plan is used. Different buildings in the map represent his home, work, shops, restaurants, etc. It is possible to watch an agent as he moves in the virtual world. The actual atomic actions, except for walking and running, are not literally executed by the agent; he simply runs to the adequate place on a map and sends a message about execution of an atomic action without performing the low level animation (Fig. 1).

4 Behavior statistics and complexity

Requirements 1, 2 and 4 from Section 2 call for generating streams of actions that will be similar to activity of humans. Note we are not discussing here believability of animations, gestures or facial expressions of agents, which are studied in the field of virtual agents often, but believability of long streams of actions as defined in Introduction. It is unclear how to measure this quality, to our knowledge the field is lacking some widely accepted measure of complexity and believability of agent's behavior. Some initial steps in this direction have already been done. Arrabales et al. [2] provide scale for a measurement of single agent's complexity but it depends on subjective human assessment of the agent's behavior and its decision making algorithm. Alternative to this approach is [19] that proposes

[2] Epic Games, 2004,
http://en.wikipedia.org/wiki/Unreal_Tournament_2004 [19.9.2011]

Fig. 1: Screenshot of a simulation showing an agent in a virtual kitchen.

objective measures like *size of a procedural knowledge* but these measures are not always related to believability of the observed behavior.

We have to ask what properties of an activity stream can contribute to its believability. Is a sequence of actions *raise hammer, hammer in nail* and so on repeatedly believable? In some contexts it could be but not for a long time. We would say that this behavior is too predictable and lacks deviations that are inherent to behavior of any creature, be it a human or an animal. To reveal these highly predictable behaviors we take two sources of inspiration from the information theory. We can model the activity stream as a Markov source and we can measure its entropy. Similar technique that tries to measure the amount of predictability of the sequence is compression ratio. Finally properties of the transition matrix can be also inspected.

Eventually to compare results obtained from our artificial domains to reality we use the same measures on several dataset obtained in real settings. Following sections detail the measures and datasets used and results of our analysis.

4.1 Measures used

Behavior entropy Entropy rate can be used as a measure of unpredictability of an agent's actions. Because the agent's behavior often includes dependencies between executed sub-behaviors we can model it as a Markov source. Under Markov assumption we can measure entropy of the source and use it as one descriptor of the behavior. Assuming the action source is 0, 1st and 2nd order Markov source the calculation of respective entropy rates H^0, H^1 and H^2 is given by the following equations:

$$H^0(S) = -\sum_i p_i log_b p_i$$
$$H^1(S) = -\sum_i p_i \sum_j p_{i,j} log_b p_{i,j}$$
$$H^2(S) = -\sum_i p_i \sum_j p_{i,j} \sum_k p_{i,j,k} log_b p_{i,j,k}$$

where p_i, $p_{i,j}$, $p_{i,j,k}$, respectively, denotes probability of action A_i, sequences of actions $A_i A_j$ and $A_i A_j A_k$, respectively, and b is a number of actions present in the stream. We can define entropies of higher orders analogically.

To get a grasp of the quantity measured by entropies of different orders see Tab. 1. Sequence $AAAA$ that can correspond to sequence *go to work, go to work ...* is entirely predictable thus all entropy rates are 0. On the other hand sequence $ABAB$ is unpredictable if we consider it as an unordered set of symbols, both A and B are equally likely, that is $H^0 = 1$. However if we view it as a first order Markov sequence it becomes perfectly regular, A is always followed by B and vice versa. Knowledge of one last symbol is sufficient to predict the next one, thus $H^1 = 0$. The example of sequence $AABB$ takes this one step further. We have to take into account two last symbols to reveal regularity of the sequence.

Sequence	H^0	H^1	H^2
AAAA	0	0	0
ABAB	1	0	0
AABB	1	1	0

Table 1: Example of entropy rates for several example sequences.

Notice that we use *log* with base b instead of 2. This is because different domains can use different number of atomic actions. Using log_b normalizes the entropy to the interval $\langle 0, 1 \rangle$ and thus makes it possible, in theory, to compare the values across different domains. The log_b normalization maintains the property that a completely deterministic sequence will have entropy 0, whereas a sequence with uniformly distributed probabilities of all actions will have entropy 1. In practice, we think it is meaningful to compare domains with a similar number of actions, but we do not claim that comparison between domains with 2 actions and 100 actions will make sense.

Compressibility Compression ratio of a sequence can be used as another measure of its complexity. Examples of using compression algorithms to measure sequence complexity can be found in genetic optimization [12] or in ethology [25].

Compressibility is connected to a more general framework of the minimum description length (MDL) [22] principle. The idealized version of the MDL searches for the shortest program written in a Touring complete programming language that can generate a given sequence. The length of the program can then be taken as a measure of complexity of the sequence. Unfortunately this definition is rather impractical since Touring complete languages are too expressive and MDL defined in this way is uncomputable. However when choosing simpler languages than the Touring complete ones we can compute shorter descriptions of the data and that is what compression algorithms do. Thus we can use compression algorithm of our choice and apply it to the sequence of atomic actions produced by the agent.

Markov process perspective Besides measuring unpredictability of the sequence we can also inspect distribution of frequency of the actions and properties of the transition matrix like distribution of *in degrees* of the actions.

4.2 Analysis methodology

Datasets We selected several sources of human activity data to compare their complexity with our activity streams generated by the programs. The datasets come from various sources, PLCouple1 [14], Huyhan [13], Kadlec [17] originate in ubiquitous computing, Restaurant game [24] is an online computer game, Behan-HTN and Cermak-3D datasets are our own generated datasets, Behan-HTN dataset was generated by the HTN planning (Sec. 3.1), Cermak-3D dataset by the 3D simulation (Sec. 3.2).

Even though some of those datasets contain additional sensory information (e.g. accelerometer data or GPS locations in ubiquitous computing datasets) for purposes of behavior complexity measure we are using only the activity annotation. We also merge subsequent executions of a same action into one. We now detail settings where the datasets were collected.

PLCouple1 dataset [14][3] was recorded in a smart home setting inhabited by two people, the dataset even includes video recordings. 100 hours of the activity of one inhabitant was then manually annotated.

Huyhn dataset [13][4] was recorded mostly in home and office setting, a participant was equipped with three 3D accelerometers whose readings were later used for activity recognition.

Kadlec dataset [17] comes from a project that tries to create an automatic diary from data collected by a smart phone. The dataset was collected during a training phase of the system where the user had to hierarchically annotate his activity. The dataset denoted *LL-Kadlec* is the set with the whole action hierarchy (e.g. one action could be $Work \rightarrow Commute \rightarrow Walk$) and *LLA-Kadlec* contains only the atomic actions (action from the previous example will be reduced to $Walk$ only).

Restaurant game dataset [24] consists of behavior logs recorded in an online 3D game, where players played a waitress or a customer. The original aim of the work is to create a library of behaviors from these logs that can be used by

[3] http://architecture.mit.edu/house_n/data/PlaceLab/PlaceLab.htm [5.7.2011]

[4] http://www.ess.tu-darmstadt.de/datasets/tud-ubicomp08 [26.8.2011]

autonomous agents playing the game. We use only the customer's actions and denote this dataset as *CUSTOMER*.

Besides these datasets selected for our analysis there are also other similar datasets, e.g. [31, 28, 18]. We could not use them, because [31] provides 245 annotated activities only and [18] lacks the activity annotation. The dataset [28][5] is large enough, but it was released only a few weeks ago. Tab. 2 compares properties of datasets used in our analysis.

As a baseline we also included two artificial datasets. One denoted as *det* consists of completely deterministic sequences (4 types of sequences that are repetitions of patterns AB, AABB, ABCABC and AABAAC), this simulates unnaturally long repetitive sequences of actions, e.g. mechanical *hammering in nails* over a very long period. The second denoted as *rAB* consists of random uniformly sampled sequences of 2 up to 5 symbols, this can be interpreted as the other extreme of the spectrum, activity driven by flipping coins.

Dataset	Type	Size	Act. types
PLCouple1 [14]	R	900	44
Huyhn [13]	R	462	34
LL-Kadlec [17]	R	5185	369
LLA-Kadlec [17]	R	5185	40
CUSTOMER [24]	V	3472*	13
Behan-HTN	S	∞	55
Cermak-3D	S	∞	62

Table 2: Overview of the activity datasets. Letters in the Type column means: R — data from real environments, manual annotation of human actions; V — human activity in a virtual environment; S — computer simulated agents in virtual environment. Size denotes number of actions in the dataset. *We use only a limited subset of the Restaurant Game dataset.

4.3 Dataset analysis

For analysis we used the R statistical software [26], graphs were created using the ggplot2 [32] package.

Entropy We measured entropy of all datasets using equations given in Section 4.1. Fig. 2 shows entropies of all the datasets. As can be seen the entropy of randomly uniformly sampled sequences is close to 1, all the other sequences including *det* has lower entropy, which is decreasing with increasing order of entropy. As we can see the *LLA-Kadlec* has higher entropy than the *LL-Kadlec*. This is because the latter dataset contains the whole hierarchy of actions, that is the atomic actions' context decreases the uncertainty. Our *Behan-HTN* has lower H^1 and H^2 entropies than others, suggesting a room for improvement.

[5] http://www.opportunity-project.eu/challengeDownload [4.10.2011]

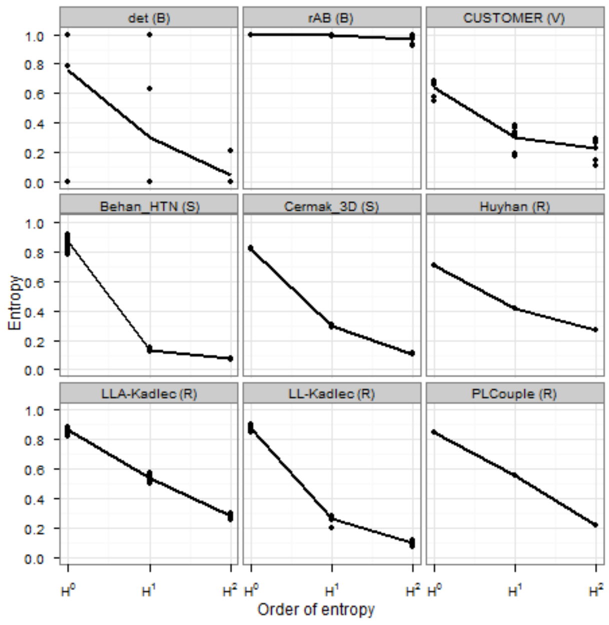

Fig. 2: Dependence of entropy on the number of past steps used for conditioning. Lines show means of the distributions. Letters V, S, R are explained in Tab. 2, B denotes baseline dataset.

Compressibility When we use compressibility ratio as a measure, the length of the sequence plays a role since longer sequences with some regularity tend to have better compression ratio than the shorter ones. Therefore we took the length of the shortest sequence (462 actions long Huyhan dataset) and splitted the other datasets into multiple sequences of this length. For each sequence we measured its compression ratio with several compression algorithms, namely: GZIP[6], Snappy[7], LZMA[8], LZF[9], QuickLZ[10] and bzip2[11]. Every action in a particular sequence was coded as a single letter (e.g. *Eat, Drink, Eat* was coded as *A, B, A*). Information about duration or objects involved was removed in this phase. Eventually we combined the measured compression ratios with entropies H^0, H^1, H^2 and also H^3 and performed a principal component analysis [16] (PCA) on these observations.

Fig. 3 shows the projection of the observations. To better understand the projection Tab. 3 shows directions of the first two principal components, PC1 and PC2. PC1 depends strongly on the compression ratios, the best compressible sequences are on the left on Fig. 3, sequences with worse compression ratios are

[6] http://www.gzip.org/ [17.9.2010]
[7] http://code.google.com/p/snappy/ [17.9.2010]
[8] http://www.7-zip.org/ [17.9.2010]
[9] https://github.com/ning/compress/ [17.9.2010]
[10] http://www.quicklz.com [17.9.2010]
[11] http://bzip.org/ [17.9.2010]

on the right, whereas PC2 depends more on the entropy rates, high entropy sequences are on the bottom, lower entropy sequences are at the top. PC1 accounts for 0.62% of variance, PC1 and PC2 together account for 0.86% of variance of the data. We can note that all compression algorithms contribute about the same to PC1, this is because of strong correlation of compression ratios (all pairwise correlations are > 0.94).

We can see that the datasets obtained in real settings (*PLCouple, LL-Kadlec, LLA-Kadlec, Huyhan*) all fall to similar cluster of sequences that are hard to compress but they have much lower entropy than random sequences. Datasets obtained by programs (*Behan-HTN*,*Cermak-3D*) are a bit easier to compress and Behan-HTN also has a bit lower entropy; there is room for improvement but still they are close to the real datasets. We also see that not only *rAB*, but also *det*, which scored similarly to natural datasets according to H^0, H^1 and H^2 alone, are substantially different.

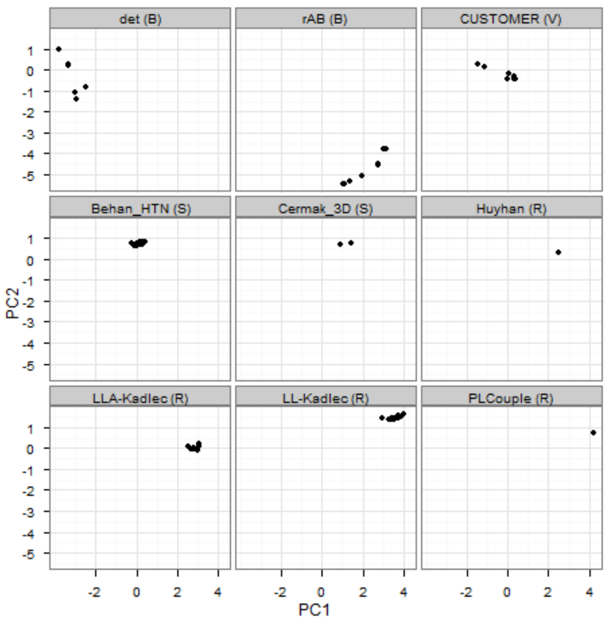

Fig. 3: PCA of behavioral datasets. Each point represents one sequence of the given type.

	H^0	H^1	H^2	H^3	BZIP2	GZIP	LZF	LZMA	QuickLZ	Snappy
PC1	0.10	0.29	0.20	0.04	0.35	0.37	0.39	0.39	0.39	0.39
PC2	-0.43	0.37	0.50	0.57	-0.24	-0.18	-0.00	-0.09	-0.02	0.02

Table 3: Vectors of the first two principal components.

Markov process perspective PCA brought some insight into structure of the datasets but we can further inspect the datasets from other perspectives. One natural is distribution of actions' frequencies. Fig. 4 shows that all the natural datasets and both our datasets, but not the baselines, follow similar exponential like distribution, having many rare actions and a few common actions.

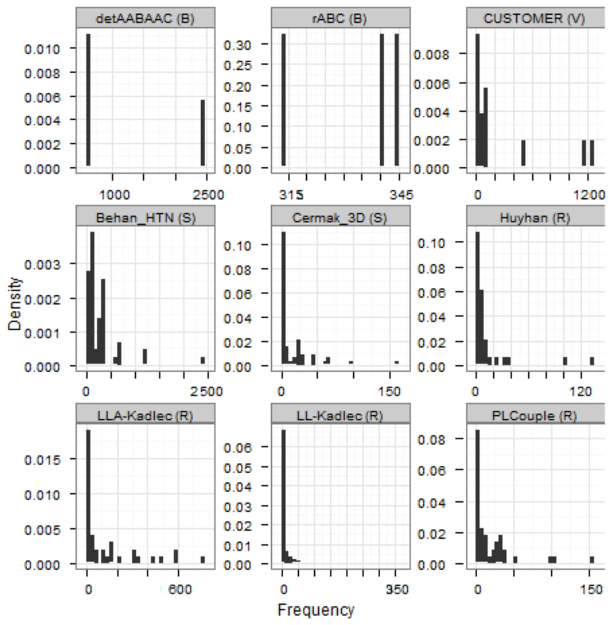

Fig. 4: Histograms of action frequencies. X axis is number of repetitions of the given action, Y axis shows density - proportion of actions with given frequency in the whole dataset.

Small variation of the previous approach is analysis of number of the action classes preceding a given class, that is *in degree* of a given action class in a transition graph corresponding to a first order Markov process learned from the dataset. Fig. 5 shows distributions of in degrees. We see that both our datasets and all real datasets with the exception of *CUSTOMER* dataset again follow similar exponential like distribution. *CUSTOMER* dataset has a lot of actions with many predecessors and fewer actions with only a few predecessors. This can be caused by the fact that players of the Restaurant game where this dataset originates from had to explore an environment of the game and all its possibilities to learn how the environment works. Thus they may try several combinations of actions that they would not do in real life; additional analysis would be needed.

5 Discussion and Future Work

We presented our two programs for generation of activities of daily living. Domain specifications, i.e. inputs of the programs, present an agent's life style.

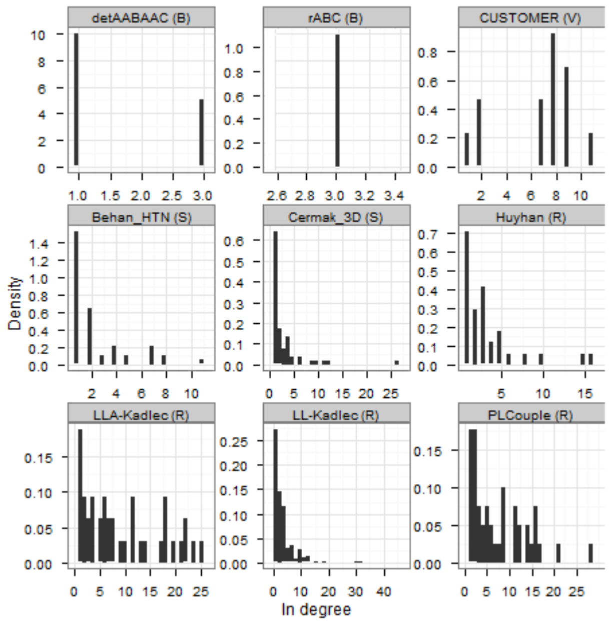

Fig. 5: Histograms of in degrees.

For every domain, we are able to generate different variants of activity streams, i.e. different courses of a particular life period; the approaches are partly non-deterministic. The generation of month long periods takes minutes of computation time. We will use the corpora for evaluating our new episodic memory model for virtual agents, which we develop in parallel.

To evaluate resemblance of the generated activity streams to streams of activities of real humans, we defined several measures of complexity of the sequences, namely conditioned entropy, compressibility and properties of transitions between actions. We compared our corpora to several other sources of activity obtained in real settings, demonstrating that our corpora resemble the real-world ones unlike two mechanistic baselines. Our corpora are still a bit more predictable than the real-world datasets since they could be compressed better and one of them has a bit lower entropy, suggesting room for improvements.

The measures *per se* are useful, because they cluster together datasets that a user would expect to share some commonalities yet reveal differences between datasets with different features.

One next step is refining the measures to show finer differences between the datasets as was the example of *CUSTOMER* dataset and its in-degree distribution. This can enhance the projections of datasets through PCA.

New measures can come from addressing limitations of the current approach: 1) we are not encoding use of objects explicitly, even though corpora are generated with objects; 2) there is no mechanism dealing with similarity of actions; 3) we are not taking into account duration of actions. In our opinion, a probabilistic graphical model [20] could address most of these issues. For purposes of episodic memory modeling we would also like to encode percepts of the agent since they can be also stored in the memory. This would make it possible to distinguish "empty" worlds from those populated with many changing objects and agents.

Another next step is improving the approaches to corpora generation. As shown above, we still "not there yet" in terms of resemblance to real datasets.

Our approach to ADL generation can be also used in the context of plan recognition, e.g. [6], or traffic simulation, e.g. [3]. Our measures could also be incorporated into development cycle of virtual agents and used along with the well established practice of unit testing. Besides unit tests whose result is fail/-success, we can record an agent's actions and compare them with previous runs of the same agent. Unexpected shifts in the measures possibly plotted using PCA can signal unwanted changes in the agent's behavior and hence probably also in its functionality. The measures could also be tested in different simulations of human behavior (e.g. [1]), or in completely different domains like maritime simulations [15].

The work that remains to be done is to find whether there is a correlation between some of our measures and human's rating of sequences on a scale from artificial to natural. E.g. it seems reasonable to require some incompressibility of the streams we do not know if this is really the important aspect of being "natural". The same question of connectedness to human's rating applies also to the other measures.

On a more general level one possible direction of a future work could be to characterize mathematically a minimal model needed to generate believable streams of actions, i.e. to find a model that is able to *fool* all measures comparing its actions to real datasets. We could see such model as an analogy to scale-free networks describing topology of World Wide Web [4], Brownian motion or recently criticized [10] Lévy flight concept [29]. Formulation of such model grounded in real data could also have implications for design of programming languages oriented on specification of virtual agents behavior like POSH [9] or AgentSpeak [27].

We do not claim that we have solved the problem of behavior complexity measure but we think that our approach combining entropy and compressibility can be used as a first step in exploratory analysis of a behavior expressed by an unknown agent. His behavior can be plotted against other known datasets and we can quickly get first impression of its complexity.

Acknowledgment

This work was partially supported by the student research grant GA UK 0449/2010/A-INF/MFF, by the SVV project number 263 314 and by the grants GACR 201/09/H057 and P103/10/1287.

References

1. Allbeck, J., Badler, N.: Simulating human activities for synthetic inputs to sensor systems. Distributed Video Sensor Networks pp. 193–205 (2011)

2. Arrabales, R., Ledezma, A., Sanchis, A.: ConsScale: A pragmatic scale for measuring the level of consciousness in artificial agents. Journal of Consciousness Studies 17(3-4), 131–164(34) (2010), `http://www.ingentaconnect.com/content/imp/jcs/2010/00000017/F0020003/art00008`
3. Balmer, M., Meister, K., Rieser, M., Nagel, K., Axhausen, K.: Agent-based simulation of travel demand: Structure and computational performance of MATSim-T. In: 2nd TRB Conference on Innovations in Travel Modeling, Portland (2008)
4. Barabási, A., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509 (1999)
5. Blaylock, N., Allen, J.: Generating artificial corpora for plan recognition. In: User Modeling 2005. pp. 179–188. Springer (2005)
6. Blaylock, N., Allen, J.: Fast hierarchical goal schema recognition. In: Proceedings of the National Conference on Artificial Intelligence. vol. 21, p. 796. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2006)
7. Brom, C., Lukavskỳ, J.: Towards virtual characters with a full episodic memory ii: The episodic memory strikes back. In: Proc. Empathic Agents, AAMAS workshop. pp. 1–9 (2009)
8. Brom, C., Pešková, K., Lukavskỳ, J.: What does your actor remember? Towards characters with a full episodic memory. Virtual Storytelling. Using Virtual Reality Technologies for Storytelling pp. 89–101 (2007)
9. Bryson, J., Stein, L.: Modularity and design in reactive intelligence. In: International Joint Conference on Artificial Intelligence. vol. 17, pp. 1115–1120 (2001)
10. Edwards, A., Phillips, R., Watkins, N., Freeman, M., Murphy, E., Afanasyev, V., Buldyrev, S., da Luz, M., Raposo, E., Stanley, H., et al.: Revisiting Lévy flight search patterns of wandering albatrosses, bumblebees and deer. Nature 449(7165), 1044–1048 (2007)
11. Gemrot, J., Kadlec, R., Bída, M., Burkert, O., Píbil, R., Havlíček, J., Zemčák, L., Šimlovič, J., Vansa, R., Štolba, M., et al.: Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. In: Agents for Games and Simulations, LNCS vol. 5920. pp. 1–15. Springer (2009)
12. Gomez, F., Togelius, J., Schmidhuber, J.: Measuring and optimizing behavioral complexity. In: Proc. Intl. Conference on Artificial Neural Networks (ICANN-09), Lamissol, Cyprus. pp. 765–774 (2009)
13. Huynh, T., Fritz, M., Schiele, B.: Discovery of activity patterns using topic models. In: Proceedings of the 10th international conference on Ubiquitous computing. pp. 10–19. ACM (2008)
14. Intille, S., Larson, K., Tapia, E., Beaudin, J., Kaushik, P., Nawyn, J., Rockinson, R.: Using a live-in laboratory for ubiquitous computing research. In: Pervasive Computing. pp. 349–365. Springer (2006)
15. Jakob, M., Vanek, O., Hrstka, O., Pechoucek, M.: Agents vs. pirates: Multi-agent simulation and optimization to fight maritime piracy. In: Proceedings of AAMAS (2012), in press
16. Jolliffe, I.: Principal component analysis. Springer (2002)
17. Kadlec, R., Brom, C.: Towards an automatic diary: an activity recognition from a data collected by a mobile phone. In: Workshop proceedings Space, Time and Ambient Intelligence, IJCAI. pp. 56–60 (2011), `http://dl.dropbox.com/u/17077973/proceedings/STAMI-20110/STAMI-11-IJCAI-Proceeding.pdf`
18. Kiukkonen, N., Blom, J., Dousse, O., Gatica-Perez, D., Laurila, J.: Towards rich mobile phone datasets: Lausanne data collection campaign. In: Proceedings of the ACM International Conference on Pervasive Services (ICPS) (2010)

19. Klügl, F.: Measuring complexity of multi-agent simulations–an attempt using metrics. Languages, Methodologies and Development Tools for Multi-Agent Systems pp. 123–138 (2008)
20. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. The MIT Press (2009)
21. Kučerová, L., Brom, C., Kadlec, R.: Towards planning the history of a virtual agent. ICAPS Workshop on Planning in Games (2010)
22. Li, M., Vitanyi, P.: An introduction to Kolmogorov complexity and its applications. Springer-Verlag New York Inc (2008)
23. Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J., Wu, D., Yaman, F.: SHOP2: An HTN planning system. Journal of Artificial Intelligence Research 20(1), 379–404 (2003)
24. Orkin, J., Roy, D.: The restaurant game: Learning social behavior and language from thousands of players online. Journal of Game Development 3(1), 39–60 (2007)
25. Panteleeva, S., Danzanov, Z., Reznikova, Z.: Estimate of complexity of behavioral patterns in ants: analysis of hunting behavior in myrmica rubra (hymenoptera, formicidae) as an example. Entomological Review 91(2), 221–230 (2011)
26. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2010), `http://www.R-project.org/`
27. Rao, A.: AgentSpeak(L): BDI agents speak out in a logical computable language. Agents Breaking Away pp. 42–55 (1996)
28. Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Forster, K., Troster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al.: Collecting complex activity datasets in highly rich networked sensor environments. In: Seventh International Conference on Networked Sensing Systems (INSS). pp. 233–240. IEEE (2010)
29. Shlesinger, M., Zaslavsky, G., Frisch, U.: Lévy flights and related topics in physics. Springer (1995)
30. Tulving, E.: Precis of elements of episodic memory. Behavioral and Brain Sciences 7(2), 223–68 (1984)
31. Van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. In: Proceedings of the 10th international conference on Ubiquitous computing. pp. 1–9. ACM (2008)
32. Wickham, H.: ggplot2: elegant graphics for data analysis. Springer New York (2009), `http://had.co.nz/ggplot2/book`