Charles University in Prague

Faculty of Mathematics and Physics

DOCTORAL THESIS

Rudolf Kadlec

DyBaNeM: Bayesian Model of Episodic Memory

Department of Software and Computer Science Education

Supervisor of the doctoral thesis:	Mgr. Cyril Brom, Ph.D.
Study programme:	Informatics
Specialization:	Theoretical Computer Science

Prague 2015

I would like to thank my supervisor Cyril Brom for his guidance and my colleagues Klára Pešková, Ondřej Burkert, Michal Čermák and Zdeněk Běhan for their work that contributed to episodic memory research in our research group.

I would also like to thank my family who has always supported me in this journey.

Název práce: DyBaNeM: Bayesovský model epizodické paměti

Autor: Mgr. Rudolf Kadlec

E-mail: rudolf.kadlec@gmail.com

Katedra: Kabinet software a výuky informatiky

Vedoucí disertační práce: Mgr. Cyril Brom, Ph.D. Kabinet software a výuky informatiky

Abstrakt:

Umělí agenti vybavení epizodickou (nebo autobiografickou) pamětí mají schopnost zapamatovat si a následně si i vybavit, co se jim stalo v minulosti. Stávající modely epizodické paměti (EP) fungují jako pouhé logy s indexy: umožňují záznam, vyhledávání a mazání vzpomínek, ale jen zřídka uchovávají agentovu aktivitu v hierarchické podobě, natož aby umožňovaly automaticky abstrahovat pozorovanou aktivitu do obecnějších epizod. V důsledku toho nejzajímavější rysy lidské EP, jako jsou rekonstrukce vzpomínek, vznik falešných vzpomínek, postupné zapomínání a předpovídání překvapivých situací, zůstávají mimo jejich dosah. V této práci představíme výpočetní model epizodické paměti pojmenovaný DyBaNeM. DyBaNeM propojuje modelování EP s algoritmy pro rozpoznávání aktivit v jednom výpočetním modelu. DyBaNeM staví na principech Bayesovské statistiky a na takzvané Fuzzy-Trace teorii vycházející z oblasti výzkumu falešných vzpomínek. V práci bude představeno několik verzí modelu ByDaNeM s různými pravděpodobnostními modely realizovanými pomocí dynamických Bayesovských sítí. Následně popíšeme fáze kódování, uložení a vybavení vzpomínek, tak jak jsou implementovány v modelu DyBaNeM. Všechny tyto fáze budou demonstrovány na jednoduché ukázkové doméně, kde také porovnáme rozdíly mezi jednotlivými variantami modelu. Poté otestujeme chování modelu DyBaNeM na dvou realističtějších doménách relevantních pro inteligentní virtuální agenty.

Klíčová slova: Epizodická paměť, inteligentní virtuální agenti, rozpoznávání aktivity

Title: DyBaNeM: Bayesian Model of Episodic Memory

Author: Mgr. Rudolf Kadlec

E-mail: rudolf.kadlec@gmail.com

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Cyril Brom, Ph.D. Department of Software and Computer Science Education

Abstract:

Artificial agents endowed with episodic (or autobiographic) memory systems have the abilities to remember and recall what happened to them in the past. The existing Episodic Memory (EM) models work as mere data-logs with indexes: they enable record, retrieval and delete operations, but rarely organize events in a hierarchical fashion, let alone abstract automatically detailed streams of "what has just happened" to a "gist of the episode." Consequently, the most interesting features of human EM, reconstructive memory retrieval, emergence of false memory phenomena, gradual forgetting and predicting surprising situations are out of their reach. In this work we introduce a computational framework for episodic memory modeling called DyBaNeM. DyBaNeM connects episodic memory abilities and activity recognition algorithms and unites these two computer science themes in one framework. This framework can be conceived as a general architecture of episodic memory systems, it capitalizes on Bayesian statistics and, from the psychological standpoint, builds upon the so-called Fuzzy-Trace Theory (FTT) stemming from the false memory research field. Several variants of ByDaNeM with different probabilistic models realized as Dynamic Bayesian Networks (DBNs) will be defined. Subsequently all stages of DyBaNeM's encoding-storage-retrieval cycle will be demonstrated on a simple toy domain. Differences between DyBaNeM's variants will be compared on this example side by side. Then proof of concept connection of DyBaNeM to two more realistic domains relevant to intelligent virtual agents (IVAs) will be shown.

Keywords: Episodic Memory, Intelligent Virtual Agents, Activity Recognition

Contents

Chapter 1

Introduction

1.1 Motivation

The Episodic Memory (EM) (Tulving, 1972, 1983) is a memory for sequences of personal events, it can be briefly described as a memory for sequences of "what, where, when" events. We use EM every time we want to tell someone what we were doing last summer.

The main goal of this thesis will be to create an EM model for intelligent virtual agents (IVAs). The IVAs are a subtype of autonomous software agents (Wooldridge, 2000) embodied in a virtual body acting in 2D or 3D environments. An example of IVA can be a computer controlled character from a computer game or a chat agent on a web page.

One of the key preconditions for EM is the ability to "make sense" of the activity of observed agents/humans. This phase is often called *activity recognition*. Activity recognition is heavily studied by the artificial intelligence (AI) community, however, its connection to EM modeling has not been extensively studied yet.

Artificial agents endowed with episodic (or autobiographic) memory systems have the abilities to remember and recall what happened to them in the past. The number of agents with EM abilities has been increasing. At the same time, the existing models work as mere data-logs with indexes: they enable record, retrieval and delete operations, but rarely organize events in a hierarchical fashion, let alone abstract automatically detailed streams of "what has just happened" to a "gist of the episode."

Consequently, the most interesting features of human EM, reconstructive memory retrieval, emergence of false memory phenomena, gradual forgetting and predicting surprising situations are out of their reach. In this work we create a computational model of episodic memory that connects episodic memory abilities and activity recognition algorithms and unites these two computer science themes in one framework.

1.2 Scope

1.2.1 Taxonomy of Memory Systems

The human memory is not seen as a homogeneous system. It is believed that there are multiple types of memory for storing different types of information. One possible taxonomy of human memory systems is shown in Fig. 1.1. However, the exact taxonomy of memory systems is still debated, for in depth discussion see (Cowan, 2008).

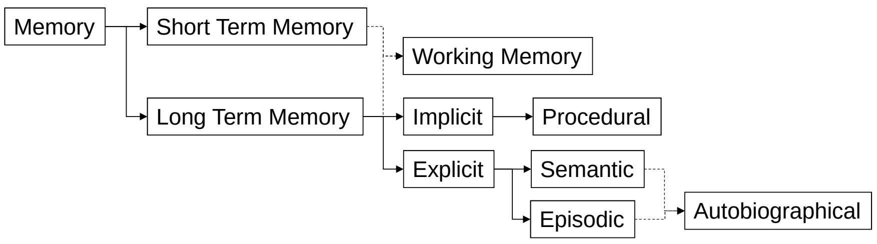


Figure 1.1: Possible classification of human memory systems. This taxonomy is based on (Cohen and Squire, 1980; Conway, 2001; Cowan, 2008). For detailed description see text.

In general the memory can be divided into a long term memory (LTM) and a short term memory (STM) (Atkinson and Shiffrin, 1968). In the LTM memories can be stored for up to a lifetime, even though some of them can be forgotten throughout that time. In the STM memories last for up to 30 seconds (Atkinson and Shiffrin, 1971) and it can hold only a limited number of items (Miller, 1956).

The LTM can be divided into an explicit and an implicit memory (Cohen and Squire, 1980). Explicit memory stores facts that can be consciously recalled and communicated to others. Implicit memory (Schacter, 1987) contains information that is not accessible consciously but it influences a person's behavior. A subtype of implicit memory is Procedural Memory (PM). This is memory for skills that can be performed by a person but one does not have to be able to describe the skill in detail. An example can be riding a bike or playing a piano.

The explicit memory consists of a Semantic memory (SM) and EM (Tulving, 1972, 1983). The SM is a store for general facts about nature of the world like "London is the capital of the UK; the Earth rotates around the Sun". As already discussed in the introduction, EM is memory for sequences of events. A subset of episodic and semantic memory that is related to the agent itself is called an autobiographical memory (Conway, 2001).

The EM is the type of memory that will be the focus of this thesis.

Another type of memory that is often mentioned in the literature is a Working memory (WM). The WM is the memory relevant for fulfilling tasks like summation of numbers or cooking. A detailed model of WM's internal structure is proposed by Baddeley (2000). The WM seems to be tightly linked with both STM and LTM (Cowan, 2008).

1.2.2 Computational Modeling of EM

Computational modeling of EM enjoys increasing interest in neuroscience, IVA, robotic research and E-memory communities. Now we will briefly review how these communities study EM.

Low level neuroscience simulations are usually focused on testing hypotheses concerning neuronal implementation of some functions of EM and sometimes even fitting real world data for isolated memory phenomena[1] (Bradski et al., 1994; Lisman, 1999; Howard and Kahana, 2002; Lehman and Malmberg, 2009; Reilly and Rudy, 2001).

On the other hand IVAs' and robotic communities try to equip agents and robots with EM to enhance their functionality, be it debriefing and summarization of educational scenarios (Rickel and Johnson, 1999; Dias et al., 2007), social skills (Dodd and Gutierrez, 2005; Burkert et al., 2010; Rabe and Wachsmuth, 2012; Mattar and Wachsmuth, 2012; Kasap et al., 2009; Lim et al., 2011; Kope et al., 2013), fulfilling delayed intentions (Li and Laird, 2011) or learning (Deutsch et al., 2008; Nuxoll and Laird, 2011; Subagdja et al., 2012) (possible uses in these domains are reviewed in (Brom and Lukavský, 2009)).

Another source of interest in EM modeling comes from applications of the so called E-memory revolution (Bell and Gemmell, 2009; O'Hara et al., 2006; Lazer et al., 2009). The E-memory revolution was enabled by the fact that more and more people are now "living in a network" (Lazer et al., 2009) meaning that traces of their behavior are logged by various services on a web or by mobile phones. We witness the rise of services logging events from our daily lives like Facebook or Google+. As the amount of stored data increases so will the need for better searching techniques over them. We can imagine feeding the data into personalized external EM that will later act as a memory prosthetic reminding us of similar events we have once experienced but probably forgot since then (Horvitz et al., 2004; Kamar and Horvitz, 2011).

[1]Note that there are tens of various memory related phenomena (see, e.g., (Wikipedia, 2015a)) and only several of them have been computationally modeled so far. Naturally, there are even fewer computational models accounting for several phenomena at once.

1.2.3 Processes in EM

EM models serve a variety of functions and they are implemented in diverse environments, however, virtually every model has to implement some common processes. Following Nuxoll's (Nuxoll, 2007, p. 39-41) terminology we can distinguish these three main processes and their subprocesses:

Encoding — what is stored and when; encoding includes: encoding initiation (when to start storing), episode determination (what consists an episode) and feature selection (what are the interesting features of an episode). When we think of EM models embodied in agents then this process may overlap with perception.

Storage — how the episodes are stored and what happens to them during storage; storage includes: defining episode structure and specifying episode dynamics like forgetting and memory consolidation. Part of this process is also sometimes called maintenance.

Retrieval — how the episodes are retrieved; retrieval includes: retrieval initiation, cue determination, best episode selection and partly missing information reconstruction.

An example of the processes is given in Fig. 1.2.

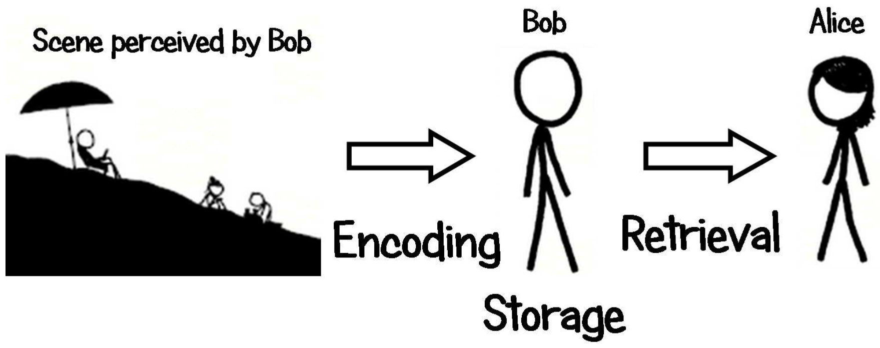


Figure 1.2: An example of processes included in every EM system. Imagine that Bob was once on a beach with his children. He encoded this episode and stored it for several years. When he wanted to re-tell the whole experience to Alice he had to retrieve the episode. This figure is based on graphics from xkcd.com.

1.2.4 Intelligent Virtual Agents

The proposed EM model is targeted mainly on IVAs. Thus here we will briefly define what IVAs are and what are their applications. We will define IVAs via manifest of the Intelligent Virtual Agents conference which states that:

> "Intelligent virtual agents (IVAs) are interactive characters that exhibit human-like qualities and communicate with humans or with each other using natural human modalities such as facial expressions, speech and gesture. They are capable of real-time perception, cognition and action that allows them to participate in dynamic social environments."[2]

Academic examples of IVAs are virtual museum guide Max (Kopp et al., 2005), tutoring agent Steve (Rickel and Johnson, 1999), agents from a serious anti-bullying game FearNot! (Aylett et al., 2005), agents in crowd simulations (Pelechano et al., 2008) or agents in a health domain (Bickmore et al., 2013).

Industrial applications may also feature IVAs. Virtually any non player character (NPC) from a computer game can be regarded as an IVA. This includes combat bots[3] from first person shooter (FPS) games[4] or characters inhabiting large virtual worlds in a massive multiplayer online role playing games (MMORPGs)[5]. As can be seen there is a variety of possible IVA applications, ranging from serious educational games or chat agents on web pages to purely entertaining video games.

The central feature of IVAs is their believability (Bates, 1994). It is important that the audience perceives the IVA's behavior as believable, even though the mechanism implementing the behavior might be relatively shallow. Specifically, it is not required that IVAs behave optimally in their environment, neither do humans. However, their behavior should be human-like, meaning that they should exhibit the same set of capabilities as humans. This includes modeling imperfection of humans. An illustration of this in the context of EM modeling might be that people perceive sometimes imperfect dating of memories (e.g., a week ago) as more human-like than precise date information (e.g., 6 days and 2 hours ago) (Brom et al., 2010). The requirement on enabling a set of human-like capabilities will shape the design of the proposed EM system.

[2]Quoted from description of the Intelligent Virtual Agents 2013 conference, URL: http://www.cstr.ed.ac.uk/iva2013/ [4.1.2014]

[3]Autonomous virtual agent in a computer game.

[4]URL: http://en.wikipedia.org/wiki/First-person_shooter [4.5.2014]

[5]URL: http://en.wikipedia.org/wiki/Mmorpg [4.5.2014]

1.3 Requirements on EM in IVAs

In this section the functions of an ideal EM model designed for use in IVAs will be specified. All these functions should be supported by the proposed EM model.

We illustrate possible use-cases that should be supported by the EM framework in the following example. Imagine a MMORPG inhabited by hundreds or even thousands of IVAs. Each IVA can be interviewed by a human player that may ask two basic types of questions:

a) "What were you doing yesterday?"

b) "What was player X doing yesterday?"

The first question asks about the IVA's recollection of its own actions, the second asks about the IVA's memory for actions of a human controlled avatar (or a different IVA). It is clear that the IVA has to be equipped with an EM model to answer both of these questions. However, the second question also requires the model to be able to *interpret* the players' (or IVAs') actions and infer his/her high level goals that are not directly observable in the environment. In addition, the model should be generic and applicable to different IVAs with minimal effort. Ideally there should be an out of the box algorithm for adjusting a possible model's parameters from logs of IVA's activity.

Additionally the EM framework should enable the following features in the dialog between the player and the IVA:

1. **Summarization:** The IVA should provide a high level summarization of an activity. For instance, when the player (P) asks: "What were you doing yesterday?", the IVA (I) equipped with our model should answer: "After getting up I went to work, in the afternoon, I visited my friends and then I returned home." instead of inadequately detailed "I got up, then I did my morning hygiene. I had breakfast, I got dressed and ..."

2. **Clarifying questions:** The player can ask further clarifying questions. E.g., P: "How did you get to your friends?"; I: "I walked there."

3. **Expressing a degree of certainty:** The IVA should express a degree of certainty for each recalled event. E.g., I: "Maybe I went there by car. I'm not sure."

4. **Believable mistakes in recall:** The IVA should make mistakes that are believable given the context. E.g., I: "I went to work by public transport." (Even though the IVA actually used a car.)

5. **Recall interesting details first:** The memory should weight interestingness of the events, thus it should highlight the most unusual memories. P: “What were you doing yesterday?”; I: “I saw a foreign army marching through the town, the rest of the day was as usual.”

It is important to note that in order to meet all these requirements pure log of observed events is not sufficient. We will need more complex data structures and algorithms to address this specification.

In order to perform a conversation between the player and the IVA sketched in the previous examples the EM model would have to be embedded in a broader agent architecture. A schema of a hypothetical complex agent equipped with our EM model is shown in Fig. 1.3. Even though the design and implementation of a complete dialog system (e.g., (Nooraei et al., 2014)) is not the focus of this thesis the proposed EM model is designed to enable all those features in a complex agent.

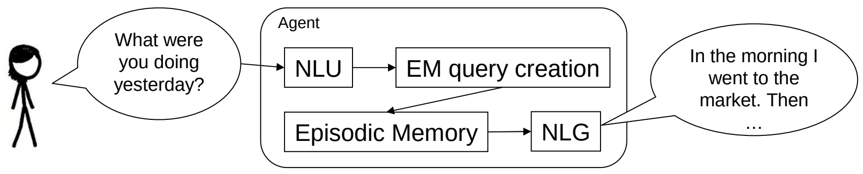

What were you doing yesterday?
Agent
NLU
EM query creation
Episodic Memory
NLG
In the morning I went to the market. Then …

Figure 1.3: An example of a complete agent architecture that is able to answer questions discussed in this section. This thesis is focused solely on the EM module. The other modules like natural language understanding (NLU) or natural language generation (NLG) are out of the scope of this thesis. This figure includes graphics from `xkcd.com`.

To our knowledge none of the current EM models supports all the previously mentioned features in one coherent computational framework (as reviewed in Sec. 2.3).

1.4 Design Implications

In this section we provide a brief discussion of design implications for our proposed EM model. In depth analysis of the model is in Chapter 3.

If the model should support only queries of the type “What were you doing yesterday?”, a plain log of events would be sufficient. However, the other requirements make the log based approach unusable. The requirement on high level summarization of observed events supposes the existence of a process that can deduce the summary of an activity from low level observations. This task is often called activity recognition. Amongst the diverse computational frameworks that can realize activity recognition (as reviewed in Sec. 2.5), hierarchical Bayesian methods are the most promising ones with respect to EM modeling. Bayesian methods can recognize hierarchical activity, which can be used to fulfill requirements 1 and 2. Further probabilistic representation that is natural to Bayesian methods can be used to implement requirements 3 and 5. Believable mistakes that are sensitive to the context of the stored episode (req. 4) can be a side effect of the recall process. Only the events persisting in the EM would be used to condition the probabilistic model and the forgotten events would be, possibly imprecisely, reconstructed as a result of Bayesian inference. This process will be called reconstructive recall.

All these features make Bayesian graphical models (Koller and Friedman, 2009) a natural choice as a computational basis of the proposed EM model. More specifically Dynamic Bayesian Networks (DBNs) (Murphy, 2002) will be used as the probabilistic model in our proposed EM model. DBNs are a type of Bayesian network that use parameter tying to compactly represent dynamics of temporal processes, such as IVA's activity. Another advantage of using DBNs is that there is a large body of prior work that applies these types of models to activity recognition (as detailed later in the text in Section 2.5). Moreover, just like Prolog programs, Bayesian computation can be used in both "directions". DBN can both infer high level episodes from low level observations and at the same time, generate sequences of low level observations from high level memories. This reversibility will be used in several stages of the proposed EM model's computation.

Our proposed model is also inspired by two psychological theories that account for several EM related phenomena. These theories are the Fuzzy-Trace Theory (FTT) (Brainerd and Reyna, 2005; Gallo, 2006) and the Event Segmentation Theory (EST) (Zacks et al., 2007). The FTT hypothesizes two parallel mechanisms that encode incoming information: *verbatim* and *gist*. Verbatim encodes the surface-form of the information in detail, gist encodes the meaning in a coarse-grained way (Gallo, 2006). An example of verbatim might be "I went by an unusual bus with red stripes." and the gist of the whole experience might be "Workday commuting." The FTT was previously implemented in the setting of static scenes (Hemmer and Steyvers, 2009). Present work extends FTT's general approach also to sequences of events. The EST assumes existence of a process that continuously segments observed events into meaningful sequences that we call episodes. Hierarchical organization of memories in our proposed models is also supported by evidence that people tend to perceive episodes in a hierarchical fashion (Zacks and Tversky, 2001; Bond, 2005).

1.5 Goals of the Thesis and the Main Contribution

The goal of this thesis is to design an EM framework that addresses all the requirements from Section 1.3. Further the proposed EM framework will be implemented in a prototype and tested on datasets resembling streams of actions generated by an IVA. The thesis is focused solely on the EM framework, creating a full IVA equipped with the EM is out of the scope of this thesis.

The main contribution of this thesis is the introduction of a new framework for EM modeling called DyBaNeM. DyBaNeM unites disciplines of plan recognition, EM modeling and compression algorithms. Besides describing the theoretical framework we will also use it to create a generic EM model for storing sequences of events. The framework is inspired by psychological findings but does not try to explicitly replicate in silico memory related phenomena like the neuropsychological models.

The additional contribution of this work is an in depth review of EM models relevant to IVAs.

1.6 Structure of the Thesis

In Chapter 2 we will discuss work related to our proposed EM model. We will focus on previous computational EM models, algorithms used for activity recognition and we will also discuss popular techniques for programming IVA's behavior. In Chapter 3 we will analyze how to best meet the requirements presented in the introduction. In Chapter 4 we will thoroughly introduce our EM framework called DyBaNeM. After that in Chapter 5 we will describe how the framework works on a set of toy examples that are easy to interpret. In Chapter 6 the framework will be tested on two more complex domains. Next in Chapter 7 we will provide high level discussion of the proposed EM model. In the end we summarize possible future directions of research connected to DyBaNeM and we finish the thesis with conclusion.

Chapter 2

Related Work

A major part of this chapter is dedicated to a review of current EM models. First, we will list prior reviews that are relevant to EM modeling. Second, we introduce a taxonomy of EM models (Sec. 2.1). Then we define features that interest us in the reviewed models (Sec. 2.2). After that we will describe two selected groups of models that are the most relevant for our work (Sec. 2.3 and 2.4). Then current approaches to activity recognition will be reviewed (Sec. 2.5) since it will be an important part of our proposed EM model. In the end we will briefly survey popular formalism for programming IVA's behavior (Sec. 2.6).

To our knowledge there are only several reviews focusing on EM systems in the context of AI. For example, Wood et al. (2011) surveys both current neuro-psychological understanding of different memory systems in humans that goes up to the neuronal level and also computer simulations of these systems. It discusses both WM, SM and EM. Compared to Wood et al. (2011) our review focuses only on computational models of EM but this enables us to provide greater detail of discussed models. Another rather limited review is given in (Nuxoll, 2007, p. 17-22). It discusses only several models and it does not provide the same level of insight as our review. Possible benefits of equipping IVA with EM are discussed in (Brom and Lukavský, 2009), while (Holland and Marques, 2010) discusses increasingly complex agent architectures and justifies the existence of an EM from an engineering point of view. Another recent review (Lim, 2012) discusses EM models in the context of intelligent social companions (ISCs) a subtype of IVAs. ISCs are agents designed to be companions to humans, thus theirs' memory is a tool that should make the interaction as natural as possible.

Our review focuses specifically on computational models of EM independent of their application, unlike the review of Lim (2012). We try to provide insight in their internal working and not just describe theirs functions. We also categorize models based on their level of description and intended use. We will focus on EM functions connected to storing sequences of events. That is we will deliberately omit models inspecting other modalities of EM, for instance link of spatial and episodic memory studied in (Byrne et al., 2007; Brom et al., 2012). The other memory systems are discussed only when it is needed to describe some function of EM. We limit depth of neuro-psychological evidence for various concepts connected to the EM provided in this review since there are other more competent works in this field, e.g., (Burgess et al., 2002; Bird and Burgess, 2008), and we concentrate on the computational models. For a review of computational neuro-psychological models of EM see (Norman et al., 2008).

2.1 Episodic Memory Model Types

EM models are often described on different levels of abstraction and with different aims in mind. To categorize these types of models we utilize Marr's tri-level hypothesis (Marr, 1982; Dawson, 1998). This hypothesis was originally formulated in the context of visual perception research. However, it can be used for any cognitive modeling task, hence for episodic memory as well. The hypothesis posits that every cognitive system can be described on these three three levels of description. These are (paraphrasing (Marr, 1982)):

Computational — this level describes what the goal of the system is, why it is needed and what the general strategy is.

Algorithmic/representational — this level describes the algorithm realizing functions specified on the computational level, what representations are used for input and output data.

Implementational — this level describes how the algorithm is physically realized, e.g., what is the neuronal circuit implementing it.

We can more or less identify these levels of abstraction with language that is used for description of hypotheses and models on these levels. Models on the computational level are usually formulated in plain text. Algorithmic level models describe the proposed function in a programming language or in a statistical framework like graphical models (Koller and Friedman, 2009). In the implementation level researchers seek the link between programs of algorithmic level and the biological neural networks that can exhibit the same function.

Let us now illustrate these three levels in the context of EM. Schacter's popular book Seven sins of memory (Schacter, 2001), characterizing several memory phenomena is an example of computational level description. Most of the agent based models reviewed in Sec. 2.3 like (Nuxoll and Laird, 2011; Derbinsky and Laird, 2009; Ho et al., 2008; Deutsch et al., 2008; Tecuci and Porter, 2007; Brom et al., 2007) are focused on the algorithmic level. They often store the memory content in lists, trees or general graphs. These are representations that are far from the neuronal substrate. Implementation level models of EM (Pollack, 1990; Lisman, 1999; Shastri, 2002; O'Reilly and Rudy, 2001; Ramamaurthy et al., 2004; Ito et al., 2013; Takac and Knott, 2013) are more focused on different architectures of neural networks capable of storing sequences of symbols. Some works do not explicitly use the neural network metaphor but the level of abstraction and sub-symbolic representation is almost the same, e.g., (Howard and Kahana, 2002).

Orthogonal to this classification we can take into account the purpose of the models. We use these categories:

Modeling a phenomena — these models usually try to fit some trend in empirical data, e.g., Directed Forgetting Paradigm (DFP), which is consciously initiated forgetting, is modeled by (Lehman and Malmberg, 2009); recency and primacy effects in serial position effect, which is the tendency to better recall items at the start and end of a remembered list, is modeled by (Burgess and Hitch, 1999; Howard and Kahana, 2002). Not all models try to fit the data quantitatively, some only capture the qualitative trends. Models whose main aim is modeling these trends are reviewed in (Norman et al., 2008), where they are labeled as biological and abstract models.

Generic models — the ambition of these models is to support multiple use-cases that require storage of sequential data at once. Generic EM models are usually embodied in some more general cognitive architecture, e.g., Soar's (Laird, 2012) EM models (Nuxoll and Laird, 2011; Derbinsky and Laird, 2009), ACT-R's (Anderson et al., 2004) EM model (Schultheis et al., 2007) or LIDA's (Baars and Franklin, 2009) model (Ramamaurthy et al., 2004). For a review of cognitive architectures that also mentions role of memory see (Langley et al., 2009). However, not all generic EM models are associated to some cognitive architecture, an example of such a standalone model could be (Tecuci and Porter, 2007). Compared to the models from the previous category these models are usually not evaluated by modeling a trend in empirical data. Performance of these models is more often assessed by their contribution to some higher level task accomplished by the agent, e.g., overall performance of a simulated tank in (Nuxoll and Laird, 2011).

Ad-hoc models — these were created with particular engineering applications in mind such as creating more engaging dialogs and they are usually tightly bound to it, e.g., (Liu and Maes, 2004; Dodd and Gutierrez, 2005; Kopp et al., 2005; Dias et al., 2007; Ho et al., 2007; Lim et al., 2009; Mattar and Wachsmuth, 2012; Rabe and Wachsmuth, 2012). They can be sometimes generalized to other domains but this usually is not the aim of these models.

In the review we will focus on the latter two categories of models because those are the most relevant to IVAs and therefore also to our proposed framework (see the list of requirements on our framework from Section 1.3). That is we will survey generic and ad-hoc models that fall almost exclusively into an algorithmic/representational level of description. Some prominent models of the implementational level will be described just to give the reader a sense of how different these models are. Even though they are not in the main scope of our review we find it important to make a link to them because a community working on ad-hoc models may not be aware of the implementational models and vice versa.

2.2 Axes Along Which the Models Will be Organized

In the rest of the review of EM models we will use following the axes along which the models will be described:

Aim — what the main purpose of the model is. What features it provides to the agent or what research questions it tries to answer.

Domain — description of the domain where the model was deployed/evaluated. What the complexity of the environment is, if it is deterministic or nondeterministic, etc.

Encoding-Storage-Retrieval — description of all the main memory processes, this includes data structures and algorithms consisting the model.

Evaluation — how the model was evaluated, e.g., does it somehow enhance agents performance or users experience when interacting with the agent?

2.3 Existing Episodic Memory Models - Algorithmic level

2.3.1 Subagdja et al. 2012

Aim. Subagdja et al. (2012) created a dual memory model that shows interplay of episodic and semantic memory and inspects the functional role of forgetting. The benefit of equipping an agent with EM was measured on a subproblem of a bot's Decision Making System (DMS) concerning picking the most efficient weapon given a distance to the enemy.

Domain. The model was implemented in a bot for a FPS game Unreal Tournament 2004 (UT2004)[1]. This game features a rich nondeterministic 3D environment. The bots' main task is to kill as many opponents as possible while surviving without being killed.

EM stores sequences containing the bot's position, health, ammo, distance to the enemy, weapon that was picked up, what behaviour the bot was executing, etc.

Encoding-Storage-Retrieval. Episodes are stored in a fusion adaptive resonance theory (ART) network (Tan et al., 2007). The ART is a type of neural network with two layers. The first layer serves as an input of the network and the second layer has neurons that cluster the inputs in an unsupervised manner. If any existing category does not describe the input well enough, the model can dynamically add a new neuron representing this new category.

The EM model in Subagdja et al. (2012) consists of two interconnected ART networks. The first network realizes *percepts* $\rightarrow$ *events* mapping and the second network realizes *events* $\rightarrow$ *episodes* mapping. Thus output of the first network is input to the second network that categorizes sequences of events into distinct episodes. Episodic memories undergo exponential decay forgetting, the strength of a memory is refreshed when it is accessed.

Once at a time the detailed sequential episodic memories are all retrieved and used to derive semantic knowledge — the rules for the agent's DMS. Semantic memory is also implemented as the ART network. The rules learned by abstracting from episodic memories and later stored in the semantic memory can be in a form: *IF distance to enemy < 299 THEN rocket_launcher effectiveness is 0.972.*

Evaluation. Performance of EM was measured by its contribution to the agent's overall performance. One subproblem of the bot's DMS is weapon selection. That is picking the right weapon given the current state of the game, e.g. distance to opponent, type of environment etc. A bot equipped with EM was compared to modified versions of that bot that performed 1) random weapon selection and 2) learned the weapon selection rules directly without using the EM as a short term buffer. Several speeds of forgetting were also evaluated.

Experiments showed that the bot with dual memory model outperformed the competitors. It is interesting that forgetting also enhanced bots performance. Authors hypothesize that it probably filters noise contained in the observations that harms semantic learning.

[1]Epic Games, Inc.: Unreal Tournament 2004, URL: http://en.wikipedia.org/wiki/Unreal_Tournament_2004 [28.12.2013]

2.3.2 Deutsch et al. 2008

Aim. Deutsch et al. (2008) investigates an agent architecture that can learn from past experience. The positive effects of learning are supported by limited experimental evaluation.

Domain. The model was implemented in agents acting in a fungus eaters like environment (Toda, 1982). The simple two dimensional environment contains obstacles, several types of terrain and energy sources. The world is inhabited by competing teams of agents, the goal is to survive as long as possible. The important aspect is that the environment encourages cooperation of agents, e.g., some food sources can be consumed only when there are two or more agents at the same time. The paper does not describe what type of information is exactly stored by the EM.

Encoding-Storage-Retrieval. The model stores plain sequences of *events*. The event consists of a triple $\langle emotions, TI\ matches, actions \rangle$. The TI match indicates that an event matched some *scenario*. Scenarios are templates of sequences of actions that have some functional meaning. Instances of templates, that is sequences of realized actions, are called *episodes*. Each event has its *salience* that relates to probability of recall. The initial value of salience is the weighted sum of saliences of emotions, TI matches and actions. After the event was stored the salience exponentially decays over time, it raises only when the event is recalled. In retrieval each stored event is compared to the (partially specified) cue and then the matching events are sorted according to a degree of overlap and activation. After this the memory can be traversed forward in time from the matching event.

Deutsch's model is an example of EM model that takes into account extended temporal relations between events. This is enabled by a scenario matching mechanism. An event can have high salience when it is part of an important scenario. Salience of the same event can be lower when it is a standalone event that does not fit to any scenario. This can be viewed as an enhancement over the other models reviewed in this section. The EM not only stores sensory data, but it also enriches them with higher level information about the scenario they belong to.

In general the task of inferring higher level scenarios/activities from low level sensory information is called an automatic activity recognition. An activity recognition algorithm based on a similar idea is presented by Kerr et al. (2011). The activity recognition layer is a key innovation compared to the previous EM models.

Retrieval is initiated by the agent by providing a partially specified event that is used as a cue. The cue is then compared to all stored events and the best matching event is returned.

Evaluation. The main metric in evaluation was the agent's expected lifetime. It was shown that agents equipped with EM survived 20% longer than the memoryless agents. The EM was used to foresee the impact of an agent's actions and thus it influenced the agent's DMS. In 83% of situations prediction from the episodic memory was accurate, thus the agent was able to use this knowledge to its advantage.

2.3.3 Soar 8

Aim. Nuxoll and Laird (2012) create a generic EM model for cognitive architecture Soar 8. The EM model is intended to support four functions of the agent's DMS. These functions are: *action modeling*, *retroactive learning*, *boosting other learning mechanisms* and *virtual sensing*.[2]

Domain. The model was evaluated in the Eaters and TankSoar environments (Nuxoll and Laird, 2012). Both are distributed with Soar and they serve as standard environments for demonstrating Soar's capabilities. Both environments are grid based. In Eaters each cell contains either an obstacle or a normal or a bonus food. An agent's goal is to eat as much food as possible. Eaters are similar to PacMan with the exception that there are no opponents. In the Eaters domain the eater agent has four actions for moving in all directions.

TankSoar is more complex. The agent controls a tank whose goal is to kill the opponent by firing a missile. The total number of actions runs into the to hundreds.

The EM model stores a limited map of an environment around the agent, an action taken by the agent and a reward.

Encoding-Storage-Retrieval. At each time step the EM encodes a tree graph representing a snapshot of the agent's WM. WM contains: the agent's perception field (a map of its surroundings); internal state and an action taken in the environment. Only items whose activation is over a certain threshold are encoded (Nuxoll et al., 2004). This way the system avoids storing irrelevant memories. During storage content of the memory may undergo forgetting (Nuxoll and Laird, 2012). However, this is only optional. Similar to the work of Deutsch et al. (2008) retrieval uses partial matching combined with the activation level of the stored event. The best matching event is returned. Now we will detail how the functions mentioned in Aim use the EM.

Action modeling refers to estimating the utility of performing an action in some context. The agent's DMS is realized via the following function:

$$DMS(c) = \arg\max_{a \in Actions} U(a, c) \tag{2.1}$$

[2]EM model from Soar 8 together with the model from Soar 9 presented in the next section are probably the most developed models from all models discussed in this chapter.

where $U(a,c)$ denotes utility of an action a in a context c. EM is used to evaluate $U(a,c)$, it finds the memory entry $S^{match}_{a,c}$ closest to the current context and possible action. Then it advances forward in time in a sequence of memories connected to the found memory entry. After a fixed number of time steps (e.g., 10) it finds the future state of the agent $S^{future}_{a,c}$, this process is called mental simulation. In the end utility is computed as $U(a,c) = score(S^{future}_{a,c}) - score(S^{match}_{a,c})$.

The second mechanism is *retroactive learning*, where the EM is used as a cache for episodes for batch learning of Soar decision rules. This can be seen as a model of memory consolidation that runs over night and converts episodic memories into more generally applicable rules.

The third mechanism, referred to as *boosting other learning mechanisms*, connects EM with Soar's chunking mechanism. This is equivalent to memorization of $DMS(c)$ function instead of evaluating it each time again. This leads to faster decisions, thus potentially enhancing the agent's behavior in relatively stable environments where the stored values remain the true estimate of the best action for a given situation.

Virtual sensing is the fourth mechanism that can be used to find locations of objects not present in the current agent's field of vision. This is similar to a type of memory studied by Ho et al. (2008), this model is described later in the text in Section 2.3.5.

Evaluation. In both Eaters and TankSoar domains the model was shown to improve the agent's performance over time as it learns important properties of the environment. For instance, in an experiment where past experiences were used in DMS the agent for TankSoar was able to learn certain behavior patterns and subsequently outperformed the baseline agent (Nuxoll and Laird, 2012).

Nuxoll et al. (2010) evaluates performance of three forgetting mechanisms in a new domain. Two other EM models discussed in this review (Tecuci and Porter, 2007; Ho et al., 2008) were also tested in this domain. It was shown that an activation based forgetting performs the best as the number of stored memories increases. A similar trend holds also for the other two models. To our knowledge this is the only evaluation that compares multiple different EM models on the same task. However, it remains unclear how performance of these models would compare to other state of the art techniques.

2.3.4 Soar 9

Aim. EM implementation in Soar 9 (Derbinsky and Laird, 2009) drops activation related features of the model in Soar 8 (Nuxoll and Laird, 2012) in favor of speed enhancements. The main aim is real time applicability of the model in agents that can run for extended time periods.

Domain. The model was applied to a diverse set of domains (Derbinsky et al., 2012) including a linguistic task of word sense disambiguation (WSD), 12 different planning domains (variations lead to 44 domain instances), video game like environments TankSoar, Eaters, Infinite Mario and also a mobile robot. This makes this model probably the most widely tested model from all the reviewed models.

Encoding-Storage-Retrieval. The model extends the previous implementation (Nuxoll, 2007) by ability of encoding whole graphs (called working memory graph (WMG)) instead of only trees. On the other hand it drops the activation based features. The model uses an SQLite relational database engine for storage of episodes which helps to meet the real time requirements. Besides this the encoding-storage-retrieval cycle is the same as in the previous version of Soar's EM.

Evaluation. The evaluation published in (Derbinsky et al., 2012) tests real time applicability of the model. It identifies tasks that can be computed with low latency and also types of queries that scale linearly with time and thus they are not suitable for agents running for extended time periods. For instance, in the planning domains cue matching queries in 12, relatively small domain instances were reactive. Whereas the remaining 32 domain instances were not suitable for real time queries.

2.3.5 Ho et al. 2008

Aim. Ho et al. (Ho et al., 2008) implemented a set of agents equipped with increasingly complex memory systems. The aim was to test enhancements in performance enabled by these memory systems.

Domain. Agents are placed in a virtual 3D environment similar to Toda's fungus eater (Toda, 1982). There are multiple agents that compete for resources. The environment also has a long term periodicity when its properties change during simulated winter and summer.

The LTM system stores in each record season when the record was created (winter/summer), type of the land square (e.g., desert), object sensed on the square (cactus) and change of agent's internal variables (e.g., Δenergy=-0.03) possibly conditioned by usage of an object (stone).

Encoding-Storage-Retrieval. Memory is implemented as a plain log of events experienced by the agent. Memory retrieval in the LTM model works in the following way. The model retrieves a sequence that is the most relevant to the task of getting from the *current* state to the *target* state. The search uses $matchKey$ to specify states that are similar to the *current* state, and $searchKey$ for *target* states. An example can be $searchKey = \{sensed(AppleTree)\}, matchKey = \{landform(Oasis)\}$. This corresponds to a situation when the agent is looking for an apple tree and he is in an oasis. The retrieval algorithm looks for a set of target states and then tries to go forward and backward in time and find states matching *matchKey*. The resulting sequence is then redone or undone by the agent depending on its direction in time.

Evaluation. The study compares performance of a purely reactive agent, an agent with a STM (where STM is a list of last n experienced events) that can "undo" previous behaviour to reach the desired resource, a LTM agent that has longer memory store and it can retrieve multiple matching records and combinations of STM and LTM memory systems. A possibility of simple communication between agents realized by exchange of memory query results is also tested.

The evaluation shows that agent architecture equipped with a combination of LTM and STM outperforms the reactive and STM agents in expected lifespan of the agents. The results are statistically significant. Evaluation also tests the setup when the agents can share memories, which also leads to increased lifespan.

2.3.6 Model of FearNot!

Aim. EM model (Dias et al., 2007; Ho et al., 2007) enhances virtual agents with the ability to remember interaction of the user's avatar with other agents. The memories are used in debriefing where they are post-processed by a natural language generator in order to be presented to the end user.

Domain. The model was used in the context of the anti-bullying educational application FearNot! (Aylett et al., 2005). FearNot! is a training environment that lets pupils experience different situations that may involve bullying. The pupils should learn how to behave in such situations and hopefully avoid them in the future. The EM stores what happened to the pupils in the last session and it is used in debriefing where pupils can assess their own behavior.

Encoding-Storage-Retrieval. The model stores experiences of agents in a plain sequence. Each entry has the form of a triple ⟨*abstract summary, detailed description, evaluation of the situation*⟩. The detailed description consists of ⟨*time, people, location, object, details, feelings*⟩. An emotional generator based on the OCC theory (Ortony et al., 1990) is used to evaluate each episode and the emotional impact is later used in retrieval. An example of the emotional generator output might be the fact "I like agent X". Retrieval is initiated by providing a partially specified cue, e.g., a set of characters that should be in the episode or a place where the episode happened. Only the most emotionally salient events can be recalled.

Evaluation. No specific evaluation of the EM module was performed. However the FearNot! application as a whole was evaluated in a study with several hundred pupils (Sapouna et al., 2009).

2.3.7 Tecuci — A Generic Memory Module For Events

Aim. Tecuci and Porter (2007, 2009) try to create a standalone generic EM model with a well defined API. It should be easy to interface the model with any agent architecture. This should promote the use of EM like systems in a wider range of applications.

Domain. Since the model is designed as generic it is not bound to any specific domain. The only restriction is that the memory entries have the structure of triples $\langle context, content, outcome \rangle$. Where context describes initial settings of the episode, e.g., a state of the world when the episode occurred; content is a sequence of actions; and outcome specifies how the episode ended, e.g., whether it was successful.

Encoding-Storage-Retrieval. In storage episodes persist unchanged as they were perceived. The only additional operation is indexing of context, content and outcome. Thus the episode can be queried by all three dimensions. In retrieval the partially specified episode serves as a cue and graph matching algorithms are used to match it against stored episodes. The index is used to speed up the search. Tecuci's model is inspired by document retrieval research, whereas most of the other reviewed models are rooted in psychology.

Evaluation. Evaluation presented in (Tecuci and Porter, 2007) tests the EM model on a corpora of artificial plans from a logistics domain generated by the planner SHOP2. The domain described the delivery of a package between three cities. In evaluation the memory is tested in three goals: planning — find a plan for a given goal (e.g., find a sequence of actions that must be executed in order to deliver a package from city A to city B); classification — answer whether the goal is solvable (e.g., is it possible to deliver the package from A to B?); goal recognition — infer the goal given a sequence of actions (e.g., is the package going to be delivered to city A or B?). The EM model was compared against the 5-nearest neighbor (5-NN) algorithm that searches through all stored entries. Evaluation showed that EM models performed comparably well to 5-NN in terms of accuracy. But at the same time it compared the cue against significantly fewer stored episodes than 5-NN. This was possible because of the indexes that guided the search.

Further evaluation in (Tecuci and Porter, 2009) tests the model on two new domains: Monroe plan corpus (Blaylock and Allen, 2005a) and sequences of Linux shell commands. Performance of the EM model was comparable to a model based on Bayesian statistics (Blaylock and Allen, 2005b).

2.3.8 Li et al. 2013

Aim. The aim of the model presented by Li et al. (2013) is to create a generic memory system applicable to IVAs. The model should simulate forgetting and recall of false memories (Brainerd and Reyna, 2005). The model accounts for WM and LTM. The EM is not the main focus of the model. However, the EM can be seen as a subset of the LTM that is modeled.

Domain. The model uses graph based representation of facts, thus no particular structure of the domain is expected. An agent equipped with the memory model was tested in a 3D environment, where its goal was to search for previously seen objects.

Encoding-Storage-Retrieval. Similarly to Soar the model uses graph based representation. Nodes represent concepts (i.e., objects and theirs properties) and directed edges represent relations between the concepts. Both nodes and edges have associated *strength*, the higher the strength the more important the node/edge.

As the first step of encoding observations enter the sensory memory which has only a limited capacity and it can store observations only for a limited time span. Thus only a subset of present objects receives the agent's attention. Then the graph representing the observations enters the working memory. Strength of edges between nodes simultaneously present in the WM are reinforced. Finally, the graph is stored in the LTM. During storage in LTM both the nodes and the edges undergo exponential decay that can result in deletion of the nodes/edges whose activation is below a forgetting threshold. In recall, the nodes/edges whose strength is below a certain threshold can be confused with a similar concept, e.g., a blue color may be incorrectly recalled instead as a red one.

Evaluation. Performance of an IVA equipped with the proposed memory model was compared to performance of humans. Both IVA and humans first explored an unknown environment, then they were instructed to find an object they saw previously. Various parameters of the model were estimated from the data so that the performance of the IVA matches human level behavior in this particular task. However, it might be the case that the same behavior might be achieved even with a simpler model.

2.3.9 Brom et al. 2007

Aim. The aim of the work presented by Brom et al. (2007) is to create a model capable of storing hierarchical structure of episodes. The model is designed with IVAs' needs in mind.

Domain. The model was tested in an agent inhabiting a simple grid world. The agent had repeating goals (e.g., eat or sleep) and random goals (e.g., cure).

The environment contains objects that are needed for completion of some goals. Dynamic changes of the environment are modeled by random changes of objects' locations. The model stores the agent's behavior on several levels of abstraction. The lowest level contains atomic actions. The higher levels contains episodes that might span multiple time steps.

Encoding-Storage-Retrieval. The model does not store only plain sequences of observations as most of the other reviewed models but it also stores hierarchical temporally extended episodes that include multiple observations/sub-episodes. The agent's DMS is encoded using AND/OR trees, thus there is inherently a hierarchical decomposition of the agent's goals. Every time the agent executes an atomic action he stores this action together with the whole path from the leaf to the root in the AND/OR tree. Each episode is represented on different levels of abstractions, which is consistent with findings from psychology (Zacks and Tversky, 2001).

During storage memories undergo gradual forgetting based on emotional salience of the remembered event, its depth in the hierarchy and time passed since its encoding. Therefore it might happen that low level details of an episode are forgotten but the high level summary of the episode remains stored in the memory. Fast retrieval in time interval queries is made possible by time indexes over the tree like memory structure. The basic model was further extended with simple statistical episode schemata (Čermák et al., 2011) and fuzzy time concepts (Brom et al., 2010) (e.g.,"around noon") implemented by a mechanism similar to a neural network.

Evaluation. The basic model was evaluated in terms of memory size with and without emotion based forgetting. In both cases the size of the memory grew linearly with time. However, emotion based forgetting helped to reduce the size of the memory. Plausibility of the fuzzy time concepts was tested in evaluation with human subjects (Brom et al., 2010). The evaluation shows preference for socially established time concepts ("around noon") rather than for an exact time specification (e.g.,"at 12:34").

2.3.10 Ziggurat

Aim. Ziggurat (Faltersack et al., 2011) is another domain independent EM model. The aim is to extend the EM from Soar 8 (Nuxoll and Laird, 2012) with the ability to store hierarchical episode representation. The memory is applicable in cases when an agent needs to reach a goal state from an initial state. Therefore, it is a system that realizes case based planning (Cox et al., 2005).

Domain. The Ziggurat model was evaluated in two testing domains, Eaters, known from the Soar, and a simple blind navigation domain. In the blind navigation domain the agent has to navigate in a maze with a tile based map from origin to a target location. The main limitation is that the agent is "blind" and it only senses the walls once it collides with them. In the presented experiments the memory stores sequences of the robot's actions.

Encoding-Storage-Retrieval. The term *episode* is used for a single ⟨perception, action⟩ pair recorded at one time step. Multiple consecutive episodes form a *sequence*. Sequences from the lower level of the hierarchy can be episodes in the higher level. This way a tree structure of more abstract episodes is formed. The hierarchy of episodes is used in context based retrieval.

In the learning phase the agent explores the structure of the environment. The agent performs sequences of random actions and records their outcome until the target state is reached. Then a set of *replacement rules* is applied to remove unnecessary actions from the sequence (e.g. replace three 90° turns left by one 90° turn right etc.). These rules have associated *confidences* that measure their reliability. If application of a rule leads to wrong conclusions then its confidence is decreased and vice versa. The learnt sequences represent plans that can be later replayed by the agent.

After the memory has been learned it can be used in the runtime. The approach is similar to (Nuxoll and Laird, 2012), the best matching situation is found in the memory and then the agent replays the retrieved actions. The difference is that in (Nuxoll and Laird, 2012) the match is based on properties of a single time step, whereas Ziggurat finds the longest matching sequence of the same states in the memory. When a future state differs from that predicted by the memory then a new longest match is found.

Evaluation. Even in the simplest blind navigation domain where an optimal solution required only 3 actions the EM controlled agent used on average 24 actions. In the remaining two test cases the agent needed about four times more actions than was the optimum. In the Eaters domain the performance was comparable to the EM model from Soar 8 (Nuxoll and Laird, 2012).

2.3.11 Kope et al. 2013

Aim. The EM model presented by Kope et al. (2013) is another generic model aimed at increasing believability of IVAs. It aims to mimic properties of human memory such as false memories and associative recall.

Domain. The model was connected to a villager agent acting in a 3D game Minecraft[3]. A player can ask the villager to recall memories associated with a certain keyword. For example, the player can initiate the agent's recall with the keyword "wood" and the agent replies "I traded some wood to Duncan McGrath". However, the model is formulated in a generic way and it stores abstract graph

[3]Minecraft's homepage: URL: https://minecraft.net/ [23.2.2014]

structures.

Encoding-Storage-Retrieval. The memory is represented as a graph of concepts and directional links between the concepts. The concepts represent objects present in the environment and possibly also actions. Each concept has its activation that changes in time, links between concepts have weights representing strength of association.

During encoding the observed situation is translated into a graph of concepts and strength of links between concepts often appearing together is increased. In storage activation of concepts exponentially decays, resulting in deletion of concepts with low activation. Additionally some concepts may be randomly altered to a semantically related concept, this simulates false memories phenomenon. Retrieval is implemented through a spreading activation mechanism (Anderson, 1983). A search cue is used to increase activation of related concepts and from them the activation spreads through the memory graph. The N most active concepts are used as a result of the query. The idea of spreading activation was previously used in the context of IVAs by Lim et al. (2011).[4]

The model does not explicitly account for time. Thus sequences of events are not directly representable in the model as presented by Kope et al. (2013). Because of this the model does not fit in the scope of our review. It is representative of spreading activation models in the context of IVAs and as such it deserves to be mentioned. Moreover the model can be extended to account for time by addition of new types of links between the concepts that will represent time precedence.

Evaluation. The model was evaluated in terms of its computational requirements. In a limited test domain it was shown to be suitable for real time use.

2.4 Existing Episodic Memory Models - Implementational level

This section describes models that usually use low level metaphor of rate coded "neurons" or units with similar function. In Marr's classification these models fall into the implementational level. Thorough review of these models is provided in (Norman et al., 2008). Here we will describe three representative frameworks just to illustrate their difference from the models described above.

[4]This model was not included in the review since it was not described as extensively as the other models.

2.4.1 Temporal context model

Aim. The main aim of the model named temporal context model (TCM) presented in (Howard and Kahana, 2002) is to create a simulation that can account for empirical data from psychological laboratory experiments concerned with remembering words from longer lists. Namely it models recency and contiguity effects in free recall. Recency effect in a free recall (Murdock, 1962) is a tendency to recall more items that appeared towards the end of a list compared to items in the middle. Contiguity effect (Kahana, 1996) is a tendency to recall items that were close together in the list, e.g., if one recalls the item on position i it is much more likely that the next recalled item will be from position $i + 1$ rather than from $i+3$. TCM models these effects with low level, neuronal-like architecture. A review of the TCM approach and current findings from neurobiology supporting this view is given in (Polyn and Kahana, 2008).

Domain. The original domain where the model was applied are lists of words. However, the model can be used for any sequences of symbolic data.

Encoding-Storage-Retrieval. Instead of using pointers to the next event as in (Nuxoll and Laird, 2012; Derbinsky and Laird, 2009; Ho et al., 2008; Brom et al., 2007) TCM uses more low level encoding of successive events. Flow of the time is not represented by an explicit pointer, instead there is a slowly shifting temporal context and events are associated to this context. There is a mapping M^{FT} between events and a context and also an inverse mapping M^{TF} from the context to the events. These mappings are learned by Hebb's rule (Hebb, 1949). Each time an event is experienced its weight associating it with the current context is strengthened. This models encoding of the event. During storage the mapping remains intact. Although the context may change, which can lead to degraded performance in recall. When an item is recalled the associated context is also reconstructed. This context then influences recall of the following events. Since the context of the events that happened close together is similar the probability of recalling them is also higher.

Evaluation. It was shown that model parameters can be fitted to account for human data for both recency and contiguity effects (Howard and Kahana, 2002).

2.4.2 Lisman 1999

Aim. Lisman (1999) hypothesized how sequences of items can be stored by a neural network located in the brain structure called the hippocampus. He discussed several possible schemata of connection of hetero-associative and auto-associative neural networks that would be able to recall sequences by providing partial clues. The proposed networks are also aligned with CA1 and CA3 regions of the hippocampus. The aim of this work was to provide a theoretical framework that can represent successive items in episodic memory.

Domain. The generic framework is applicable to any sequence of items.

Encoding-Storage-Retrieval. The model proposes that the sequences can be stored in two interconnected networks, a hetero-associative network that performs mapping of successive events, and an auto-associative network that can correct outputs of the hetero-associative network. For instance, suppose that the hetero-associative network is presented with pattern A and it outputs B', then B' is passed to the auto-associative network that can correct previous inaccurate recall and map B' to the true value B. Then B goes back to the hetero-associative network that recalls C'. Thus the sequence is stored in weights of the neural network. This is similar to the model of Howard and Kahana (2002) where the list is represented by many distributed weights.

Evaluation. As presented in (Lisman, 1999) the framework was not implemented. Hence no empirical validation was performed.

2.4.3 LIDA

Aim. LIDA (Franklin and Patterson Jr, 2006) is a general cognitive architecture, hence the main purpose of its EM module is to provide generic, biologically plausible memory store. LIDA's memory system (Anwar and Franklin, 2003; Ramamaurthy et al., 2004; Ramamurthy et al., 2006; Snaider and Franklin, 2012) is based on a sparse distributed memory (SDM) (Kanerva, 1988).

Domain. The research on sequential memory in LIDA (Snaider and Franklin, 2012), which is of primary interest for this review, was done only with sequences of abstract symbols. The rest of the memory system (Anwar and Franklin, 2003) was already implemented in a virtual secretary, CMattie, that communicates with users in natural language through emails.

Encoding-Storage-Retrieval. SDM works as an auto-associative memory. The memory consists of a set of *sparse locations* and counters. Each sparse location is a word of length n over a binary alphabet. Additionally for every position i inside each location there is a counter that is updated during memory encoding. In encoding each fact to be remembered is first translated into a binary word w of length n. Then a set of locations within some predefined distance d from w is searched. For each location counters are updated by the following rule: if i-th bit of w equals one then increment counter i of location l, otherwise decrement the same counter. Thus the memory is represented by a set of locations and associated counters.

In retrieval the memory is provided with a partially specified cue that is then reconstructed and matched to the closest recallable fact. Retrieval can use several iterative cycles where the recalled memory can be again used as a cue for the next turn of retrieval. This process "focuses" the retrieved memory. This can be useful when the initial cue contains noise. However, the recall can end in divergence. Therefore real world implementations must detect this divergence and end with some kind of exception. This iterated recall can be seen in several other memory architectures including Hopfield networks (Hopfield, 1982) or ART (Carpenter and Grossberg, 2003). There have been several extensions to SDM in LIDA, e.g., handling of incomplete knowledge (Ramamaurthy et al., 2004), forgetting (Ramamurthy et al., 2006) and recently also with representation of sequences (Snaider and Franklin, 2012). The schema proposed in (Snaider and Franklin, 2012) is similar to sequence encoding used in recurrent neural networks (Pollack, 1990). Each memory location consists of $2n$ bits. The first n bits represent a content of an episode for time t and the second n bits represent content for time $t + 1$. Iterative retrieval of sequences works in the following way. Suppose that you have the memory location for time t, therefore you can use the second half of this location as a cue to retrieve location for $t+1$. Analogically you can continue to retrieve $t + 2$ and so on. This schema is nearly identical to the one proposed by Lisman (1999).

Evaluation. Properties of the memory for sequences (Snaider and Franklin, 2012) were tested on abstract sequences of words. Evaluation has shown that with careful parameterization it is possible to store and retrieve up to 100 sequences of 20 elements in an SDM with 200 000 locations and 2000 bits per location, thus $4 \cdot 10^8$ counters were needed.

2.5 Activity Recognition

Activity/plan/goal recognition[5] (Schmidt et al., 1978; Kautz and Allen, 1986) can be formulated as a general problem of labeling a stream of actions with the most probable activity/plan/goal. This problem is jointly studied by several computer science disciplines. There are applications of activity recognition in video processing (Turaga et al., 2008; Ryoo and Matthies, 2013), cognitive assistants (Kautz et al., 2003; Liao et al., 2007b), ubiquitous computing (Yin et al., 2004; Oliver and Horvitz, 2005; Lu et al., 2010; Huynh et al., 2008; Stikic and Schiele, 2009; Kadlec and Brom, 2011), virtual environments (Kerr et al., 2011; Fagan and Cunningham, 2003), natural language story understanding (Charniak and Goldman, 1993) and security (Lisý et al., 2012).

Several different types of techniques were used to tackle the plan recognition problem. The first group of algorithms is based on propositional logic represen-

[5]Different fields of computer science tend to use different names for this problem. Activity recognition usually refers to recognition of low level actions like sitting or walking. Plan/goal recognition is usually focused on higher level activities like commuting or cooking.

tation (Kautz and Allen, 1986; Lesh and Etzioni, 1995; Sindlar, 2011; Ramirez and Geffner, 2009). These need to have a representation of the possible executed plans described in a formalism similar to propositional logic. For instance, Sindlar (2011) uses a logic-based approach called mental state abduction. Mental state abduction tries to deduce all possible explanations that lead to an observed behavior. In a sense, it tries to "parse" the observed events with a logic program encoding "grammar" of all possible plans. A similar technique utilizing formalism of automated planning (Ghallab et al., 2004) was proposed in (Ramirez and Geffner, 2009). It uses the assumption that if a rational agent tries to accomplish a goal G, then the sequence of actions observed so far should match a sequence of actions of an optimal plan P solving G.

The second main approach uses Bayesian frameworks. Most of these algorithms rely on a probabilistic model that can be represented as a DBN. Different DBN architectures have been used through time, beginning with the most simple Hidden Markov Model (HMM) (Rabiner, 1989) used in (Oliver and Horvitz, 2005). More complex generic DBN topologies suitable for activity recognition were proposed in subsequent work. For instance, Hierarchical Hidden Markov Model (HHMM) (Fine et al., 1998) (later reformulated as DBN in (Murphy and Paskin, 2002)) is a model that has several layers of interconnected HMMs. A slightly modified version of HHMM that allows to recognize the same goals on different levels of hierarchy was presented in (Bui et al., 2004). Other DBN based architectures include Abstract Hidden Markov Model (AHMM) (Bui et al., 2002), Abstract Hidden Markov Memory Model (AHMEM) (Bui, 2003), Cascading Hidden Markov Model (CHMM) (Blaylock and Allen, 2006), Layered Hidden Markov Model (LHMM) (Oliver et al., 2004), or ad-hoc network topologies as used in (Liao et al., 2007b). There are also models that were not originally formulated as a DBN but there exists DBN architecture that can recognize the same class of plans. This it the case of Probabilistic State-Dependant Grammars (PSDGs) (Pynadath and Wellman, 2000) and its counterpart AHMEM formulated as the DBN that both recognize the same set of plans. Further extensions of DBN models make it possible to account for duration of activities. Those models are known as Hidden Semi-Markov Models (HSMMs). A recent example of such a model is Coxian hidden Semi Markov Model (CxHSMM) (Duong et al., 2009).

A different type of probabilistic graphical model — Condition Random Field (CRF) (Lafferty et al., 2001), originally developed for natural language processing, was used also for activity recognition (Vail et al., 2007; Liao et al., 2007a). Dynamic CRFs (Sutton et al., 2007), the natural counterpart of DBNs, might be interesting for applications in activity recognition as well. There are also other probabilistic models not directly interpretable as a graphical model like PHATT (Geib and Goldman, 2009).

Alternative approaches tried to use Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998) formalism for plan recognition (Ramirez and Geffner, 2011). This can be viewed as an extension of the approach presented in (Ramirez and Geffner, 2009) to nondeterministic domains.

Even though Bayesian Networks (BNs) are the dominant technique in most applications, there are also formalisms that try to extend this basic concept. For instance, mixed networks (MNs) (Dechter and Mateescu, 2004) extend BNs with deterministic constraints known from constraint satisfaction programming (CSP) (Dechter, 2003). An extension of MNs to dynamic domains can be directly applied to activity recognition (Gogate et al., 2005).

A different extension of graphical models is made in Markov Logic Networks (MLNs) (Richardson and Domingos, 2006). MLNs unify probabilistic reasoning with first order logic and they were recently applied to activity recognition in various domains (Sadilek and Kautz, 2010; Ha et al., 2011; Song et al., 2013).

Another extension of the DBN formalism are Continuous Time Bayesian Networks (CTBNs) (Nodelman et al., 2002; Nodelman, 2007). The distinction between DBNs and CTBNs is that DBNs compute each process/episode at granularity that is given by the rate of the process that changes the fastest, which can be associated with significant computational overhead. CTBNs overcome this limitation and they make it possible to update each process at a different rate.

An alternative direction that recently attracted a lot of attention is to use models based on neural network (NN). An example of advanced NN is convolutional neural network (CNN) from (Simonyan and Zisserman, 2014).

2.6 Hierarchical Activity Representation of IVAs

One of the core assumptions of our proposed EM framework will be that flow of events can be represented in a hierarchy of so called *episodes*. This seems to be true both for humans (Zacks and Tversky, 2001) and for many IVAs. Many formalisms popular for programming agents in both academia and industry use some kind of hierarchy. Hierarchical decomposition of behavior is beneficial for the designer since it can be more easily maintained and extended over time. The formalisms utilizing some kind of hierarchy are, for example hierarchical finite state machines (HFSMs) (Harel, 1987; Houlette and Fu, 2003), behavior trees (BTs) (Colledanchise and Ogren, 2014) (represented by, e.g., POSH (Bryson, 2001; Bryson and Stein, 2001)) or hierarchical task networks (HTNs) (Ghallab et al., 2004).

The HFSM is a Finite State Machine (FSM) with addition of so called hierarchical states. The hierarchical state consists of another FSM. It is possible to transition from state to state but also from state to hierarchical state and vice versa. Therefore it should be easy to author a single FSM for some specific behavior and add it to an already existing IVA. States of the HFSM usually represent a low level behavior like *walking* or *eating*. Transitions have associated conditions that enable/disable the possibility of transitioning from one state to another based on the state of the agent and environment. An example condition might be $isEmpty(PLATE) : eating \rightarrow walking$ that translates to: "Once the plate is empty finish *eating* and start *walking*".

BTs provide a slightly different approach to programming an IVA's behavior. BT is a tree that contains *selectors* and *sequences* as internal nodes and *actions* and *conditions* as leaves. Selector nodes execute the first applicable sub-nodes which are sorted according to their priorities. A sequence node executes all its children in a fixed sequence. Conditions determine whether the subtree containing this node is executable. Action nodes perform actions in the environment. Execution of every node might fail. In this case the decision logic tries to pick the next applicable node. For instance, the IVA might fail to *start the car* therefore the whole subtree *COMMUTE BY CAR* fails and the decision making logic tries to execute the next applicable behavior which is *COMMUTE BY PUBLIC TRANSPORT*.

HTN extends the popular STRIPS (Nilsson and Fikes, 1971) formalism with a notion of hierarchy. The designer can describe the hierarchical nature of the plans by so called *compound tasks*. Compared to HFSMs and BTs which are procedural techniques HTN provides a declarative approach for prescribing the IVA's behavior. The designer describes the structure of the task, effects and preconditions of actions, the initial state and the desired target state and he lets the planning algorithm find a solving plan. The disadvantage of planning is its time complexity, for analysis see (Erol et al., 1994).

As we can see all the previously mentioned formalism provide high level abstractions of the final behavior. In HFSM we can obtain the hierarchy from hierarchical states, in BT from the selector and sequence nodes and in HTN from the compound tasks. All of them provide summary of the underlying behavior.

2.7 Summary

This review tried to contrast different approaches used in EM modeling by algorithmic level models (Section 2.3) and implementational level models (Section 2.4). Further we reviewed probabilistic models used for activity recognition (Section 2.5), some of them will be used in our proposed EM model. In the end we briefly reviewed popular techniques for programming agents' behavior. All of these techniques use some kind of hierarchy, this observation will be used later in our model.

The algorithmic level models often use a list of experienced events as the main data structure. On the other hand implementational level models model flow of time by lower level mechanisms. For example, by associative properties of neural networks (Lisman, 1999).

The next section will briefly summarize survey of the activity recognition algorithms. Then we summarize the algorithmic level models since these are more relevant for our proposed model than the implementational level models.

2.7.1 Summary of Activity Recognition

Our review of activity recognition algorithms shows that there is a wide range of formalisms for describing structure of recognized activities, ranging from symbolic representation used in logic to sub-symbolic representations like weights of a NN. Among the presented approaches those based on DBNs are the most interesting for us since they: 1) allow to use the same probabilistic model for multiple tasks; 2) make it possible to represent hierarchical structure of IVA's behavior (as discussed in Section 2.6).

2.7.2 Summary of Algorithmic Level Models

This section summarizes capabilities of the reviewed algorithmic level modes and discusses how they fulfill the requirements listed in Section 1.3.

1. **Summarization and clarifying questions:** The model of Brom et al. (2007) makes it possible to summarize activity of the agent itself. The model of Deutsch et al. (2008) has the same ability. Additionally in principle it can summarize activity of the other agents, even though it was not used in this way. Models presented in Subagdja et al. (2012) and Faltersack et al. (2011) are able to construct hierarchical representation of low level actions. However, the hierarchy is constructed in an unsupervised manner. Therefore it is not possible to learn the model how it should interpret observed actions based on a set of labeled example sequences. None of the reviewed models uses advanced activity recognition techniques surveyed in Section 2.5 to interpret behavior of observed agents. All models that represent episodes in a hierarchical structure make it in principle possible to answer further clarifying questions.

2. **Expressing degree of certainty:** Expressing certainty of recall is made possible by the models that encode memories in a graph of interconnected concepts (Kope et al., 2013; Li et al., 2013). An activation associated with concepts and possibly edges can be used as a measure of certainty. The higher the activation the more certain is the recall.

3. **Believable mistakes in recall:** Some models (Kope et al., 2013; Li et al., 2013) simulate believable mistakes in recall by randomly altering the content of the memory. The recalled concept is usually replaced by a similar concept with certain probability. A different process of inducing false memories is used by Čermák et al. (2011) which extends the model of Brom et al. (2007). Here the false memories emerge as a result of merging similar memories that were close in time.

4. **Recall interesting details first:** Several models propose to use emotional feedback to measure how interesting the stored events are (Brom et al., 2007; Dias et al., 2007; Deutsch et al., 2008; Gomes et al., 2011; Kope et al., 2013). This implies that the agent has to be equipped with an emotion evaluator module. We will cast the problem of measuring interestingness as computation of "distance" of two probabilistic distributions.

We can see that none of the models accounts for all our requirements in a coherent framework.

Further, based on intended use of the models we can identify at least two model subgroups. The first group (Deutsch et al., 2008; Ho et al., 2008; Derbinsky and Laird, 2009; Faltersack et al., 2011; Nuxoll and Laird, 2012; Subagdja et al., 2012) uses EM as a tool that should enhance performance of an agent living in a simulated environment. This type of agents and environments can be usually formalized by the framework of Markov Decision Processs (MDPs) (Bellman, 1957) or POMDPs (Kaelbling et al., 1998). In this case the agent's performance can be exactly measured by feedback from the simulator (e.g., damage suffered by the agent, number of collected bonus items, etc.). However, these models are not compared against the state of the art techniques for solving MDPs and more general POMDPs studied in the AI community. Since there are still a lot of open issues in solving both types of problems interesting results might arise from casting this type of EM research as a subfield of POMDP research.

The second group (Brom et al., 2007; Dias et al., 2007; Ho et al., 2007; Li et al., 2013; Kope et al., 2013) is more focused on increasing believability of IVAs equipped with these EM models. Therefore it is more complicated to define performance of these agents since it depends on human assessment.

In this chapter we have shown that none of the current EM models fulfills all our requirements. Additionally we surveyed approaches to activity recognition and several popular techniques for programming an IVA's behavior. Knowledge of both these areas will inform analysis of our EM framework that follows in the next chapter.

Chapter 3

Problem Analysis

This chapter provides detailed analysis of our proposed EM model. First we will introduce our notation and we will analyze all processes needed in our model on an example scenario. Then we will analyze requirements on the probabilistic model used in our EM. In the end we summarize the encoding-storage-retrieval cycle of our model.

3.1 Notation

Uppercase letters will denote random variables (e.g., X, Y, O) whereas lowercase letters will denote their values (e.g., x, y, o). Probability mass function (PMF) of a random variable X will be denoted by $P(X)$. When X is discrete, $P(X)$ will be also used to refer to a table specifying the PMF. Conditional probability mass function (CPMF) denoted as $P(X|Y)$ expresses PMF of X given the true state of Y. Domain of X will be denoted as $D(X)$. Notation $X_{i:j}$ will be a shorthand for a sequence of variables $X_i, X_{i+1} \dots X_j$, analogically $x_{i:j}$ will be a sequence of values of those variables. When we want to stress that all variables in this sequence have the same value we will use notation $a_{i \sim j}$, this will be equivalent to $\forall k, i \leq k \leq j : x_k = a$ (this notation is used for instance in Figure 3.1). The subscript will usually denote time. A set of random variables will be sometimes also denoted by bold letters, e.g., $\mathbf{Z} = \{X_0, X_1, \dots, X_n\}$. A probabilistic distribution over the set of variables $\mathbf{Z}$ will be defined as $P(\mathbf{Z}) = P(X_0, X_1, \dots, X_n)$.

$\mathcal{M}$ will be a probabilistic model and $\mathcal{V}$ will be a set of all random variables in the model.

3.2 Example Scenario

In the following example scenario we will analyze all steps of the encoding-storage-retrieval cycle and we will discuss how the model can fulfill the requirements listed in Section 1.3.

Suppose that Bob observes Alice and he tries to remember what she does. Figure 3.1 shows this situation. Alice performs a sequence of atomic actions $\rho_{o:T}$. Bob's *perception* might be imperfect, therefore he observes a sequence $o_{0:T}$. In the following example we will assume that Bob observes only Alice's atomic actions. However, we can also include broader context of the situation in observations. For instance, each observation might contain also objects possibly used by Alice and her location. Note that Alice's higher level goals that drive her to perform this sequence remain hidden to Bob. The higher level goals are known to Alice, but Bob observes them only indirectly via the sequence of atomic actions. Thus to fulfill the requirements to support summarization (req. 1) and for further clarifying questions (req. 2) there has to be an activity recognition component that deduces the high level unobserved episodes (denoted as $a^1_{0\sim T}, b^0_{0\sim 1}$ and $c^0_{2\sim T}$ in Figure 3.1).

An input of the activity recognition phase is the sequence of observable actions and an output is a segmentation of this sequence where every segment has an associated "label". The label gives a name to the episode represented by this segment. Let both o_0 and o_1 from Figure 3.1 be atomic actions $STEP$ and let the episode $b^0_{0\sim 1}$ be $WALK$. Note that in order to perform segmentation and labeling the activity recognition component has to have a knowledge about the usual structure of episodes. This knowledge will be called *episodic schemata*. For example episodic schemata might encode the following rules: a sequence of multiple steps is walk, walk followed by bus ride might be a trip but it might also be commuting depending on the context. Everyone knows what it means to commute, even though there are more possible realizations of commuting. One might go by bus and then switch to subway or switch to another bus but both of these episodes will be labeled as commuting. If Bob says that Alice was commuting one gets the general idea what was going on.

Episodic schemata can be either handcrafted or learned from annotated examples of episodes. Handcrafted episodic schema can be defined by the designer based on his experience with the domain. The second approach, that is learning the schemata from data requires automated procedure that will extract significant properties of the example sequences and it will adjust parameters of the probabilistic model accordingly. Obtaining the schemata models procedural learning during childhood[1]. Once learned, the schemata remain fixed, becoming an un-

[1] In the present work, procedural learning refers to the kind of learning performed in cognitive

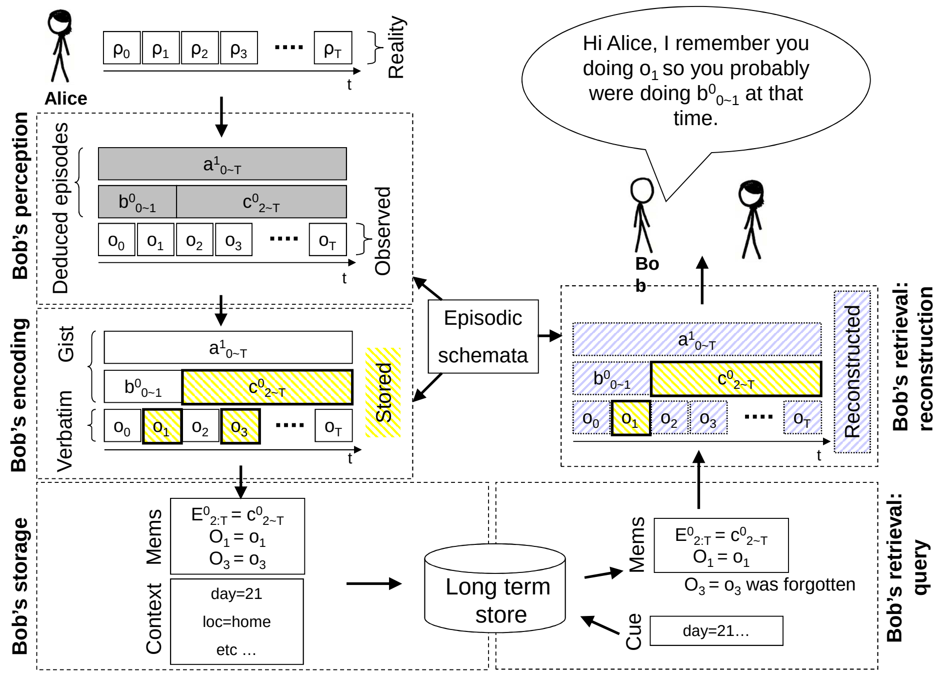

Alice
ρ_0 ρ_1 ρ_2 ρ_3 .... ρ_T
Reality
t
Bob's perception
Deduced episodes
$a^1_{0\sim T}$
$b^0_{0\sim1}$ $c^0_{2\sim T}$
o_0 o_1 o_2 o_3 .... o_T
Observed
Bob's encoding
Gist
Verbatim
Stored
Episodic schemata
Hi Alice, I remember you doing o_1 so you probably were doing $b^0_{0\sim1}$ at that time.
Bob
Reconstructed
Bob's retrieval: reconstruction
Bob's storage
Mems
$E^0_{2:T} = c^0_{2\sim T}$ $O_1 = o_1$ $O_3 = o_3$
Context
day=21 loc=home etc ...
Long term store
$E^0_{2:T} = c^0_{2\sim T}$ $O_1 = o_1$
$O_3 = o_3$ was forgotten
Cue
day=21...
Bob's retrieval: query

Figure 3.1: Perception-encoding-storage-retrieval cycle in DyBaNeM. Alice performs a sequence of actions $\rho_o \ldots \rho_T$, Bob perceives this sequence as $o_0 \ldots o_T$ and he deduces Alice's episodic hierarchy which is $a^1_{0\sim T}, b^0_{0\sim1}, c^0_{2\sim T}$ in this case. Then he stores the encoded representation $(E^0_{2:T} = c^0_{2\sim T}; O_1 = o_1; O_3 = o_3)$ together with context cues $(day = 21; loc = home; \ldots)$ in a long term store. The notion of verbatim and gist was already introduced in Section 1.4. In retrieval, Bob uses a cue $day = 21$ to query for relevant episodes from the long term store. Since the mem $O_3 = o_3$ was forgotten Bob retrieves only $E^0_{2:T} = c^0_{2\sim T}; O_1 = o_1$ and he uses these mems to reconstruct the original episode and observations. Note that both perception, encoding and reconstructive retrieval phases use episodic schemata as their input. This is used as a parameter of the probabilistic inference that takes place at these stages. This figure includes graphics from `xkcd.com`.

derlying structure for encoding, storage and episodic retrieval, as exemplified on Figure 3.1.

Our notion of episodic schemata is closely related to scripts or memory organization packets (MOPs) as introduced by Schank (1999). Scripts are sequences of prototypic activities, e.g., there can be one script for commuting and another script for cooking. MOPs are higher level activities constituting of several scripts. Our episodic schemata extend these two structures with probabilistic representation.

The activity recognition module can be realized by a multitude of different approaches as discussed in Section 2.5. In general we need a function that computes probability $P(\text{episodes}|o_{0\sim T}, \text{episodic schemata})$.

Suppose that Bob has identified the hierarchy of episodes describing Alice's behavior. Now we move to the next phase which is *encoding*. In encoding the EM has to decide which details deserve to be remembered. One possible approach to tackle this problem is to remember only the details that are unpredictable given the facts that are already remembered and the schemata. We will illustrate this on the following example. Suppose that every morning Alice commutes by bus, however, this morning she went by car ($c^0_{2\sim T}$ in Fig. 3.1). This will be surprising for Bob since he knows Alice's habits (encoded in the episodic schemata). Therefore Bob will remember this detail. These remembered details are called *mems* in our framework. The second next most surprising detail could be that Alice had to clean a front window from frost (o_3) because it was freezing that morning. To enable this iterative encoding process the model has to be able to:

1. Make predictions of a form $P(\text{episodes}|\text{mems, episodic schemata})$, that is to compute probable episodes hierarchy given the episodic schemata and mems remembered so far.

2. Measure a "difference" between a hypothetical temporary recall $P(\text{episodes}|$ already stored mems, episodic schemata$)$ and a result of the activity recognition which is $P(\text{episodes}|o_{0\sim T}, \text{episodic schemata})$. One possible strategy is that the detail that differs the most will be remembered as a new mem.

There are many functions that quantify difference between two probabilistic distributions. For instance, (Cha, 2007) lists 56 different measures applicable to probabilistic distributions. We will use Kullback-Leibler (KL) divergence (Kullback, 1959) because it was shown that KL divergence matches the human sense for surprise (Itti and Baldi, 2009).

architectures like Soar (Laird, 2012) and not to the meaning of this term as used in psychology where it is connected with low-level motoric learning.

After encoding the observed situation as a list of mems we can focus on the next phase — *storage*. The mems are stored in a long term store (LTS) and they should be indexed by a set of context cues like the day of the week, day of the month, season, location or weather etc. These cues will be used later when Bob tries to recall all memories associated with, e.g., "sunny weekends spent at the cottage". During storage, some of the details may be forgotten (such as o_3 in Figure 3.1). Thus, the list of mems might be modified.

In *retrieval* Bob first has to construct the query and then get the list of associated mems from the LTS. The retrieved mems will be used (together with the episodic schemata) to reconstruct the original episodes and observations using the probabilistic model. This will be done by computing $P(\text{episodes}|\text{retrieved mems, episodic schemata})$.

3.3 Requirements on the Probabilistic Model

In the previous analysis we have identified several requirements on the probabilistic model that will have to be addressed by the proposed EM framework. Our probabilistic model has to be able to compute these probability distributions:

1. In activity recognition — $P(\text{episodes}|o_{0\sim T}, \text{episodic schemata})$
2. In encoding — $P(\text{episodes}|\text{mems, episodic schemata})$
3. In reconstructive retrieval — $P(\text{episodes}|\text{retrieved mems, episodic schemata})$

DBNs fit this schema particularly well since they provide a compact factorized representation of the whole probabilistic distribution. The different types of queries will be performed over the same probabilistic model using the Bayes' rule. The type of the DBN used determines the type of episodic schemata that can be easily encoded in the DBN. It is desirable to find a model with a few parameters that can express the true dynamics well enough. The more parameters the model has, the more expressive it is. However, we should always prefer simpler models to prevent over-fitting when we try to estimate a model's parameters from data. Another consideration is complexity of inference in different models.

The simplest schemata can encode rules like: "when Alice *commutes* and the last action she does was *exit bus* she will probably *transfer to subway*". We can transcribe this as probability $P(O_{t+1} = \text{transfer to subway}|O_t = \text{exit bus}, E^0_t = \text{commute}) = 0.95$. This type of rules can be represented by a simple HMM (Rabiner, 1989). However, to express more complex rules we need an advanced model. For instance, rule: "when Alice drinks a second coffee and she has not drunk water recently she will drink water more probably than drinking a third coffee in a row" could not be easily expressed in the HMM.

3.4 Summary of EM Work Cycle

Here we briefly summarize the main phases of our proposed EM framework:

1. Get episode schemata. Those can be either handcrafted or learned from annotated examples of episodes.
2. Use the model for remembering and retrieving episodes. Remembering and retrieving steps can be interleaved at will.
   (a) Remembering episodes includes:
      i. Encoding an episode instance with respect to the schemata.
      ii. Storing the episode. Forgetting may take place there.

   (b) Retrieve the episode by some search criteria and reconstruct the stored representation to obtain a retrieved episode instance.

Chapter 4

Proposed EM Framework — DyBaNeM

In this chapter we present EM framework DyBaNeM that addresses all the requirements from Section 1.3. DyBaNeM unifies the FTT and probabilistic activity recognition algorithms (reviewed in Section 2.5). The framework also builds on the fact that behavior of NPCs often has hierarchical structure (this point is discussed in Section 2.6).

In this chapter we will first introduce prerequisites of our framework. Then we detail all the phases of the memory's work cycle on a rigor basis. This includes schemata learning, encoding, storage and retrieval together with possible DBN architectures used for probabilistic inference. We will analyze computational complexity of the framework and we will show features enabled by DyBaNeM in human-agent interaction. Afterwards we will discuss other possible uses of the probabilistic activity model.

Examples of the DyBaNeM on a simple toy domain together with a step by step walk through of the encoding process will be discussed in Chapter 5. Experiments on data closer to real world domains are shown in Chapter 6.

4.1 Framework Parts — Prerequisites and Definitions

The core of our framework consists of the probabilistic model, in our case DBN. The episodic schemata are represented by parameters $\hat{\theta}$ of the DBN. The schemata encode statistical regularities of the observed hierarchical behavior; for instance, frequencies of transitions between different goals, goal/subgoal relations and possibly internal progress of each goal. More complex DBN architectures can express more of these regularities. Sequences stored in the EM are encoded as sets of facts — called mems — that represent the sequence the best given the statistics stored in the schemata.

In the rest of this section we will introduce formalisms and algorithms necessary to describe the DyBaNeM framework.

4.1.1 Dynamic Bayesian Network Definition

DBNs are the primary probabilistic model used in DyBaNeM. In this section we will formally define DBN following the definition provided by Murphy (2002).

A DBN is defined by the two-slice temporal Bayes network (2TBN) $B_{\rightarrow}$ and an initial network B_0. Suppose that a system's state at time t is described by a set of random variables $\mathbf{Z}_t$. Then B_0 defines a prior distribution $P(\mathbf{Z}_0)$ at the beginning of the system's evolution. The 2TBN $B_{\rightarrow}$ represents temporal dynamics of the system $P(\mathbf{Z}_t|\mathbf{Z}_{t-1})$ as follows:

$$P(\mathbf{Z}_t|\mathbf{Z}_{t-1}) = \prod_{i=1}^{|\mathbf{Z}_t|} P(Z_t^i|Pa(Z_t^i)) \tag{4.1}$$

where $Z_t^i \in \mathbf{Z_t}$ is the i-th random variable at time t and $Pa(Z_t^i)$ are the parents of Z_t^i in the directed graph representing structure of the DBN (an example of such a graph is in Fig. 4.2).

In the graphical representation, random variables are drawn as circles with a name of the variable inside the circle. The fact that $Z_{t_r}^j \in Pa(Z_{t_s}^i)$ is denoted by a directed arc going from $Z_{t_r}^j$ to $Z_{t_s}^i$. In a first order Markovian model the arcs go either from a slice $t-1$ to a slice t ($t_s = t_r + 1$), those represent temporal influence; or they are within the slice t ($t_s = t_r$), those represent instantaneous effects. The DBN can represent longer temporal dependencies as well. For instance, when an arc goes from $t-2$ to t, we call the model second order Markov model. Higher order models better capture statistical regularities of the domain; however, they are impractical since the number of the model's parameters grows exponentially with the model's order. Thus exponentially more training data is required to obtain accurate estimates of the parameters.

To obtain the joint probability distribution defined by the DBN for a process running for T time steps we *unroll* the 2TBN and get:

$$P(\mathbf{Z}_{1:T}) = \prod_{t=1}^{T} \prod_{i=1}^{|\mathbf{Z_t}|} P(Z_t^i|Pa(Z_t^i)) \tag{4.2}$$

One important assumption made by this definition is that the transition probabilities of the DBN are the same for every time step. This type of process is called a stationary process.

In the context of activity recognition and the DyBaNeM, B_0 can represent probability distribution over beginnings of all possible remembered "days". For instance, a particular agent can wake up 85% of days at home, 10% in a weekend cottage, 4% in a hotel on holiday and 1% is shared by all the other possibilities. Further evolution of the day is captured in the parameters of $B_{\rightarrow}$. $B_{\rightarrow}$ contains information that, for instance, after getting up, 70% of the time the agent brushes his teeth and then has breakfast, in 30% the order is reversed. However, not all DBN topologies can capture such dependencies, limitations of particular topologies will be discussed later.

4.1.2 Virtual Evidence

We will often use the DBN to compute a probability distribution over a set of nodes given some evidence. In general there are two types of evidence: hard evidence and virtual evidence. When we are sure that a random variable X has value a we can use hard evidence to condition the new probability distribution, e.g., $P(\dots|X = a)$. However, it is often the case that even our evidence is uncertain and we have a PMF $P(X)$ over possible values of X. In this case we can use the virtual evidence mechanism (Pearl, 1988; Bilmes, 2004) that allows us to use the whole $P(X)$ as new evidence.

When using virtual evidence for X we add a new node V that has always value 1, that is $P(V = 1) = 1$. Additionally we connect X and V with a CPMF $P(V = 1|X = x) = P(X = x)$. This way, we encode uncertainty about the true state of X in the Bayesian network formalism.

An example of a network extended with a virtual evidence node is shown in Figure 4.1.

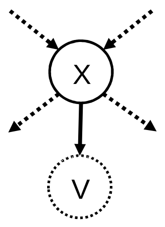


Figure 4.1: An original network represented by a random variable X extended with a virtual evidence node V that provides evidence for X.

4.1.3 Learning parameters of DBN — Learning Schemata

Episodic schemata are represented by parameters $\hat{\theta}$ of a DBN. In our case $\hat{\theta}$ will be fully specified by a set of all CPMFs in the network, that is $\hat{\theta} = \{P(Z_t^i|Pa(Z_t^i))\}$. Expressiveness of schemata depends on the structure of a model at hand. We will discuss this later on an example of three DBN architectures. Two of them will be used in our computational experiments. We will suppose that the DBN's topology is fixed. Thus learning schemata will reduce to well known parameter learning methods. DBN with unobserved nodes, that is, nodes whose values are not in the training data, can be learnt by Expectation-Maximization algorithm (EM algorithm). Topologies without unobserved nodes are learnt by counting the sufficient statistics (Koller and Friedman, 2009). There are also algorithms that can adapt topology of the network, e.g., (Boyen et al., 1999; Abbeel et al., 2006; Campos and Ji, 2011), however, we will not use them in this work. Now we will describe the two well known parameter learning methods.

If $\mathcal{D}$ is a set of observations/learning examples, then learning the model is a task of finding parameters $\hat{\theta}$ such that:

$$\hat{\theta} = \arg\max_{\theta} P(\mathcal{D}|\theta)$$

That is, finding parameters that maximize the probability of observing training data given the parametrized model. Such $\hat{\theta}$ is called maximum likelihood estimate (MLE).

There are two cases of MLE learning, First, $\mathcal{D}$ either contains values for all variables $\mathcal{V}$ contained in the probabilistic model. Second, $\mathcal{V}$ can be divided into a subset of observable variables $\mathcal{O}$ that have values in $\mathcal{D}$ and hidden/unobserved/latent variables $\mathcal{H} = \mathcal{V} \setminus \mathcal{O}$.

In the first case, we have to find *sufficient statistic* for all the variables. If the domain contains only discrete variables, which is our case, this reduces to simply counting the conditional probability table for each $Y \in \mathcal{V}$ given its parents $Pa(Y)$. The set of all these probability tables is the estimated $\hat{\theta}$.

In the second case, EM algorithm (Dempster et al., 1977)[1] can be used. EM algorithm runs in a sequence of alternating steps where it tries to maximize probability of unobserved variables and unknown parameters of the model. In the end, it outputs an estimate of $\hat{\theta}$.

Both types of learning will be used in our work since we will use models with and without latent variables.

4.1.4 DBN Architectures

For computing probabilities, our framework makes it possible to use any DBN architecture that fulfills the following two criteria: some nodes represent observations and some the hidden state of the episode. In this work we use two architectures, simple CHMM (Blaylock and Allen, 2006) and more complex AHM$EM$ (Bui, 2003). Both architectures are shown in Figure 4.2. For explanatory purposes we

[1]Some proofs in this paper were later corrected in (Wu, 1983)

also describe HHMM (Fine et al., 1998; Murphy and Paskin, 2002) which is an intermediate step between these two architectures.

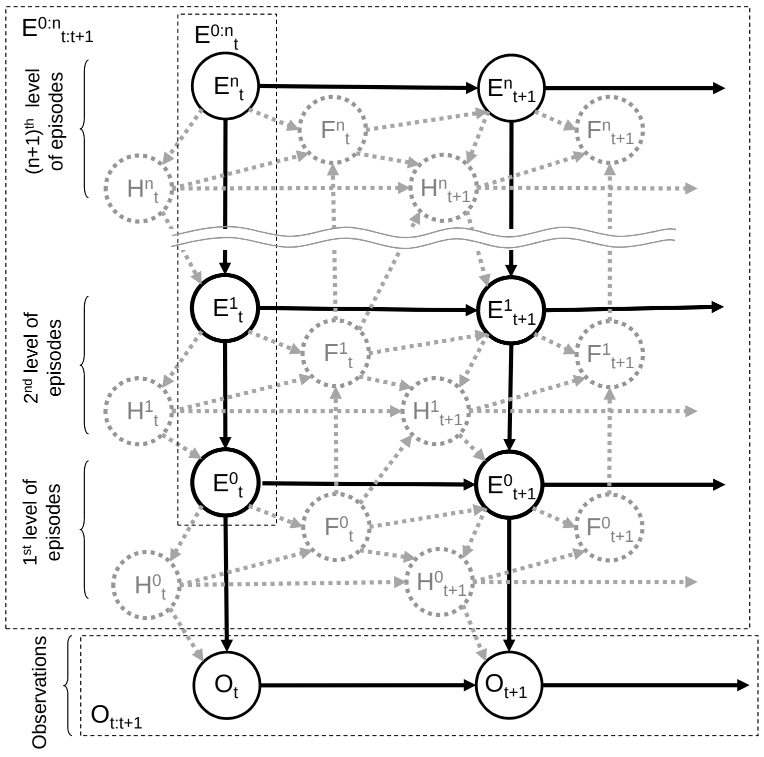


Figure 4.2: An example of a DBN's structure together with our notation. Solid lines show network architecture of CHMM (Blaylock and Allen, 2006). When the dotted drawing is added we obtain network of AHM*EM* (Bui, 2003).

As we said, the schemata are represented by parameter $\hat{\theta}$, that is, by all CPMFs of the DBN's nodes. Expressiveness of the schemata depends on the structure of DBN. In CHMM, episodic schemata encode probability of an episode given previous episodes on the same level in the hierarchy and also given its parent episode. Therefore CHMM is specified by $P(E^i_t|E^i_{t-1}, E^{i+1}_t)$, $P(O_t|E^0_t, O_{t-1})$ and $P(E^i_0)$ for $i \in \{0, \ldots, n\}$. This is one of the possible hierarchical extensions of a well known HMM (Rabiner, 1989). The HMM is a simpler model that assumes only one level of episodes. HMM is specified by $P(E_t|E_{t-1})$, $P(O_t|E_t)$ and $P(E_0)$. When the whole episode hierarchy is known, CHMM is learnt by counting the sufficient statistic.

We can have more detailed schema representation by employing HHMM. In HHMM, each episode can be represented by a probabilistic finite state machine (PFSM) where the states of the PFSM are the episode's sub-episodes. HHMM extends CHMM with notion of terminal states. The schema "knows" that some sub-episodes are the last in a sequence of sub-episodes and when they terminate, the parent episode also ends. This is achieved by extending the model with variables F^i_t and CPMFs $P(F^i_t|F^{i-1}_t, E^i_t, E^{i-1}_t)$ and $P(E^i_t|E^i_{t-1}, E^{i+1}_t, F^i_{t-1}, F^{i+1}_{t-1})$. F^i_t is a variable indicating that some sub-episodes of the PFSM are terminal ($D(F^i_t) = \{terminal, nonterminal\}$). HHMM is better suited for finding borders between episodes than CHMM.

AHMEM is an augmentation of HHMM that extends each episodic schema with a limited internal state (memory) represented by a PFSM with terminal states. This automaton is represented by random variables F^i_t and H^i_t (see Fig. 4.2). This adds even more richness to the represented schemata. For example, AHMEM can encode rules necessary to describe the example with two coffees discussed in Section 3.3. H^i_t represents an internal memory of the episode ($|D(H^i_t)|$ is the number of the PFSM's states). It tracks progress of the episode. The internal state H^i_t of the episode does not have to be observable. However, the transitions of unobserved PFSM acting as the episode's internal memory can be learned by EM algorithm. This architecture makes it possible to correctly recognize behavior generated by a hierarchical PFSM. The downside of AHMEM is that it is computationally more expensive than CHMM. Since AHMEM contains unobservable variables H^i_t, it has to be learnt by EM algorithm.

4.1.5 Formalizing the Episodic Representation, World State, and Inputs/Outputs.

In this section we formalize what representation of episodes and world state is assumed by the DyBaNeM framework.

Let us begin with the definition of a basic memory element called mem.

Definition 1 ***Mem*** *is an assignment* $X^i_{t_1:t_2} = v$. *It says that the agent remembers that* v *was a value of all variables* X_t *where* $t_1 \leq t \leq t_2$. $X_{t_1:t_2}$ *can represent either some level of episode (when* $i > 0$*) or observable environment property at a particular time (when* $i = 0$*).*

An example of the mem can be $E^0_{0:1} = \text{WALK}$ and $O_0 = \text{STEP}$.

Definition 2 ***Episode*** *is a sequence (possibly of length 1) of observations or more fine-grained episodes (sub-episodes) that from a human point of view has a clear beginning and an end.*

Note that episodes may be hierarchically organized. Episode e can be a sub-episode of a higher level episode f. At the same time, e can consist of lower level episodes g and h.

Definition 3 ***Episodic schema*** *is a general pattern specifying how instances of episodes of the same class look.*

In DyBaNeM episodic schemata are encoded in parameters $\hat{\theta}$ of the Bayesian model.

Definition 4 ***Episodic trace*** $\epsilon_t^{0:n}$ *is a tuple* $\langle e_t^0, e_t^1 \dots e_t^n \rangle$ *representing a hierarchy of episodes at time* t*;* e_t^0 *is the lowest level episode at time* t*,* e_t^1 *is its direct parent episode and* e_t^n *is the root episode in the hierarchy of depth* n*. Sometimes we will omit the upper indexes, that is,* $\epsilon_{i:j} \equiv \epsilon_{i:j}^{0:n}$*.*

An example of episodic traces can be $\epsilon_1^{0:1} = \langle WALK, COMMUTE \rangle$ and $\epsilon_2^{0:1} = \langle BUS, COMMUTE \rangle$. The notation of episodic trace reflects the fact that an agent's behavior has often hierarchical nature.

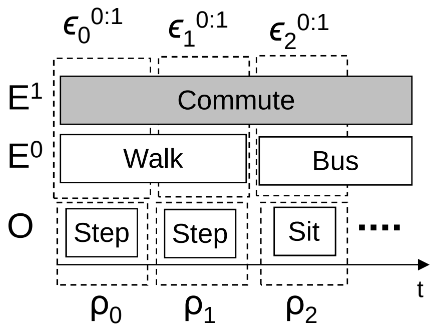


Figure 4.3: An example of episodic traces. The trace for $t = 0$ contains a low level episode $WALK$ and a high level episode $COMMUTE$. The trace in the third time step contains a low level episode BUS and the same high level episode as in the previous step. Observable environment properties are two atomic actions $STEP$ and one action SIT.

One of the key requirements is the ability to handle uncertainty. For this reason we define probabilistic distribution over all possible episodic traces.

Definition 5 ***Probabilistic episodic trace*** $E_t^{0:n}$ *is a tuple of random variables* $\langle E_t^0, E_t^1 \dots E_t^n \rangle$ *corresponding to episodes at different levels of abstraction. A PMF over* $E_t^{0:n}$ *represents an agent's belief about what happened at time* t*. Analogically,* $E_{0:t}^{0:n}$ *denotes probabilistic episodic trace over multiple time steps.*

Probabilistic episodic trace allows us to model uncertainty in both perception and recall. We know that there is only one objectively valid episodic trace for every agent at each time step. However, it is often the case that there are multiple possible explanations of the observed behavior. And even when the initial observation was certain, recall of the event might be uncertain due to forgetting. Therefore our proposed EM framework works mostly with probabilistic representations. For instance, the recall at $t = 1$ for the high level episode might be $COMMUTE$ with high probability, $P(E_1^1 = COMMUTE) \approx 1$. However, it might be unclear whether the lower level episode was BUS or $SUBWAY$ since probability of both possibilities is nearly the same, e.g., $P(E_1^0 = BUS) = 0.4$ and $P(E_1^0 = SUBWAY) = 0.38$.

The following data structure represents an agent's true perception of the environment's state.

Definition 6 *Let ρ_t denotes **observable environmental properties** at time t.*

For instance, ρ_t can hold atomic actions executed by an observed agent, e.g., $\rho_1 = STEP$, $\rho_2 = SIT$. An example of two episodic traces and corresponding observable properties is depicted in Fig. 4.3.

More complex structure of the observed environment's properties is also possible, ρ_t can be a tuple encoding an agent's action together with an object that may be a resource of the action. For instance, $\rho_1 = \langle SIT, seat\_7 \rangle$ where $seat\_7$ represents a particular instance of a seat. Further details can be added to ρ as needed. For example, a different extension can be $\langle SIT, seat\_7, bus\_22 \rangle$.

Analogically to $E_t^{0:n}$ and $\epsilon_t^{0:n}$, O_t is a random variable representing belief about observation ρ_t.

Definition 7 ***Probabilistic observable environment properties** O_t is a random variable representing observations at time t. A PMF over O_t represents an agent's belief about what happened at time t. Analogically, $O_{0:t}$ denotes probabilistic observable environmental properties over multiple time steps.*

Figure 4.2 shows how these definitions translate to CHMM and AHM*EM*.

4.1.6 Quantifying Difference Between Two Distributions — KL Divergence

In encoding, the framework works with quantity, measuring difference between the expected state of a random variable given observations and its expected state given the remembered facts. We call this quantity surprise. In Bayesian framework, surprise can be defined as "difference" between prior and posterior probability distributions. We adopt the approach of (Storck et al., 1995; Itti and Baldi, 2009) who propose to use KL divergence (Kullback, 1959) to measure surprise.

Definition 8 ***KL divergence*** *of two PMFs $P(X)$ and $P(Y)$ (Kullback, 1959), where $D(X) = D(Y)$ is defined as:*

$$KL(P(X) \rightarrow P(Y)) = \sum_{x \in D(X)} P(X = x) ln \frac{P(X = x)}{P(Y = x)}$$

We use notation with $\rightarrow$ to stress directionality of KL divergence; note that it is not symmetrical. We will use KL divergence as a core tool of our framework.

4.1.7 Smoothing Probabilities

To avoid cases where probability of some outcome is zero we use convex smoothing on all PMFs. This way even events with zero probability are not impossible, they are only highly unlikely. We achieve this by replacing the original PMF $P(X = x)$ with distribution P' defined as:

Definition 9 *Convex smoothing P' of a PMF P is defined as:*

$$P'(X = x) = Smooth(P) = \alpha \cdot U_{|D(X)|} + (1 - \alpha) \cdot P(X = x)$$

where $\alpha \in (0, 1\rangle$ is the smoothing parameter and $U_{|D(X)|} = \frac{1}{|D(X)|}$ is a discrete uniform distribution over $|D(X)|$ elements.

Thus, the effect of smoothing is that the new PMF P' is closer to uniform distribution than the original P. The amount of smoothing is controlled by α. Note that $\forall x : P'(X = x) > 0$.

4.1.8 Applying General Parameters Learning — Learning Schemata

In our case, examples of episodes that we want to use for schemata learning will be denoted by $\mathcal{D} = \{d_1, d_2 \dots d_n\}$, where each d_i can be one day of an agent's life, or any other appropriate time window. d_i itself is a sequence of time equidistant examples c_t, that is, $d_i = \{c_0^i, c_1^i \dots c_{t_i}^i\}$. Each c_t^i is a tuple $\langle \epsilon_t^{0:n}, \rho_t \rangle$; it contains the episodic trace and the observable state of the environment.

When we use CHMM which contains only E_t^i and O_t variables the model can be learnt by counting the sufficient statistics since d_j contains values of all the model's variables. For learning AHMEM we can derive value of variables F_t^i from d_j, however, the state of H_t^i variables tracking internal progress of episodes is unobservable. Therefore we have to use the EM algorithm.

4.2 Encoding

The encoding algorithm computes a list of *mems* on the basis of the agent's perception, $Per_{0:T}$, of the situation to be remembered. Here we assume that the agent observed $T+1$ consecutive time steps of the situation.

Definition 10 ***Agent's perception*** *$Per_{0:T}$ is a set of PMFs such that $Per_{0:T} = \{f_X : X \in Observable\}$, where f_X is PMF for each variable X of interest.*

Concerning variables that are observable for the agent, we have to distinguish whether the agent is going to remember its own activity or activity of another agent. This corresponds to the requirements a) and b) from Section 1.3:

1. $Observable = O_{0:T}$ — Bob is going to encode Alice's activity whose $\epsilon_{Alice,0:T}$ is hidden to Bob, nevertheless Bob perceives Alice's atomic actions that are contained in $\rho_{Alice,0:T}$. This is the use-case described in Figure 3.1. We introduce an observation uncertainty by defining $\forall t \in \{0,T\} : f_{O_t}(x) \equiv Smooth(\mathbb{1}(\rho_{Alice,t}, x))$, where $Smooth$ is the function from Definition 9 and $\mathbb{1}$ is the identity function[2]. In this case $P(E^i_t)$ will be deduced within the encoding algorithm. Realization of the activity recognition phase for this case is shown in Fig. 4.4.

2. $Observable = E^{0:n}_{0:T} \cup O_{0:T}$ — Bob is going to encode his own activity, in this case the episodic trace $\epsilon_{Bob,0:T}$ is available to Bob since he knows what he wanted to do (Bob can introspect its his own DMS). Values of f_{O_t} are computed as above and $\forall i \in \{0,n\}, \forall t \in \{0 : T\} : f_{E^i_t}(x) \equiv Smooth(\mathbb{1}(\epsilon^i_{Bob,t}, x))$.

Whenever the evidence is described by a whole PMF instead of only the most probable state, we use the virtual evidence mechanism discussed in Section 4.1.2.

Algorithm 1 is a skeleton of the encoding procedure. The input of the algorithm is $Per_{0:T}$, where the time window $0 : T$ is arbitrary. In our work we use a time window of one day. The output is a list of mems encoding this interval with respect to the episodic schemata.

In each cycle, the $GetMemVar$ function returns the variables range $X^i_{t_1:t_2}$ that will be remembered. It is required that all variables $X^i_t : t_1 \leq t \leq t_2$ in this range have the same most probable value. For instance, when there are two

[2] We define the identity function in a standard way, that is:

$$\mathbb{1}(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

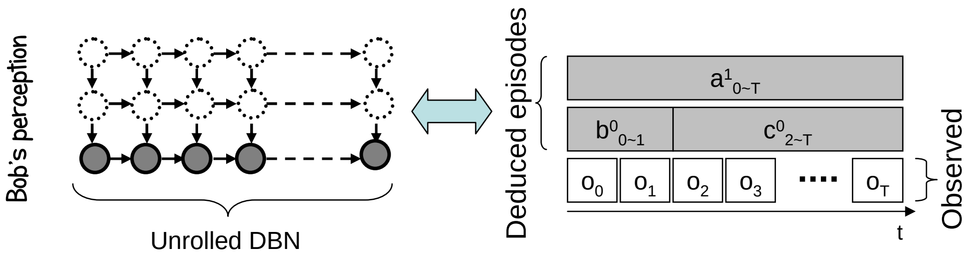


Figure 4.4: Activity recognition/perception phase as implemented by the DBN. This figure is a continuation of the example from Fig. 3.1. Solid circles denote evidence variables whereas dotted circles are the query variables.

Algorithm 1 General schema of encoding algorithm

```
Require: $Per_{0:T}$ — PMFs representing the agent's perception of the situation (i.e. smoothed observations)
Require: $\mathcal{M}$ — probabilistic model representing learned schemata
1: procedure ENCODING($Per_{0:T}, \mathcal{M}$)
2:     $mems \leftarrow empty$                                  ▷ List of mems is empty
3:     while $EncodingIsNotGoodEnough$ do
4:         $X^i_{t_1:t_2} \leftarrow GetMemVar(\mathcal{M}, Per_{0:T}, mems)$
5:         $x_{max} \leftarrow MLO_{P_\mathcal{M}}(X^i_{t_1}|mems)$          ▷ Find the most probable value
6:         $mems.add(X^i_{t_1:t_2} = x_{max})$                 ▷ Remember the assignment
7:     end while
8:     return $mems$
9: end procedure
```

consecutive time steps, where the high level episode does not change, this range can be remembered in a single mem. As an example consider variables E^0_0 and E^0_1 from Figure 4.3. They both have the same value $WALK$, therefore we can remember this whole episode in a single mem $E^0_{0:1} = WALK$.

Inside our algorithms we often need to compute the most probable value of a random variable. For this purpose we define the following standard function:

Definition 11 *The* ***MLO function (most likely outcome)*** *is defined as:*

$$MLO_{P_\mathcal{M}}(X|evidence) \equiv \arg\max_{x \in D(X)} P_\mathcal{M}(X = x|evidence) \tag{4.3}$$

MLO is used in step 5 of the encoding algorithm where we get the most probable value for the previously identified candidate mem and we add this assignment to the list of mems. In the end the procedure returns the list of mems.

We have developed two variants of the $GetMemVar$ function, each has its justification. The key idea of both variants is to measure the difference between the result of the activity recognition and the recall with mems computed so far. To measure the difference for a single random variable Y, we introduce the following auxiliary function:

$$Diff_{\mathcal{M}}(Y, Per_{0:T}, mems) = KL\left(\underbrace{P_{\mathcal{M}}(Y|Per_{0:T})}_{\text{Activity recognition}} \rightarrow \underbrace{P_{\mathcal{M}}(Y|mems)}_{\text{Recall given mems}}\right) \tag{4.4}$$

where $P_{\mathcal{M}}(Y|Per_{0:T}) \equiv P_{\mathcal{M}}(Y|X = f_X : f_X \in Per_{0:T})$. In other words we condition the probability on all observations via the virtual evidence mechanism.

In the first variant of the encoding algorithm, the idea is to look for a variable whose observed PMF and PMF in the constructed memory differs the most. This variable has the highest surprise as defined by KL divergence and hence it should be useful to remember it. This strategy will be called *retrospective maximum surprise* (RMaxS). It is retrospective since it assumes that the agent has all observations in a short term memory store and, for instance, at the end of the day, he retrospectively encodes the whole experience. RMaxS strategy can be formalized as:

$$X \leftarrow \operatorname*{arg\,max}_{Y \in \mathbf{VOI}} Diff_{\mathcal{M}}(Y, Per_{0:T}, mems) \tag{4.5}$$

where $\mathbf{VOI} \subseteq \mathcal{V}$ is a set of random *variables of interest* whose value can be remembered by the model. There can be some variables in the DBN that we do not want to remember since they are hardly interpretable for humans (for instance, H_t^i from AHMEM that represents internal progress of activities). In our implementation we used $\mathbf{VOI} = E_{0:T}^{0:n} \cup O_{0:T}$.

The alternative strategy called *retrospective minimum overall surprise* (RMinOS) assumes a more sophisticated process. RMinOS picks the variable–value pair whose knowledge minimizes sum of surprises from the original state of each $Z \in \mathcal{V}$ to its recalled state. The following equation captures this idea:

$$X \leftarrow \operatorname*{arg\,min}_{Y \in \mathbf{VOI}} \sum_{Z \in \mathcal{V}} Diff_{\mathcal{M}}(Z, Per_{0:T}, mems \cup \underbrace{\langle Y = \bar{y} \rangle}_{\text{potential mem}}) \tag{4.6}$$

where $\bar{y} \equiv MLO_{P_{\mathcal{M}}}(Y|mems)$. After substituting for the $Diff$ function the equation translates to:

$$X \leftarrow \operatorname*{arg\,min}_{Y \in \mathbf{VOI}} \sum_{Z \in \mathcal{V}} KL\left(P_{\mathcal{M}}(Z|Per_{0:T}) \rightarrow P_{\mathcal{M}}(Z|Y = \bar{y}, mems)\right)$$

The idea is to add the most probable outcome of each variable of interest to a list of mems and test which assignment would minimize difference on all other variables. A third alternative approach might be not to minimize the sum of KL divergences but to minimize KL divergence of the full joint probability distribution over all possible episode "trajectories", that is, to compute KL divergence from $P(E^{0:n}_{0:T}, O_{0:T}|Per_{0:T})$ to $P(E^{0:n}_{0:T}, O_{0:T}|Y = \bar{y}, mems)$; however, that would require computing the whole joint probability. Thus, we use summation of surprise in each variable instead. The downside of our approach is that it only approximates KL divergence of the whole joint probability.

Note that we remember only the most probable value, including the time index, in the *mems* list instead of the whole distribution. For instance, mem $E^0_{0:1} = WALK$ stores information that for the first two time steps the episode was walking. No other alternatives are explicitly stored even though the result of activity recognition was uncertain. Storing only one possible value in each mem helps to make mems clearly interpretable. However, full distributions can be used in the mems as well, e.g., $E^0_{0:1} = \langle (WALK, 0.7), (RUN, 0.2), (rest, 0.1) \rangle$

The algorithm runs in a loop that terminates once the *EncodingIsNotGoodEnough* function is false. There are more possible terminating conditions. For instance, we can model limited memory capacity for each day by one of the following rules. The first rule enforces a constant number of mems for each day, that is:

$$|mems| < K \tag{4.7}$$

The second rule quantifies the cumulated difference between the agent's initial observation and its reconstructed memory for those events. One way to formalize this criterion can be:

$$\sum_{X \in Observable} Diff(X, Per_{0:T}, mems) < Diff_{max} \tag{4.8}$$

That is, the accumulated difference on all variables should be below a fixed threshold $Diff_{max}$.

The first rule (Eq. 4.7) leads to encoding where every day is represented by exactly the same number of mems. The second criterion allows for a variable number of mems. If the day is close to episodic schema (an average day), remembering only a few mems would be sufficient to describe it well enough. On the other hand, unusual days will require more mems since knowledge from schemata will be useless. In our experiments (see Chapters 5 and 6) we use the first rule from Eq. 4.7.

4.3 Storage and Forgetting

The list of mems for a particular day is stored into the LTS and it is indexed to facilitate subsequent retrievals. Indexes might be seen as metadata that are useful when searching for the right episode. An index can be the day of the week, agents present in the scene, weather etc. The set of indexes is to be determined by the developer and it should follow the needs of retrieval.

During storage, the mems can undergo optional time decayed forgetting. The relation between an age of a remembered event, its initial strength and probability of its recall is often called a *forgetting curve*. Over time several functions have been proposed as a mathematical model for the forgetting curve. These include exponential, Pareto function, logarithm or power function, see (Averell and Heathcote, 2011) for a review of these approaches and comparison of selected functions on empirical data.

Here we will use the exponential function to model the forgetting curve as proposed by Anderson (1983). The following equation shows a relation between an age t of the mem m_j, its initial strength S and its retention R (e is Euler's number):

$$R(m_j) = e^{-\frac{t}{S}} \tag{4.9}$$

We want to model so that interesting events are remembered for a longer period than less interesting events. In order to achieve this the initial strength S of a mem $m_j = \langle X_t^i = x_{max} \rangle$ is related to the strength of the mem in encoding. This reflects the fact that mems deviating from an average day will have higher initial strength than the mems that follow the average schema. Therefore the strength of the mem is equal to the value of KL divergence in Eq. 4.5 in case of the RMaxS, that is:

$$S = Diff_{\mathcal{M}}(X_t^i, Per_{0:T}, mems_{j-1}),$$

where $mems_{j-1}$ is a set of mems stored before computation of mem m_j. When one uses the RMinOS the strength is equal to the value of the sum in Eq. 4.6, that is:

$$S = \sum_{Z \in \mathcal{V}} Diff(Z, Per_{0:T}, mems_{j-1} \cup \langle X_t^i = x_{max} \rangle)$$

Note that more complex model should extend the initial strength with other terms, since it is known that retention depends also on other factors. For instance, it makes a difference when one was instructed that he will not be asked to recall this memory in future (van Dongen et al., 2012).

Once $R(m_j)$ decreases under the threshold β_{forget}, m_j will be deleted from the list of mems and it will not contribute to the memory recall. The shape of the forgetting curve is shown in Fig. 4.5.

Of course, there can be alternative implementations of the forgetting mechanism. For instance, mems does not have to be completely deleted but their contribution to recall can decrease with time. This can be implemented by mixing the value stored in a mem with prior (possibly uniform) distribution for a

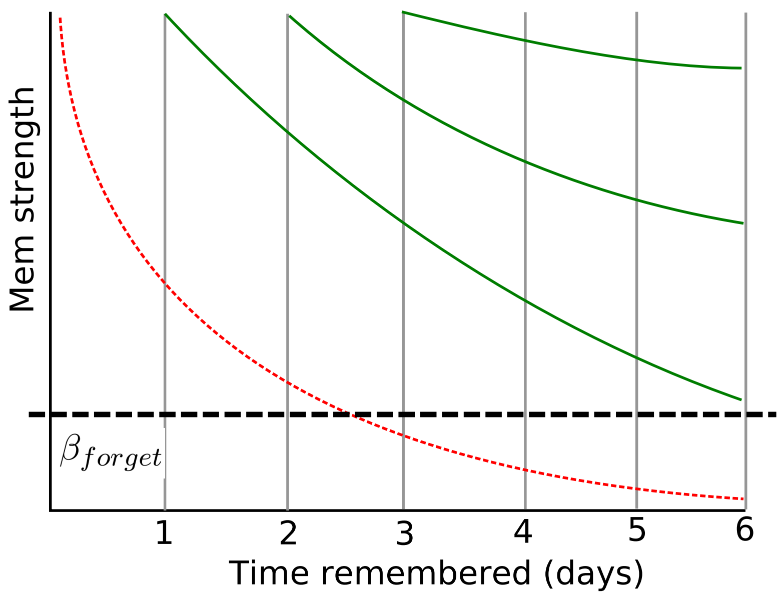


Figure 4.5: Forgetting curves for various initial strengths of mems. Suppose that the encoding algorithm computed a list of four mems. The smaller the strength of a mem the steeper the curve. Thus the least interesting mem has the steepest curve, here drawn in dotted red. When the retention of this mem decreases below β_{forget} between the second and the third day it will be forgotten. The other three mems will be still available for recall. This figure is a modified version of an image (Wikipedia, 2015b) released into the public domain.

random variable on that level. Thus the information stored in a mem will continually fade towards the prior.

4.4 Retrieval

Retrieval in DyBaNeM is a two step process of, first, querying LTS for relevant mems, second, reconstructing the remembered sequence by combining the schemata with mems. We developed Algorithm 2 which shows this straightforward process.

We simply obtain the list of mems for search cue k (which can be, e.g., a time index $day = 21$, as in Fig. 3.1, or $weather = sunny$). For this purpose one can use any database engine. Then we use assignments in the mems list as evidence for the probabilistic model. The resulting PMFs for all variables of interest are returned as a reconstructed memory for the cue k. Use of the DBN for this task is shown in Fig. 4.6.

There are multiple options how a probabilistic model can be used to obtain

Algorithm 2 Retrieval

```
Require: k — cue for obtaining the episode
1: procedure RETRIEVAL($k, \mathcal{M}$)
2:     $mems \leftarrow getMemsFor(k)$        ▷ List of mems associated with the cue
3:     return $\{P_{\mathcal{M}}(Y|mems) : Y \in \mathbf{VOI}\}$
4: end procedure
```

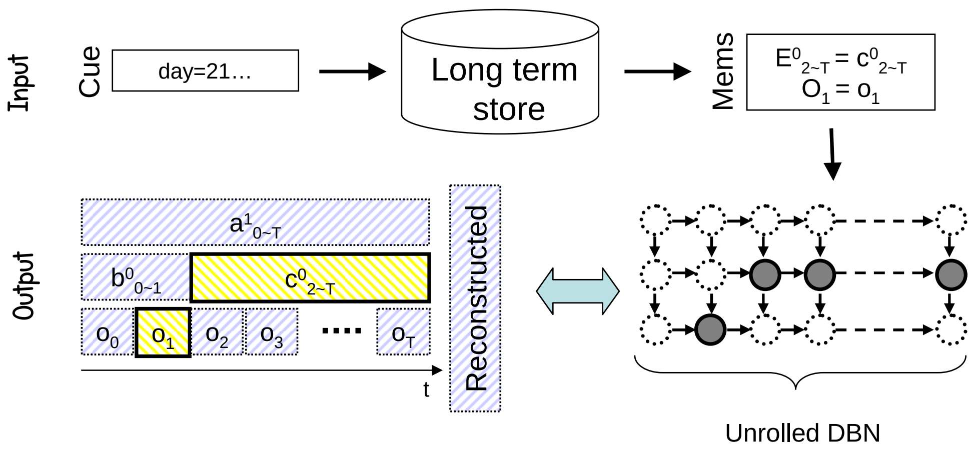


Figure 4.6: (colour figure) Reconstructive retrieval phase as implemented in DBN. This is a continuation of the example from Fig. 3.1. Solid circles denote evidence variables, those that have values from the mems, whereas dotted circles are the query variables. In the final output blue dashed boxes represent reconstructed episodes, whereas the yellow solid boxes are the episodes remembered in detail.

the reconstructed episodes. One way, used in Algorithm 2, is to use marginal probabilities for every random variable representing observation or episode as an answer. Thus the answer is a set of distributions over every variable Y from **VOI**. Marginal probabilities summarize belief for a single random variable in isolation. However, this only provides a compressed version of the whole information computed by the probabilistic model.

The other option is to use full joint probabilistic distribution over multiple time steps. However, when one takes into account multiple time steps, things get more complex. For instance, in general $P(O_t) \times P(O_{t+1}) \neq P(O_t, O_{t+1})$, this would hold only if O_t and O_{t+1} were conditionally independent. Thus the most probable states of O_t and O_{t+1} might be a and b respectively ($a = \arg\max_{x \in D(O_t)} P(O_t = x)$ and $b = \arg\max_{x \in D(O_{t+1})} P(O_{t+1} = x)$), but the most probable tuple of states in the joint probability distribution over O_t, O_{t+1}, that is $c, d = \arg\max_{x,y \in D(O_t) \times D(O_{t+1})} P(O_t = x, O_{t+1} = y)$, might be different. Therefore the alternative recall procedure could return probability distribution over the whole joint probability. In general this approach will be more accurate, however, it will also require much more time and space.

Suppose that we have a sequence of N actions and at every time step there are M possible actions. This results in M^N possible sequences, thus the full joint probability has to be defined for the whole domain of M^N sequences. On the other hand when one needs to output only marginal probabilities for each time step then only $M \cdot N$ values are needed. That is, marginal probabilities can be used to make a compressed summary of possible sequences. The downside is that marginal probabilities may be misleading in a way that there are no guarantees on their connection to the most probable sequences. This follows from the previously mentioned fact that $P(O_t) \times P(O_{t+1}) \neq P(O_t, O_{t+1})$.

Outputting the whole joint distribution whose size is exponential to the length on the sequence is clearly impractical. One way to overcome this limitation might be to use only the most probable sequence of states as an answer. In Bayesian inference this is called maximum a posteriori (MAP) query. MAP is defined as:

$$MAP(\mathbf{X}, \mathbf{E} = \mathbf{e}) = \underset{\mathbf{x} \in \prod_{\mathbf{Y} \in \mathbf{X}} \mathbf{D}(\mathbf{Y})}{\arg\max} P(\mathbf{X} = \mathbf{x}, \mathbf{E} = \mathbf{e}) \tag{4.10}$$

where $\mathbf{X}$ is a set of query variables, $\mathbf{E}$ is a set of evidence variables, $\mathbf{x}$ is a vector of states for variables in $\mathbf{X}$ and $\mathbf{e}$ is the value of evidence variables.

MAP is often called a point estimate, this refers to the fact that its result is only one individual assignment whereas bayesian queries usually output probability distribution. The disadvantage of MAP is that it does not say anything about the probability distribution of possible sequences. It may happen that the most probable sequence seq_1 has a probability 5% and there are nine sequences $seq_2, \ldots, seq_{10}$ that are quite similar among themselves (they differ in only one action) and dissimilar with seq_1 but their summed probability is 30%. In this case outputting only the most probable sequence would result in loss of important information.

Therefore in the current version we use marginal probabilities and we leave the other options as possible future work.

4.5 Summary of The Main Phases

Table 4.1 summarizes the main phases of DyBaNeM's working cycle. It shows high level perspective of episodic schemata learning, perception, encoding, storage and retrieval as implemented in DyBaNeM.

Phase	Implementation
Schemata learning	Estimate DBN parameters $\hat{\theta}$ from annotated examples of episodes via counting the sufficient statistic or using the EM algorithm.
Activity recognition/perception	In a case when the agent observes another agent use observed events as evidence for the DBN and perform inference — $P(episodes\ hierarchy \mid observation)$. This is the activity recognition step that infers higher level behavior unseen in the environment. When the agent wants to remember its own actions it may use introspection to obtain the episode hierarchy.
Encoding	Repeatedly find the most surprising event from the encoded sequence and remember it as a mem. The mem is a tuple ⟨time range, level of abstraction, event⟩. KL divergence is used to measure degree of surprise from $P(episodes \mid observation)$ to $P(episodes \mid mems\ stored\ so\ far)$.
Storage	Store sequence of mems into an LTS and index them by features important for later retrieval. Optionally erase mems based on the forgetting Equation 4.9.
Retrieval	Query the mems from the LTS and use them as evidence for the DBN. Perform inference in the DBN and use the result as reconstructed memory — $P(episodes\ hierarchy \mid retrieved\ mems)$.

Table 4.1: Summary of the main phases of DyBaNeM's working cycle.

4.6 Complexity Analysis

Complexity of DyBaNeM's phases will be inspected in this section.

DyBaNeM makes extensive use of the probabilistic model, represented here as DBN. In general exact inference in BNs is NP-hard (Cooper, 1990). Even approximate inference is NP-hard (Dagum and Luby, 1993). On the other hand, inference in some special topologies can be performed in linear time (Pearl, 1986). Also the approximate inference can be performed in polynomial time when there are restrictions on conditional probabilities in the network (Dagum and Luby, 1997).

Schemata learning. Learning the schemata takes time linear in the number of example sequences M (e.g., days), i.e. $O(M)$. In the case of DBN models with fully observable random variables, like CHMM, learnt by counting the sufficient statistics, the complexity is $O(MN)$ where N is length of a single sequence. More complex models with hidden variables are learnt by EM algorithm, which involves inference step, and is NP-hard in general case.

Encoding. Encoding involves repeated inferences over the DBN that may be NP-hard, i.e. when an exact inference algorithm is used, the time complexity is $O(exp(N))$ for one inference step. The number of inference steps is $O(|mems|)$.

Storage. Storing mems in a database like LTS usually can be done in $O(log(K))$, where K is the number of already stored memories. Logarithmical time is needed to update database indexes.

Retrieval. Querying mems from LTS again takes $O(log(K))$. The reconstructive phase involves inference over DBN unrolled to N slices, thus time complexity using exact inference algorithm is $O(exp(N))$ for one query.

When DyBaNeM has to be used in a real time system like IVA then it might be worth considering algorithms for real time inference (Guo, 2002). A recent example of an anytime algorithm used in activity recognition is shown by Kabanza et al. (2013). In our computational experiments we use an exact inference clustering algorithm (Huang and Darwiche, 1996).

4.7 How DyBaNeM Supports Rich Dialogs With IVAs

Now we will show how the functions listed in Section 1.3 and analyzed in Chapter 3 are enabled by DyBaNeM. We will use an example dialog where a human user interviews IVA Bob about his observation of IVA Alice.

1. High level summarization is enabled by the hierarchical nature of DyBaNeM. The recalled sequences contain not only atomic actions but also high level episodes that can be used as summarization of the sequence of actions. Thus if Bob has DyBaNeM with two levels of abstraction, the values reconstructed by the bayesian inference on the highest level can be used to provide a summary of the day. Fig. 4.7 shows an example of such a situation.

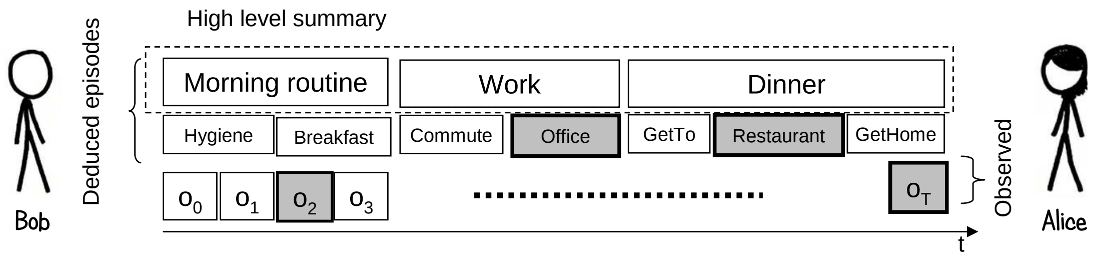


Figure 4.7: Summarization example. Bob is a detective who monitors Alice. First, let Bob observe Alice's atomic actions $o_{0:T}$. Second, Bob can deduce the higher level episodes from this observation. Third, Bob encodes the whole day by the most salient events — these are the mems computed in the encoding algorithm. Mems are marked as the shaded boxes. When Bob is asked to summarize what Alice did yesterday he recalls the mems and reconstructs the rest with the use of the episodic schemata. In the end he responds by episodes in the highest level: "Morning routine, work and dinner." This figure includes graphics from `xkcd.com`.

2. Possibility of further clarifying questions is another useful feature of the hierarchical memory organization. When the user asks for details of an episode, Bob can reply by its sub-episodes as illustrated in Fig. 4.8a.

3. Expressing degree of certainty for recalled events is enabled by the probabilistic nature of the framework. Each action/episode is represented by at least one random variable in the DBN. During reconstructive recall we obtain a probability mass function (PMF) for each variable that encodes probability of every action/episode at this point in time. When the probability of the most probable outcome dominates the other outcomes, we can say that the IVA is sure. However if there are two competing alternatives, the IVA can reflect this in the dialog. See Fig. 4.8b for an example.

4. Believable mistakes in recall can emerge as interplay of forgetting and reconstructive retrieval. When only a few mems remain stored then during the recall the forgotten events are reconstructed from the episodic schema. It can happen that the schema predicts an event that did not actually happen but it fits well to the way the episode usually unfolds. A different approach to this so called false memories phenomenon is described in (Čermák et al., 2011). Continuing in the example from Fig. 4.8, it may be the case that Alice used *Public transport* that day, but Bob does not remember this in a mem and his schema favors other options.

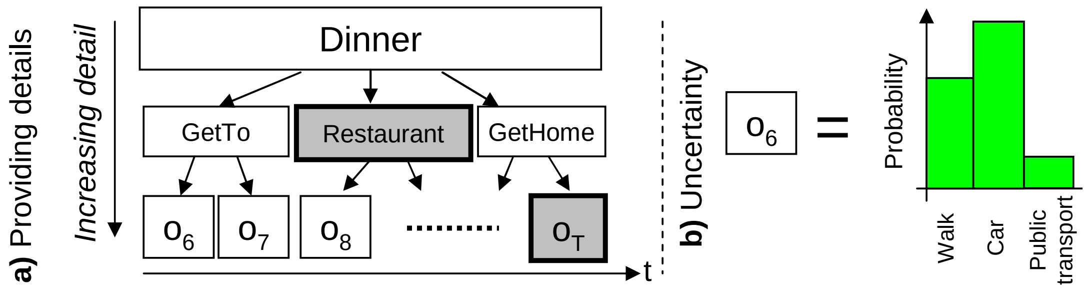


Figure 4.8: **a)** When Bob is asked to say more about Alice's dinner, he will reply: "She left from work and went to the restaurant, she ate there and then she went back home." Shaded boxes represent mems, white represent reconstructed events. **b)** a further question can be: "How did she get to the restaurant?" which asks about recall of atomic actions represented by observations o_6 and o_7. In the case of o_6 the associated PMF computed in recall assigns similar probability to both $Walk$ and Car. Thus Bob is not sure and he can reflect this in his answer: "She went by car or she just walked, I am not sure, sorry." In the case of o_7 Bob might have much higher confidence for walking since the restaurant is in the middle of a park and you always have to walk there.

5. Measuring interestingness of events can be achieved by comparing the actual events to prediction from the schema. Imagine that 95 percent of days start by a sequence: *Get up, Brush teeth, Have a shower, Have breakfast.* If the schema is expressive enough to capture this sequence, those events will become completely uninteresting. They are predictable, thus they do not distinguish one day from other. However, meeting foreign soldiers marching through one's home town is much less probable. Thus it is the event that deserves more attention in the dialog than brushing teeth every morning again and again.

4.7.1 Implementation of the Dialog Supporting Features

Now we show how DyBaNeM's dialog supporting features can be implemented in greater detail.

1. High level summarization and 2. Further clarifying questions are possible because of the hierarchical structure of DBN used in both encoding and retrieval. Values of variables $E^n_{0:T}$ (see Fig. 4.2) can be used for summarization. If the user asks for details of time interval $\langle t_1, t_2 \rangle$, values of $E^{n-1}_{t_1:t_2}$ can be used to construct the answer (or $O_{t_1:t_2}$ when $n = 0$).

3. Expressing degree of certainty of recall is implemented by computing entropy of random variables corresponding to the action/episode. Entropy is a standard measure that can quantify uncertainty over the whole probability distribution. Entropy $H(X)$ of a random variable X is defined as:

$$H(X) = -\sum_{x \in D(X)} P(x) \cdot log_{|D(X)|} P(x) \tag{4.11}$$

The higher the entropy is, the more uniform the PMF over X is. Thus there is more uncertainty since all outcomes of X seem similarly probable. On the other hand when entropy is close to zero there is only a little uncertainty about X's value.

4. Believable mistakes in recall result from forgetting and the inference process in retrieval. It can happen that there was an action a at time t' and during storage the mem for t' was forgotten. Later in retrieval, that is when computing PMF $f_{t'} = P_{\mathcal{M}}(O_{t'}|mems)$, the value had to be deduced from remembered mems and the probabilistic model $\mathcal{M}$ that includes the episodic schemata. If action b is more probable under this assumption $(P_{\mathcal{M}}(O_{t'} = b|mems) > P_{\mathcal{M}}(O_{t'} = a|mems))$, b will be recalled instead of a. There is no specific process for this feature, it is DyBaNeM's emergent property.

5. Interestingness of events is measured by KL divergence in the same way as it is done by the encoding algorithm. The more different a PMF predicted by the schemata from the recalled PMF is the higher the value of KL divergence is. The first mem picked by the encoding algorithm is the one that deviates most from the prediction from schema. Subsequent mems contain less and less information. Thus if an IVA wants to communicate the interesting events first it can start with the first recalled mem (representing the most salient deviation from the schema) followed by the second and so on. If both the IVA and the human player have the same episodic schemata they will be able to reconstruct the same episodes.

4.8 Other Uses of Probabilistic Model

The probabilistic model $\mathcal{M}$ with parameters $\hat{\theta}$ captures regularities of the agent's behavior. Thus it can be reused also for other functions besides the EM model, namely for:

1. estimating behavior/perceptual surprise, and
2. determining behavior/perceptual uncertainty

By behavior surprise we mean surprise of an observer over high level intentions of the observed agent. Suppose that Bob observes Alice. Based on the observation up to the time $t-1$, he thinks that Alice will pursue a goal g_1 in time t but after observing her actions in time t it turns out that g_2 is now more probable than g_1.

For related work on the topic of surprise modeling in the field of artificial agents see (Macedo and Cardoso, 2001). Comparison of several functions computing surprise and their fit to human data is presented in (Macedo et al., 2004).

These works used several probability measures but they omitted the KL divergence that was shown to provide a good model for human data on a visual attention task (Itti and Baldi, 2009). Since KL divergence fits well in our probabilistic framework we decided to use it also for surprise computation.

Definition 12 *We define* ***current behavior surprise*** $S(X_t)$ *of a variable* X_t *as a KL divergence from prior to posterior probability of* $P(X_t)$*, that is:*

$$S(X_t) = KL\left(P(X_t|o_{0:t-1}) \to P(X_t|o_{0:t})\right) \tag{4.12}$$

When $X_t = E_t^i$ *we measure* ***current episode surprise*** *on the* i*-th level in time* t*. When* $X_t = O_t$ *we measure* ***current observation surprise*** *associated with the time step* t*.*

Analogically, we can measure how sure Bob is about Alice's current intention, a quantity we call current behavior uncertainty. For instance, there can be two goals that have almost the same probability, thus Bob can be confused about Alice's true intention. To measure a degree of uncertainty, we use an entropy. The entropy is a natural choice for this measure. If there are more possible explanations then the entropy will be high, whereas if only one intention seems to be the right one, the entropy is low. The entropy $H(X)$ of random variable X is defined in equation 4.11.

The more uniform a distribution of X the higher the entropy; on the other hand, $H(X) = 0$ if X has only one possible outcome. Thus there is no uncertainty over its value.

Behavior uncertainty is formalized in the following definition.

Definition 13 ***Current behavior uncertainty*** $U(X_t)$ *in time* t *is entropy of* X_t *conditioned on* $o_{0:t}$*, that is:*

$$U(X_t) = H\left(P(X_t|o_{0:t})\right) \tag{4.13}$$

We will refer to $U(E_t^i)$ *as* ***current episode uncertainty****,* $U(O_t)$ *will be* ***current observation uncertainty****.*

Previous definitions referred to the current time t: We measured how observation obtained at time t influences uncertainty and surprise for time t. However, we can extend this approach and ask how observation in time t changed the expected behavior at time u where u can be either before or after t. If $u < t$, we can ask how this new observation made Bob reinterpret Alice's past intentions. When $u > t$ we measure how the new observation changed Bob's prediction of the future.

Definition 14 *We define* ***extended behavior surprise*** $\bar{S}(X_u, t)$ *as:*

$$\bar{S}(X_u, t) = KL\left(P(X_u|o_{0:t-1}) \to P(X_u|o_{0:t})\right) \tag{4.14}$$

$\bar{S}(X_u, t)$ measures how observation at time t influences the agent's belief about behavior at time u. We will refer to $\bar{S}(E^i_u, t)$ as ***extended episode surprise****, $\bar{S}(O_u, t)$ will be* ***extended observation surprise****.*

Note that $S(X_t) = \bar{S}(X_t, t)$. Analogically we can define extended behavior uncertainty.

Definition 15 *We define* ***extended behavior uncertainty*** $\bar{U}(X_u, t)$ *as a measure of how observation at time t changed certainty of behavior at time u:*

$$\bar{U}(X_u, t) = H\left(P(X_u|o_{0:t})\right) \tag{4.15}$$

We will refer to $\bar{U}(E^i_u, t)$ as ***extended episode uncertainty****, $\bar{U}(O_u, t)$ will be* ***extended observation uncertainty****.*

As an example, it might be the case that all the time Bob was thinking that Alice is an art lover, therefore she visits the museum every week. Thus up to the time $t_{theft} - 1$ Bob would consider her visits as a *leisure activity*. However, when he finds out in time t_{theft} that Alice has stolen a painting from that museum he would be surprised by hearing that since he never suspected her. This means that Bob's $S(E^i_{t_{theft}})$ will be high. At the same time Bob would reinterpret every one of Alice's museum visitsf to *theft preparation*. This will be indicated by high values of $\bar{S}(E^i_{t_{theft}}, u)$ for every u when Alice visited the museum. Similarly the surprise might affect Bob's thinking about the future.

We will demonstrate meaning of these definitions in an example toy domain in Section 5.5.

4.9 Model Implementation

The models described in this thesis were implemented in Java and are freely available for download[3]. For belief propagation in DBNs, SMILE[4] reasoning engine for graphical probabilistic models was used.

The implemented model covers the perception, encoding and reconstructive retrieval phases as described in the previous sections. This includes both RMaxS

[3]Project is available online at `https://code.google.com/p/dybanem`.

[4]SMILE was developed by the Decision Systems Laboratory of the University of Pittsburgh and is available at `http://genie.sis.pitt.edu`.

and RMinOS memory encoding strategies and CHMM and AHMEM probabilistic models. The storage is implemented in a most simplistic way just to support the experiments presented in later chapters. High level description of steps describing how to interface an NPC with DyBaNeM is provided in Section 7.1.

4.10 Summary

In this chapter we described internals of our EM model. We have specified how DyBaNeM works during perception, encoding, storage and retrieval and how we can create different versions of DyBaNeM based on choice of the memory encoding strategy or the structure of the probabilistic model. The next chapter demonstrates how DyBaNeM works on synthetic toy domains.

Chapter 5

DyBaNeM in Example Synthetic Toy Domain

In order to demonstrate how the DyBaNeM memory framework developed in this thesis works we will first trace several stages of the algorithm on a synthetic toy domain, second we show how DyBaNeM behaves on data from more realistic domains. This chapter is concerned with the toy domain examples, more realistic domains will be introduced in Chapter 6.

This chapter will first show how the activity recognition phase works with different probabilistic models, namely CHMM and AHM*EM* on exactly the same input. This will exemplify that some types of dependencies are not captured by the CHMM but they can be expressed in AHM*EM*. After this, a few steps of the encoding algorithm will be traced on the synthetic data which helps to build intuitive understanding of the algorithm's working. This time both probabilistic models will be tested with different mem selection strategies RMaxS and RMinOS defined in equations 4.5 and 4.6. This yields four combinations: $AHMEM + RMinOS$, $AHMEM + RMaxS$, $CHMM + RMinOS$ and $CHMM + RMaxS$. In the end, recall of all model variants will be compared side by side. Behavior surprise and certainty discussed in Section 4.8 will be demonstrated on the same example. After reading this chapter the reader should have a sense of possible difficulties in the encoding process and how more complex models can overcome them.

5.1 Domain Description

In our toy domain we have three possible observations, $\forall t : D(O_t) = \{A, B, C\}$, and a single level of episodes E^0_t with two possible values, $\forall t : D(E^0_t) = \{X, Y\}$. The following three rules define our domain:

Episodes	Observations
XX	AABAAB
XY	AABAAC
XY	AABAAC
YX	AACAAB
YX	AACAAB
YY	AACAAC
YX	AACAAB
YX	AACAAB
YY	AACAAC

Table 5.1: Training examples of the toy dataset.

1. Sequence AAB is interpreted as an episode X and AAC is Y.
2. Transition probabilities of episodes are defined as follows:

   $$P(X\ starts\ in\ time\ 3t | X\ finished\ in\ time\ 3t-1) = {}^1\!/_3$$

   That is, when X finishes, it will repeat with a probability ${}^1\!/_3$ while with probability of ${}^2\!/_3$, the next episode will be Y. Also note that every episode has to be exactly three time steps long. The same rule applies also for Y (informally: $P(Y|Y) = {}^1\!/_3$, $P(X|Y) = {}^2\!/_3$).
3. At the beginning, X is twice less likely than Y.

The toy domain can be explained by a set of training examples as well. A dataset listed in Table 5.1 contains exactly the same regularities required by rules 1, 2 and 3.

In the rest of this section both CHMM and AHMEM with three hidden states ($|D(H_t^0)| = 3$) will be used as the underlying probabilistic model. This will help to illustrate the differences in predictive power of these two models, because CHMM is quite simple and, on the other hand, AHMEM is a fairly complex one.

5.2 Activity Recognition

In this section we will demonstrate the first possible source of problems: inaccuracies in the activity recognition phase. When Bob misinterprets the true meaning of Alice's actions he tries to remember wrong facts.

In the first step, DyBaNeM performs activity recognition when it tries to "make sense" of the observations; that is, deduce higher level episodes that probably lead to the observed activity. We know that sequence $AABAAB$ should be translated to two episodes X. Figure 5.1 shows prediction of the CHMM and Figure 5.2 shows prediction of the AHMEM. Both models were trained on the same set of example episodes (see Table 5.1) following the rules of the toy domain. CHMM was trained by counting the sufficient statistics. AHMEM was trained by the EM algorithm where H_t^0 was the unobserved variable without corresponding data in the training dataset.

We can see that despite rule 1, CHMM deduces that, at time steps 0,1,3 and 4, both X and Y were possible explanations. This tendency is the strongest for time step 0, less strong for 1 and relatively weak for 3 and 4. This is due to an inherent limitation of the CHMM model that relies on the Markov assumption which posits that the state in time $t+1$ depends solely on the state in t. Even though it is clear that the high level episode is X when the model observes B in $t = 2$, CHMM is unable to propagate this knowledge back in time. Instead it is influenced by the fact that Y is more probable than X initially. MAP query would correctly answer that X was the most probable episode for all time steps; however, marginal probabilities show that not all rules specifying our toy domain are correctly captured by the CHMM. CHMM cannot distinguish between the first and the second occurrence of observation A. To be more specific, CPMF of the transition model for variable O contains the following information:

$$P(O_t = A|O_{t-1} = A, E_t^0 = X) = P(O_t = B|O_{t-1} = A, E_t^0 = X) = 1/2 \quad (5.1)$$

Translated to plain English: the model states that when the last observation was A and the episode is X then both A and B are equally likely to occur at the next time step and their probability is $1/2$ (the same applies also when we condition the probability on Y instead of X). However, we know that the $first$ A is always followed by the *second* A that is followed by B given the episode is X. This rule cannot be expressed in CHMM, there is no way to encode a context of the observations (A preceded by another A is something different than A preceded by either B or C). That is CHMM cannot distinguish between the first and the second occurence of the observation A.

AHMEM (Fig. 5.2) performs better compared to the CHMM. AHMEM captures all the episodic schemata defined in rules 1 to 3. Posteriors clearly show that prediction of AHMEM is accurate: X is the single possible explanation for all time steps. This is what one would naturally expect.

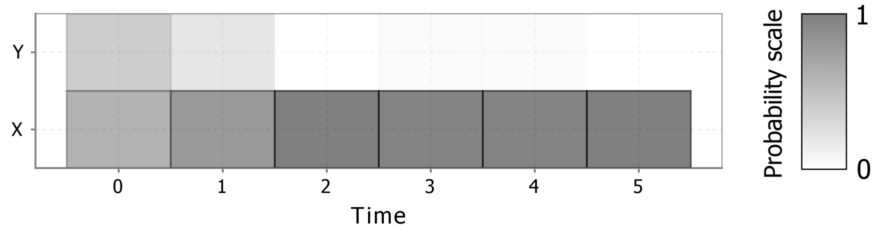


Figure 5.1: Posterior marginal probability $P(E^0_t|O_{0:5} = AABAAB)$ for each t from 0 to 5 in CHMM. X-axis shows time and Y-axis shows marginal probabilities of possible outcomes of E^0_t that are $\{X, Y\}$. Probability of each outcome is expressed by gray level. White corresponds to impossible outcome, $P = 0$, black to certain outcome, $P = 1$. This kind of plot will be used to display evolution of $P(E^0_t)$ over time.

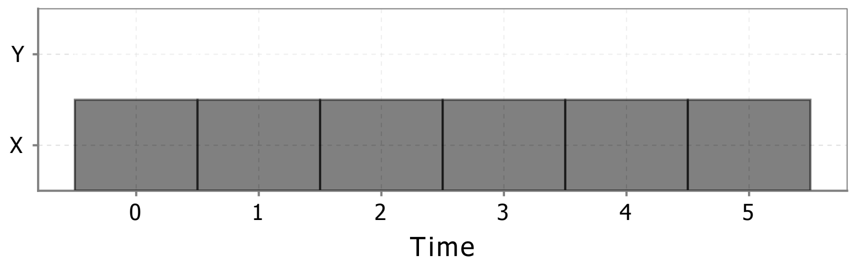


Figure 5.2: Posterior $P(E^0_t|o_{0:5} = AABAAB)$ in AHM$EM$. Legend is the same as in Fig. 5.1.

5.3 Encoding

In encoding, DyBaNeM performs a sequence of steps when it looks for the most memorable mem given the list of mems remembered so far. For a detailed description, see Section 4.2. No matter whether RMinOS or RMaxS strategy is used the first step starts with a comparison of the result of the activity recognition to the prior prediction of the activity. That is, the algorithm compares the most probable explanation of what has just been seen with what would be recalled if the model does not have any specific memories of the situation, i.e., the recall would be based solely on the episodic schemata. No mems would be used at this time since no mems were yet computed.

The next section continues in the toy domain example and it describes the prior predictions of both CHMM and AHM$EM$. The following sections discuss how the different encoding strategies affect created mems.

5.3.1 Prior Prediction for Episode Encoding

Now let us have a look at the prior predictions of observations, $P(O_t)$, and prior predictions of episodes, $P(E_t^0)$, that are used in episode encoding algorithm. The term prior refers to the fact that these are distributions computed by the model before using any observations as evidence. We start with observations whose predictions are shown in Figures 5.3 and 5.4, i.e., probability that A, B or C happens in time step t, according to the given probabilistic model. The figures clearly illustrate the difference in predictive power of the CHMM and AHMEM models.

In the first time step, both CHMM and AHMEM predict an observation A, because both episodes X and Y start with this observation and this probability is determined by the initial time slice probability (see Sec. 4.1.1 for details of the initial probability distribution denoted as $P(\mathbf{Z}_0)$). In the second time step CHMM predicts A as the most probable outcome ($P(O_1 = A) = 0.5$). However, we know that this probability should be exactly one. This is a manifestation of another limitation of the CHMM. Since the probability of observing A at time step 1 is 0.5 the CHMM also assigns some probability to the two remaining possibilities B and C ($P(O_1 = B) \approx 0.2$ and $P(O_1 = C) \approx 0.3$).

At this point it might not be obvious how we obtain those probabilities. Therefore we will explain the computations that lead to this result in Box 5.3.1.

Box 5.3.1 An example of marginalization.

In this box we will perform all steps necessary to compute $P(O_1 = B)$ and $P(O_1 = C)$ in the CHMM. This computation will also demonstrate marginalization, one of the simplest algorithms that can be used to compute probabilities in graphical models.

We will start with computation of $P(O_1 = B)$. To compute this value we have to marginalize out values of O_0, E_0^0 and E_1^0. We do this by computing the following equation:

$$\begin{aligned} P(O_1) &= \sum_{E_0^0} \sum_{E_1^0} \sum_{O_0} P(E_0^0) \cdot P(O_0) \cdot P(O_0|E_0^0) \cdot P(E_1^0|E_0^0) \cdot P(O_1|E_1^0, O_0) \\ &= \sum_{E_0^0} P(E_0^0) \sum_{E_1^0} P(E_1^0|E_0^0) \sum_{O_0} P(O_0) \cdot P(O_0|E_0^0) \cdot P(O_1|E_1^0, O_0) \quad (5.2) \end{aligned}$$

Now we can rewrite this equation with actual values of the random variables:

$$P(O_1 = B) = \sum_{e \in \{X,Y\}} P(E_0^0 = e) \sum_{f \in \{X,Y\}} P(E_1^0 = f | E_0^0 = e) \\ \sum_{a \in \{A,B,C\}} P(O_0 = a) P(O_0 = a | E_0^0 = e) P(O_1 = B | E_1^0 = f, O_0 = a) \quad (5.3)$$

We already know that $P(O_0 = A) = 1$ (and consequently also $\forall e : P(O_0 = A | E_0^0 = e) = 1$) thus the previous equation simplifies to:

$$\begin{aligned} P(O_1 = B) = \\ \sum_{e \in \{X,Y\}} P(E_0^0 = e) \sum_{f \in \{X,Y\}} P(E_1^0 = f | E_0^0 = e) \cdot P(O_1 = B | E_1^0 = f, O_0 = A) \\ = P(E_0^0 = X) \cdot \\ \Big(P(E_1^0 = X | E_0^0 = X) \cdot P(O_1 = B | E_1^0 = X, O_0 = A) + \\ P(E_1^0 = Y | E_0^0 = X) \cdot P(O_1 = B | E_1^0 = Y, O_0 = A) \Big) + \\ P(E_0^0 = Y) \cdot \\ \Big(P(E_1^0 = X | E_0^0 = Y) \cdot P(O_1 = B | E_1^0 = X, O_0 = A) + \\ P(E_1^0 = Y | E_0^0 = Y) \cdot P(O_1 = B | E_1^0 = Y, O_0 = A) \Big) \quad (5.4) \end{aligned}$$

Initial probabilities $P(E_0^0)$ and $P(O_0)$ and transition probabilities $P(E_1^0 | E_0^0)$ and $P(O_1 | E_1^0, O_0)$ can be counted from the training data in Table 5.1. After substituting we obtain the final probability:

$$P(O_1 = B) = \frac{1}{3}(\frac{17}{19} \cdot \frac{1}{2} + \frac{2}{19} \cdot 0) + \frac{2}{3}(\frac{2}{13} \cdot \frac{1}{2} + \frac{11}{13} \cdot 0) \approx 0.2 \quad (5.5)$$

The value of $P(O_1 = C)$ can be computed analogically or one can use the fact that $P(O_1 = C) = 1 - P(O_1 = A) - P(O_1 = B)$.

Even though these probabilities are understandable given what type of regularities can be captured by the CHMM, they are against the rules specifying the structure of the toy domain. For instance, the rules state that in every position $3t$ and $3t + 1$, we can observe only A, and in $3t + 2$, we can observe either B or C but never A. Fig. 5.3 thus illustrates deficiencies of the CHMM and also one possible source of encoding errors — inaccurate prior prediction of the model.

Fortunately, this inaccuracy is solved by the AHM*EM* model. As can be seen, it predicts that $P(O_{3t} = A) = P(O_{3t+1} = A) \approx 1$ whereas $P(O_{3t+2} = A) \approx 0$.[1] Probability of B and C is nonzero only in time $3t+2$ which is true by requirement 1. $P(O_2 = C) \approx 2/3$ whereas $P(O_2 = B) \approx 1/3$ this follows from rule 3 from the definition of the toy domain. The next time a high level episode ends, that is in $t = 5$, the difference between B and C fades out. Probabilities are now $P(O_5 = C) \approx 5/9$ and $P(O_5 = B) \approx 4/9$. This follows from the fact that uncertainty over possible episode evolutions accumulates over time (see Fig. 5.4). We may say that the process converges to the so called *stationary distribution* of a Markov chain which would be $\lim_{t \to +\infty} P(O_{3t+2} = B) = \lim_{t \to +\infty} P(O_{3t+2} = C) = 1/2$ in this case.

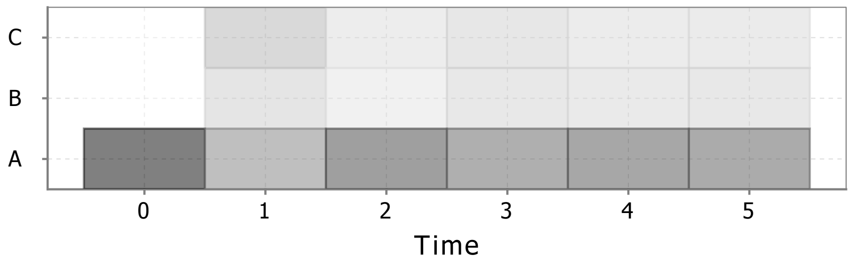


Figure 5.3: Prior probability of $P(O_t)$ in CHMM. Legend is the same as in Fig. 5.1.

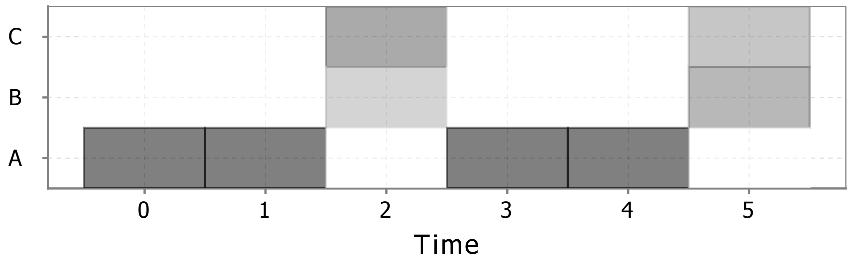


Figure 5.4: Prior of O_t in AHM*EM*. Legend is the same as in Fig. 5.1.

Analogically prior predictions $P(E^0_t)$ for the first level of episodes can be obtained. Fig. 5.5 shows prior probability in CHMM and Fig. 5.6 shows the same in AHM*EM*.

CHMM assigns correct probabilities to X and Y in the first time step, $P(E^0_0 = Y) = 2/3$ and $P(E^0_0 = X) = 1/3$. However, it fails to capture the requirement that

[1]Due to smoothing used in implementation of our model the probabilities are not exactly 1 and 0, respectively, thus we use $\approx$ sign to stress this.

each episode has to last for exactly three time steps. This is manifested by the fact that $P(E_0^0) \neq P(E_1^0) \neq P(E_2^0)$ even though all those probabilities should be the same.

Again, this rule can be expressed in AHMEM. Fig. 5.6 shows that probability remains the same for three consecutive time steps. Exact values of these probabilities would also show that the ratios of probabilities are correct, that is 1 : 2 for the first three steps and 5 : 4 for the second three steps, as detailed in Box 5.3.2.

Box 5.3.2 Ratio of episode probabilities for the second three steps.

In this box we will show why the ratio of prior episode probabilities is 5 : 4 in the second three time steps. The PMF of variables E_3^0, E_4^0 and E_5^0 is given by the following equation:

$$P(E_t^0) = \sum_{E_2^0} P(E_2^0) \cdot P(E_3^0|E_2^0), \text{ for } t \in \{3,4,5\} \tag{5.6}$$

We know that $P(E_t^0 = X) = {}^1\!/\!_3$ and $P(E_t^0 = Y) = {}^2\!/\!_3$ for $t \in \{0,1,2\}$ (rule 3). We also have the transition probabilities $P(E_{3t}^0 = i|E_{3t-1}^0 = i) = {}^1\!/\!_3$ for $i \in \{X,Y\}$ and $P(E_{3t}^0 = i|E_{3t-1}^0 = j) = {}^2\!/\!_3$ for $\langle i,j\rangle \in \{\langle X,Y\rangle, \langle Y,X\rangle\}$ (rule 2). When we put this together we get:

$$\begin{aligned} P(E_3^0 = X) = P(E_2^0 = X) \cdot P(E_3^0 = X|E_2^0 = X) + \\ P(E_2^0 = Y) \cdot P(E_3^0 = X|E_2^0 = Y) = \\ \frac{1}{3} \cdot \frac{1}{3} + \frac{2}{3} \cdot \frac{2}{3} = \frac{5}{9} \end{aligned} \tag{5.7}$$

Analogically we can compute that $P(E_3^0 = Y) = {}^4\!/\!_9$. Therefore the final ratio is 5 : 4.

5.3.2 Mem Creation: AHMEM and RMaxS

In the previous two sections we demonstrated both ability to perform activity recognition and to create prior predictions based solely on the episodic schemata. Now these two steps can be combined to compute mems, i.e., compare the result of activity recognition and the prior prediction. We will walk through the process of mem creation as described in Sec. 4.2 and show what will be the resulting mems in all combinations of probabilistic models (AHMEM and CHMM) and encoding strategies (RMaxS and RMinOS).

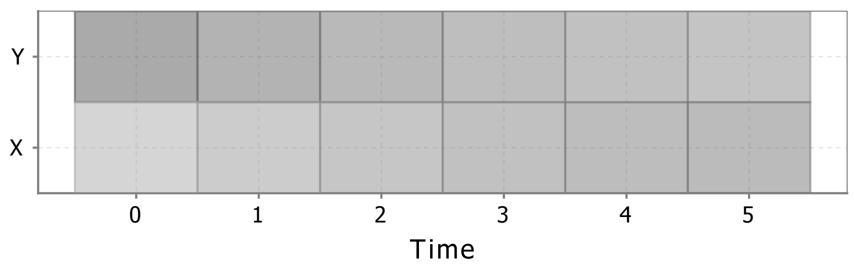


Figure 5.5: Prior of E_t^0 in CHMM. Note that with increasing time the distribution converges to the same probability for both outcomes X and Y. Legend is the same as in Fig. 5.1.

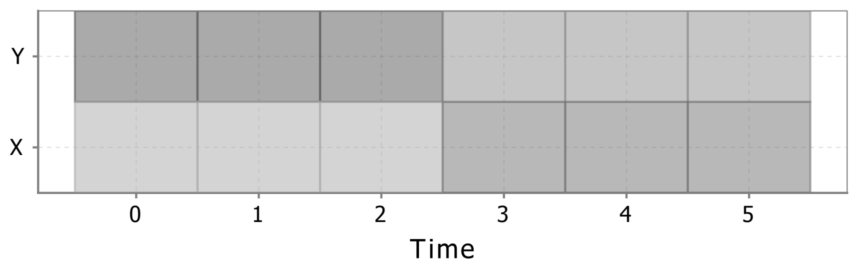


Figure 5.6: Prior of E_t^0 in AHMEM. Note that the probability remains the same for three consecutive time steps. This correctly models structure of the toy domain. Legend is the same as in Fig. 5.1.

In this section, we will start with AHMEM with RMaxS strategy as described in Eq. 4.5. The following three sections will show storage of the same sequence in AHMEM with RMinOS and CHMM with RMaxS. CHMM with RMinOS yields the same result as with RMaxS, thus it will be omitted. This step by step walk through the encoding algorithm will help to highlight important differences between the two probabilistic models and between the two mem picking strategies.

In all examples the goal will be to remember the sequence $AABAAB$ already used to demonstrate the activity recognition phase.

First mem

As already discussed, encoding in DyBaNeM works as follows: the result of activity recognition[2] is compared to the recall computed with a current set of mems.

[2]Denoted as $P_{\mathcal{M}}(Y|Per_{0:T})$ in Equation 4.4.

In the first step, no mems were computed so far, hence this degrades to comparison of the result of activity recognition to prior prediction from the episodic schemata. This situation is depicted in Fig. 5.7.

RMaxS strategy basically picks the mem that has the highest amount of surprise. That is, where the prior prediction differs the most from the activity recognition in the value of KL divergence considering all the six time steps of the sequence to be remembered. In this case the picked mem is $O_2 = B$. All other options had lower values of RMaxS. Note that the value of surprise associated with O_0, O_1, O_3 and O_4 is zero. This results from the fact that a value predicted by the schemata matches exactly the observed one, that is A. On the other hand in O_2 schemata predict C as the most probable observation with probability $2/3$. However, B that was actually observed has lower probability given the schema (only $1/3$). Thus there is discrepancy between prior prediction from schemata and the real observation. Memorizing $O_2 = B$ will reduce mismatch between the recalled sequence and the real observation. Besides correcting the value in O_2 it will also change values of the other variables as discussed in the next step.

Second mem

The second iteration of the encoding algorithm is shown in Fig. 5.8. When compared to Fig. 5.7 the reader can note changes in the new prediction. In the previous step the prior prediction from schemata assigned some probability to both X and Y episodes in time interval $\langle 0, 2 \rangle$. Now when the first mem $O_2 = B$ was created the model is certain that the only possible episode is $E^0_{0:2} = X$. This is in accordance with the result of the activity recognition, thus in the lower middle graph with values of RMaxS there is zero surprise for $E^0_{0:2}$. Remembering assignment for O_2 together with knowledge from the episodic schemata leads to a correct recall of $E^0_{0:2}$ even though this value does not have to be stored explicitly. It is computed by inference in the DBN. This is one example of the reconstructive recall enabled by DyBaNeM.

Compared to the previous step we can also note that now the probability of observing $O_5 = C$ has increased. This is because after episode X it is more likely to encounter episode Y (Req. 2) which always ends with C. Thus observing B instead of C is the most surprising fact and it becomes the new mem.

After adding this mem to the list of all mems the recall will be perfect, it will exactly match the result of activity recognition, that is, $Diff \approx 0$ for all recalled variables (see Eq. 4.4). No other mems will be needed to describe the observed sequence. Thus explicit storage of only two mems — $O_2 = B$ and $O_5 = B$, together with the episodic schemata can reconstruct the original sequence $AABAAB$ and the two high level episodes XX.

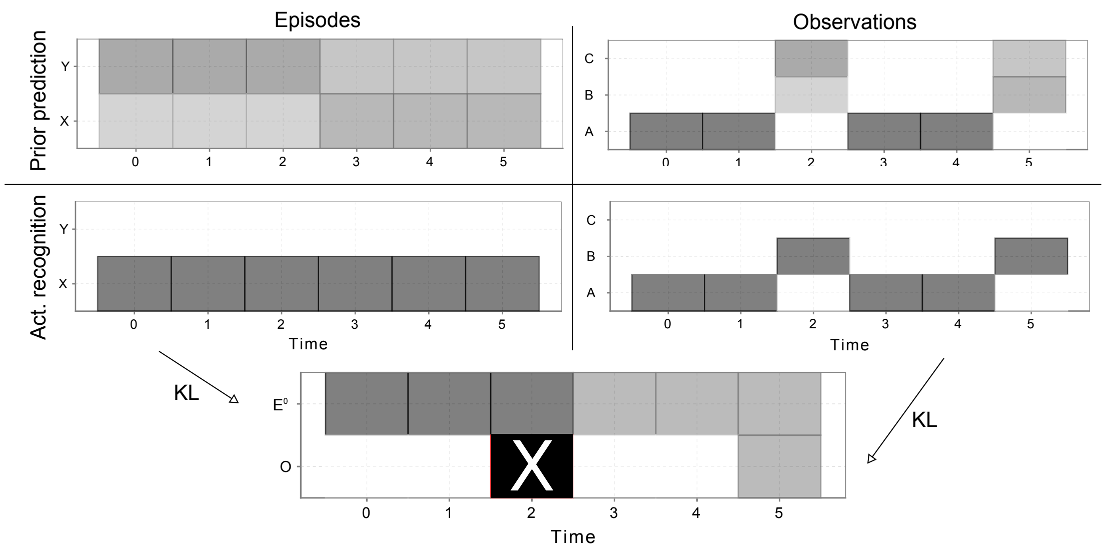


Figure 5.7: Computation of the first mem in AHM*EM* with RMaxS strategy on an input sequence $AABAAB$. The upper left figure depicts a prediction of the schemata on the level of episodes, this was already shown in Fig. 5.6. The lower left figure shows the result of the activity recognition, previously discussed in Fig. 5.2. The upper right figure depicts the prediction from schemata for observations (Fig. 5.4). The lower right figure shows the actual observations, that is a sequence $AABAAB$ that served as an input for activity recognition. The lower middle figure shows the result of RMaxS strategy (see Eq. 4.5). The first line of the middle graph shows RMaxS for each E^0_t, the lower line shows values for O_t. Levels of gray indicate the value of RMaxS. The darker the color the higher the magnitude of RMaxS for a particular level and time. The scale of gray is linearly interpolated between 0 and the maximal value of surprise achieved at any of the O_t or E^0_t variables. Note that the value of KL divergence is not bounded. Therefore the maximum may differ from case to case. The black box with a white cross highlights a variable with the highest value of RMaxS, this will become a new mem.

Alternative sequence.

What would happen if the model has to store sequence $AABAAC$ that corresponds to episodes XY instead of the $AABAAB$ sequence from the previous example? The first mem would be the same, $O_2 = B$, although the process of mem creation would diverge in the second step as shown in Fig. 5.9. Here the most probable prediction $P(O_5|O_2 = B)$ would already be C which is correct. However, there would still be some probability left for B and therefore also for the X episode. Thus even if the second mem $O_5 = C$ would be forgotten the

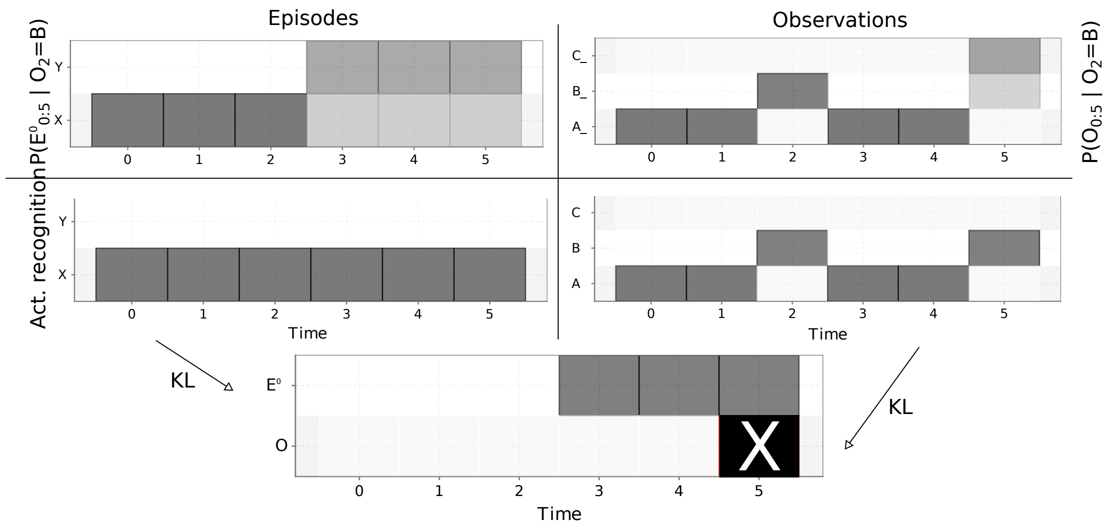


Figure 5.8: Computation of the second mem in AHMEM with RMaxS strategy on input sequence $AABAAB$. The two graphs in the first line show new predictions conditioned on the first mem that is $O_2 = B$. The left graph shows $P(E^0_t|O_2 = B)$, the right shows $P(O_t|O_2 = B)$. The second line of graphs is the same as in Fig. 5.7 since the result of the initial activity recognition remains unchanged over the whole encoding process. The lower middle graph with values of RMaxS shows that the next mem will be $O_5 = B$. Legend is the same as in Fig. 5.7.

most probable answer of the model would still be the same and it will be correct. But when the mem is accessible the recalled memories will be more certain.

We can illustrate this with entropy of the recalled sequences shown in Fig. 5.10.

5.3.3 Mem Creation: AHMEM and RMinOS

We will continue in the example from the previous section. The probabilistic model, AHMEM, will remain the same as well as the remembered sequence $AABAAB$. We will change only the memory creation strategy from RMaxS to RMinOS. This will help us to illustrate the fundamental difference between these two strategies on the familiar example.

First mem.

RMinOS as formalized in Eq. 4.6 tries to find the mem that minimizes the sum of KL divergence on all random variables corresponding to observations and episodes. Fig. 5.11 depicts values of the RMinOS on the $AABAAB$ sequence. The mem picked by the RMaxS, $O_2 = B$, and RMinOS, $E^0_{0:5} = X$ differs in this

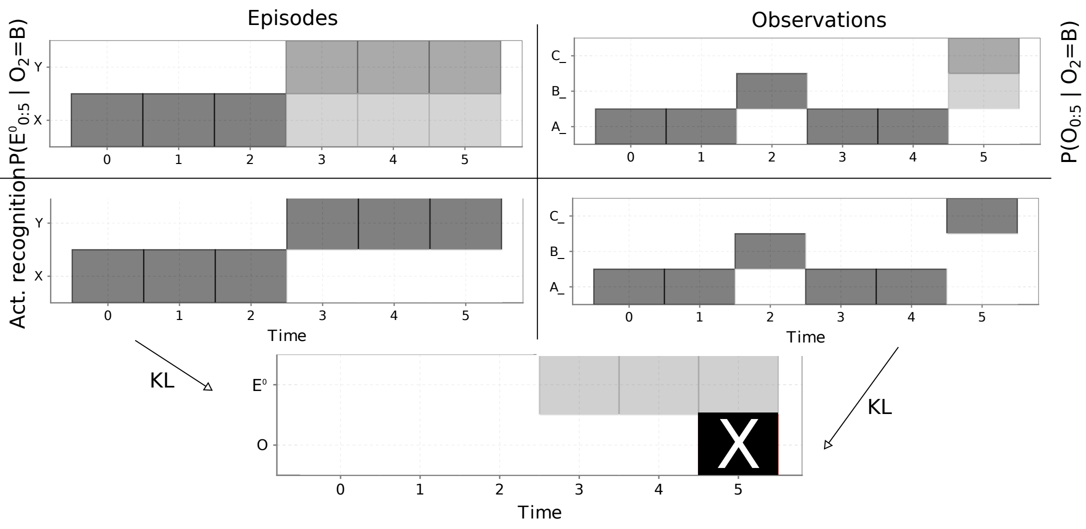


Figure 5.9: Computation of the second mem in AHMEM with RMaxS strategy on input sequence $AABAAC$. The two graphs in the first line show new predictions conditioned on the first mem that is $O_2 = B$. This is the same as in Fig. 5.8. However, the second line of graphs showing the result of the activity recognition differs. Observations are $AABAAC$ and the two recognized episodes are XY. In this case the initial prediction for $P(O_5)$, $P(E^0_3)$, $P(E^0_4)$ and $P(E^0_5)$ is almost correct in the sense that the observation/episode with the highest probability matches the reality. The lower middle graph with values of RMaxS shows that the next mem will be $O_5 = C$. Remembering this mem will remove the last uncertainty in the recall. Legend of the figure is the same as in Fig. 5.7.

case.[3] RMinOS picks $E^0_{0:5} = X$ because this assignment explicitly describes the two consecutive X episodes. Moreover values of both O_2 and O_5 can be deduced given the knowledge of the first mem ($E^0_{0:5} = X$). Thus RMinOS strategy outperforms RMaxS in this case. Evaluation on more realistic datasets shows that RMinOS is usually better than RMaxS (see Chapter 6). Perhaps the biggest disadvantage of RMinOS is that it runs slower than RMaxS (see Table 6.2 that empirically compares the time needed by both strategies).

[3]Note that in this case the mem covers two consecutive X episodes. The fact is that these two episodes can be derived in AHMEM where variable F_t segments episodes. However, in CHMM it is not possible to infer segmentation of two consecutive instances of the same episode. Since not all models can correctly infer segmentation of episodes we do not require a mem to contain exactly one episode (see Definition 1).

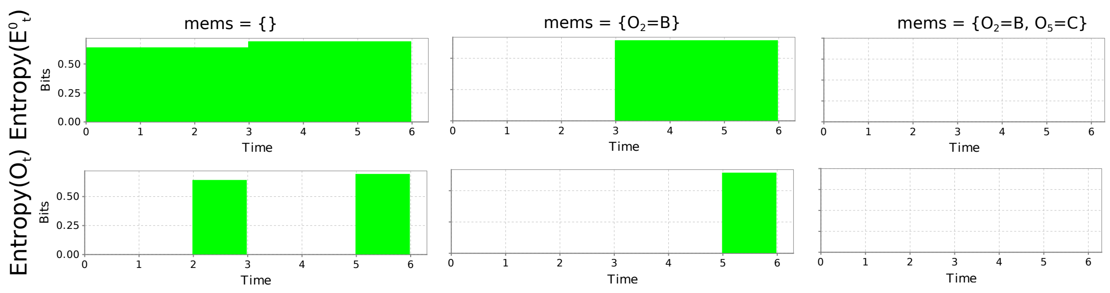


Figure 5.10: This figure illustrates how entropy can be used to measure uncertainty of a model's recall. The first line shows entropy of recalled episodes, the second line shows entropy for recalled observations in the AHMEM on the input sequence $AABAAC$. The first column is the entropy of recall with no mems, the second column is with one mem and the third is entropy of recall with two mems. In the beginning the model is quite unsure of the episodes and observations in $t = 2$ and $t = 5$. After remembering one mem the model is still unsure what is the real end of the sequence, however its most probable guess would be correct. When the second mem is taken into account there is no more uncertainty on both recalled episodes and observations.

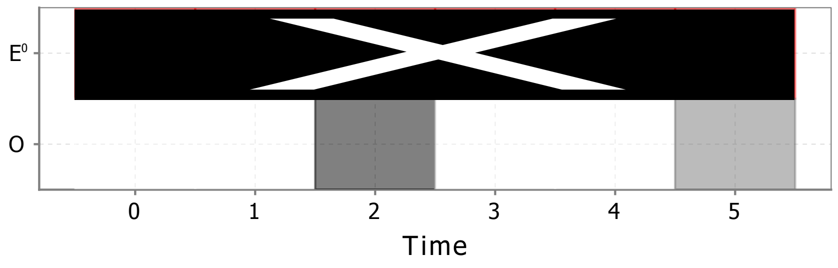


Figure 5.11: Mem picked by the RMinOS in AHMEM on the sequence $AABAAB$. Level of gray indicates the value of RMinOS for each particular group of variables.

5.3.4 Mem Creation: CHMM and RMaxS

Now we continue by showing how the encoding would work with CHMM, which is a simpler probabilistic model than the previously used AHMEM.

First mem.

Fig. 5.12 depicts the process of finding the first mem. We can see that even though the CHMM does not capture all the rules of the toy domain the picked mem is by coincidence the same as in the model with AHMEM. All important differences between AHM$EM$ and CHMM were already discussed in Section 5.3.1.

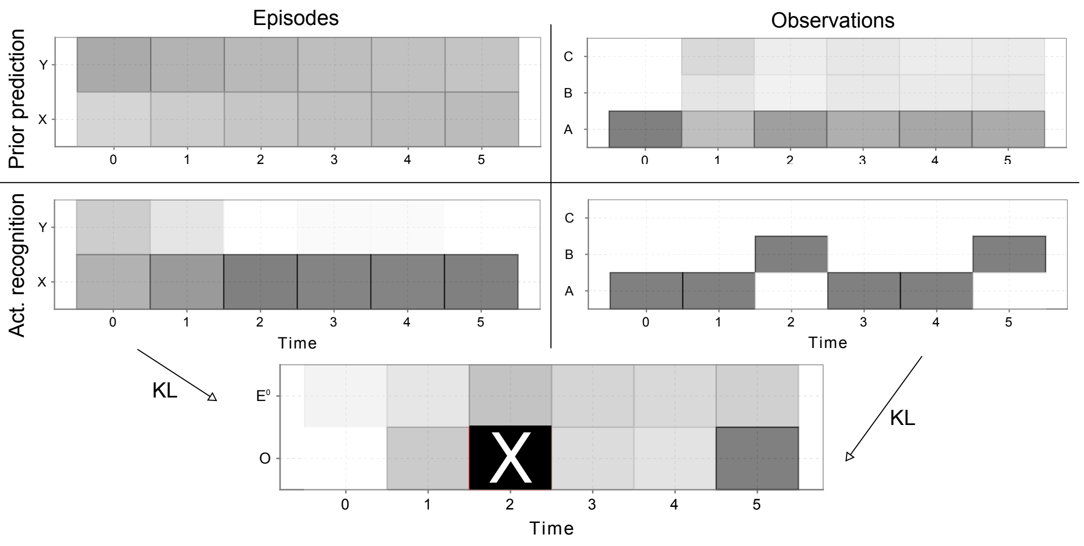


Figure 5.12: Computation of the first mem in CHMM with RMaxS strategy on input sequence $AABAAB$. This figure shows the same situation as was shown in Fig. 5.7 with the exception that now we use CHMM instead of AHM$EM$. The upper left figure shows prior prediction from the episodic schemata for episodes (previously discussed in Fig. 5.5). The upper right figure shows the same for observations (Fig. 5.3). The lower left figure shows recognized episodes (Fig. 5.1), the lower right figure shows the initial observations. The lower middle figure shows values of RMaxS, the picked mem is $O_2 = B$. Legend is the same as in Fig. 5.7.

Second mem.

It is interesting that even the second mem computed with CHMM is the same as with AHM$EM$. Fig. 5.13 shows that $O_5 = B$ will be picked in this architecture as well. However, the inaccuracy of CHMM will affect the reconstructive recall as we will demonstrate later.

Third mem.

Contrary to the example with AHM$EM$ shown in the previous section two mems are not sufficient to produce correct recall. In CHMM there is still uncertainty about the episode in the beginning of the sequence when only two mems are used. Thus the third mem will be $E_0^0 = X$ and it removes this uncertainty as shown in Fig. 5.14.

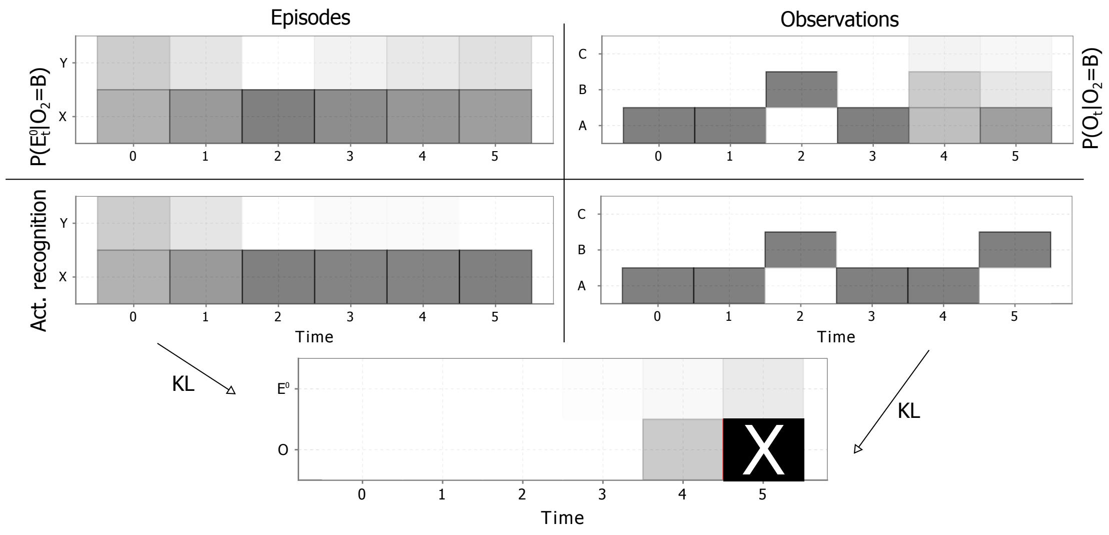


Figure 5.13: Computation of the second mem in CHMM with RMaxS strategy on input sequence $AABAAB$. Note the difference between prediction for observations here (upper right) and in AHMEM (upper right in Fig 5.8). Legend is the same as in Fig. 5.7.

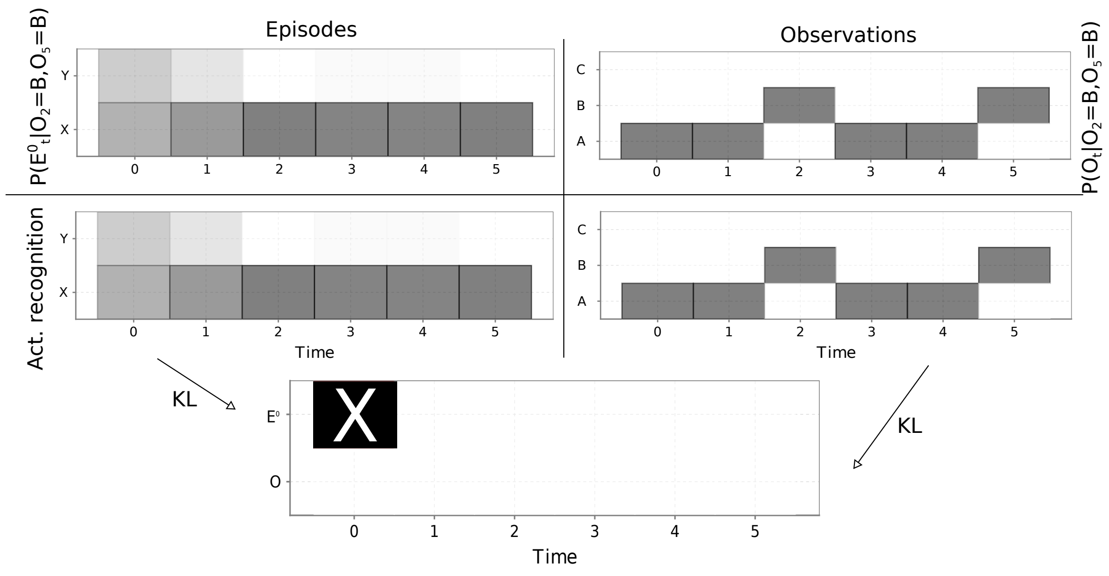


Figure 5.14: Computation of the third mem in CHMM with RMaxS strategy on input sequence $AABAAB$. Legend is the same as in Fig. 5.7.

Alternative sequence

Now analogically to the AHMEM example we will discuss what would happen when an alternative sequence $AABAAC$ is stored. Due to the inherent limitations of CHMM the activity recognition will never be perfect. Recall that activity recognition is the first step in the encoding process where the agent interprets behavior of the other agents. Activity recognition/perception takes place in the first iteration of Algorithm 1 (the same phase is also discussed in Table 4.1). Because mems are always created with respect to the result of the activity recognition this will also affect accuracy of the recall.

The first mem is again $O_2 = B$. We skip this step and show the situation when the second mem is computed, see Fig. 5.15. The most important difference compared to the same case in AHM$EM$ (Fig. 5.9) is that the activity recognition step is fairly inaccurate. Prediction of the episodes is "blurred", i.e. in $t = 3$ episode X is more probable than Y, in $t = 4$ probabilities of X and Y are reversed. However, we know that the right answer is $E^0_{0:2} = X$ and $E^0_{3:5} = Y$, which is exactly the prediction of AHM$EM$ (see Fig. 5.9). Entropy of the activity recognition is shown in Fig. 5.16, it illustrates that the CHMM is fairly uncertain about the episodes that generated the observations. As we can see, despite this, the second mem will be again $O_5 = C$. With these two mems the recall on the level of observations will be exactly the same as in AHM$EM$ but the recall on the episodic level will remain inaccurate.

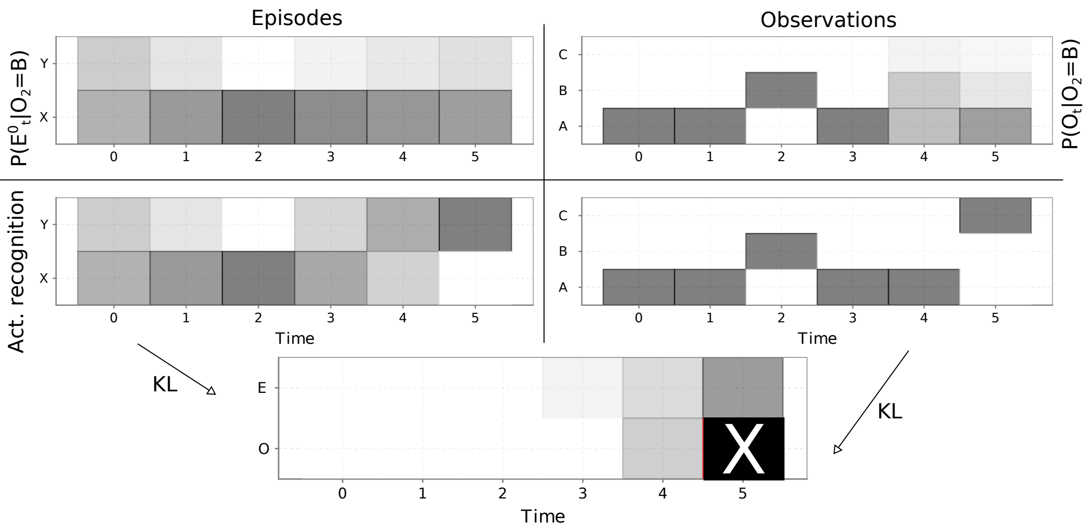


Figure 5.15: Computation of the second mem in CHMM with RMaxS strategy on input sequence $AABAAC$. Note the difference in activity recognition (lower left) compared to the $AABAAB$ sequence shown in Fig. 5.13 and more importantly the same sequence in AHM$EM$ in Fig. 5.9. Legend is the same as in Fig. 5.7.

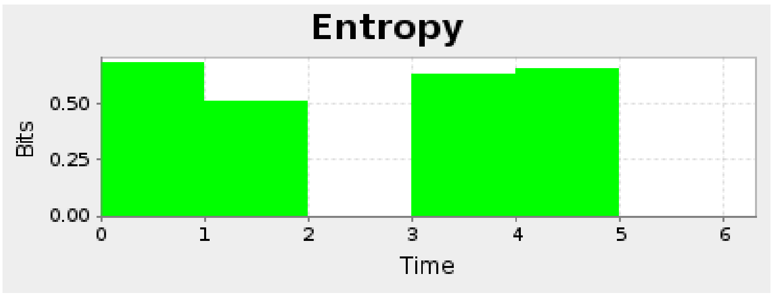


Figure 5.16: Entropy of recall on the level of episodes in CHMM on sequence $AABAAC$ when two mems ($O_2 = B, O_5 = C$) are used. Compared to the same situation in AHMEM (shown in the upper right sub-figure in Fig. 5.10) we can see that CHMM is much more uncertain.

5.3.5 Mem Creation: CHMM and RMinOS

In CHMM the mems computed with RMinOS are exactly the same as in the previous case with RMaxS (Section 5.3.4) although the numerical values of both strategies differ. Therefore the graphs are omitted in this section. This shows interplay between probabilistic models and encoding strategies since in AHMEM both strategies resulted in different mems.

5.4 Models and Strategies Comparison

Here we will compare side by side the recall of the $AABAAB$ sequence from all the architectures used in the previous sections. Fig. 5.17 shows how the number of available mems and quality of the predictive model affect the reconstructed sequence.

It can be seen that after computing two mems AHMEM+RMaxS achieves a perfect recall. In AHMEM+RMinOS only a single computed mem is sufficient for flawless recall. The least powerful CHMM+RMaxS (or CHMM+RMinOS) needs three mems for perfect recall. With only two mems there is still some uncertainty about the beginning of the sequence on the level of episodes. The insufficient power of CHMM to capture the basic domain regularities is demonstrated also in the prior prediction that uses no mems. The schemata as expressed in CHMM are not as exhaustive as the schemata in AHMEM.

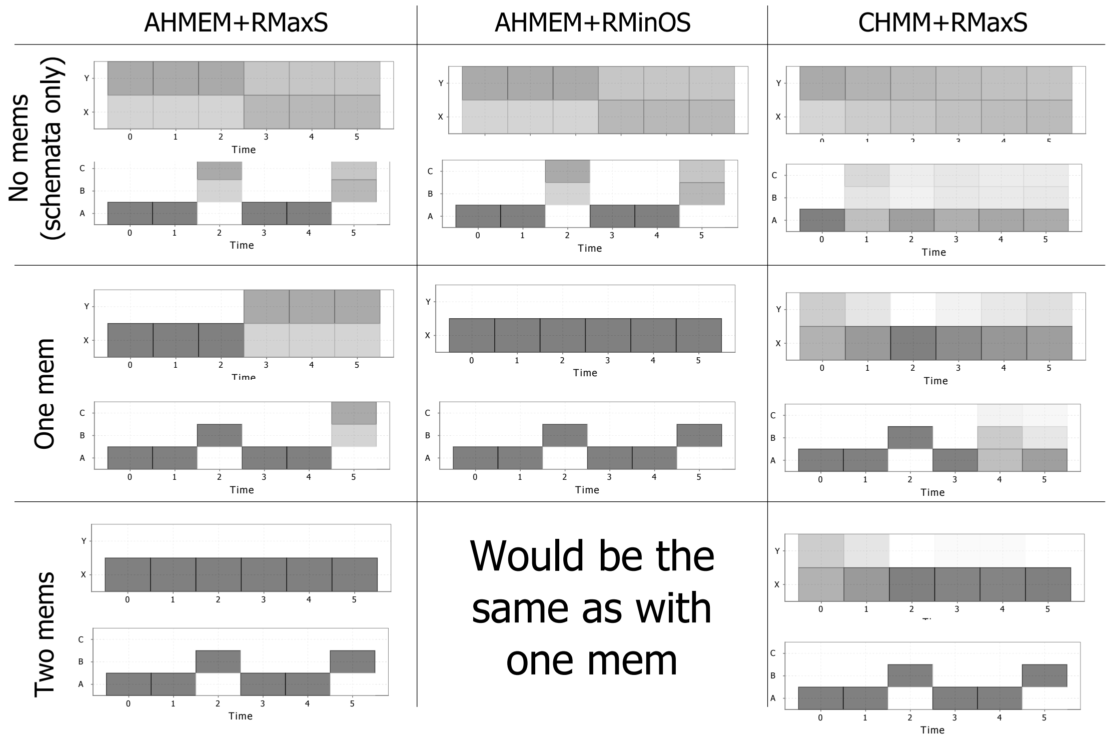


Figure 5.17: Recall of the sequence $AABAAB$ in AHMEM+RMaxS, AHMEM+RMinOS and CHMM+RMaxS architectures which is the same as CHMM+RMinOS. Each row shows recall from the given model when zero, one or two mems are available for the reconstructive process. In every cell the upper graph represents recall on the episodic level and the lower graph on the level of observations.

5.5 Online Use of the Model for Behavior Surprise and Uncertainty Estimation

In the previous sections we have seen how probabilistic models are used in activity recognition, encoding and recall. Now we will briefly show a different use of the probabilistic model. The model can be also used to compute behavior surprise and behavior uncertainty as defined in Sec. 4.8. Now we will demonstrate these definitions on the example from the domain used in the previous sections. In encoding it was always the case that Bob had already all the observations in short term memory and he retrospectively compared them against the prior prediction from schemata. Now we will inspect the dynamics of what Bob would think while gathering new observation of Alice's behavior. More specifically we will focus on measuring how sure Bob is about Alice's current behavior and whether observations match his expectations. Mismatch between Bob's expectations and reality can be interpreted as surprise.

Both these quantities can directly influence Bob's decision making process. For instance, when Bob observes Alice and he is unsure about her current activity he might ask her: "What are you doing?". When he was quite sure what she was doing but then she performs an unexpected action which in turn changes his belief about Alice's past intention he may say: "Oh! Now I now what you were doing."

Now consider the following abstract example. Bob will observe a sequence $AABAAB$ one symbol after another and we will inspect how new observations influence Bobs beliefs, his extended behavior surprise (Definition 14) and extended behavior uncertainty (Definition 15). Fig. 5.18 shows these quantities on the level of episodes, Fig. 5.19 shows the same on the level of observations. Predictions shown on those figures are made with use of the AHM$EM$ with schemata learnt on the dataset from Table 5.1.

Intuitively, Bob is not surprised by the first observation, he expected A, since it is the only possible first action. However, he is not sure whether the high level episode is X or Y. The situation remains the same even after observing the second A. This observation cannot be used to make any novel inferences. The situation changes with the third observation B in $t = 2$. First, Bob now knows that the episode in the last three time steps was X. Thus his uncertainty over the possible episodes in $t = 0, 1, 2$ decreases almost to zero[4]. He will be also a bit more certain over the next episode, Y now seems more likely than X (this follows from rule 2 of the toy domain). Second, observing B will be a bit surprising to Bob. He knows that there are two possibilities for the third symbol: B and C. Thus when Bob is not sure what will follow either observation will be a bit surprising. Prior to observation in $t = 2$ Bob considered observation C and thus episode Y as the more probable option. Therefore observing C would not surprise him as much as observing B. This discrepancy between Bob's prior belief (C was expected) and actual observation B at time step 2 is manifested by high surprise measured by the KL divergence.

In the next three time steps the situation is analogical to the first three steps. The only difference is that now Bob favors episode Y. Thus he will be surprised by observing B at $t = 5$ where he expected C with higher probability.

Figures 5.20 and 5.21 show the same situation when Bob is using simple CHMM model. This illustrates important differences between AHM$EM$ and CHMM. In the episodic level (Fig 5.20) Bob has high uncertainty in the first two time steps even after observing the whole sequence. Another flaw is that

[4]Technically there will be still a small amount of uncertainty due to smoothing. Therefore the uncertainty will not be exactly zero.

surprise in $t = 2$ and 5 does not propagate backwards but it is limited only to the respective time steps. Considering the level of observations (Fig. 5.21) Bob is unsure about the future almost in all cases. AHMEM performed better in this task. There is also deficiency in surprise estimation. Compared to AHMEM Bob is incorrectly surprised by observations in $t = 1$ and 3. The reason for this worse performance was already discussed in the examples of encoding in Sec. 5.3.4. CHMM does not correctly represent all rules of the domain thus predictions of the model are sometimes misleading.

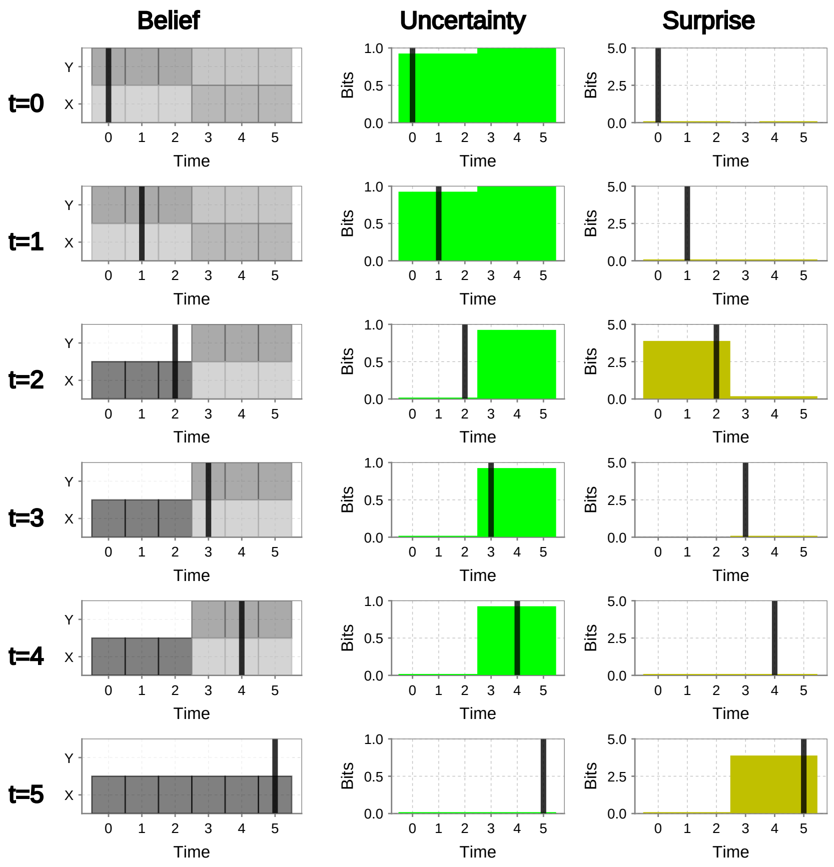


Figure 5.18: Evolution of belief, extended episode uncertainty and extended behavior surprise in AHMEM for the sequence $AABAAB$. Rows show how the previously mentioned quantities evolve as Bob gathers new observations. In $t = 0$ Bob has observed the first symbol, that is A. In $t = 2$ he has seen three symbols AAB. Analogically in the last time step $t = 5$ he observed the complete sequence $AABAAB$. The thick vertical line on all graphs marks the current time. Everything to the left of this line refers to the past and values for observations O_t (where $t \leq$ position of the thick line) are used as evidence. Everything to the right of the thick line is a future. In the first two steps, observing AA completely matches Bob's expectation. Thus Bob is not surprised (the last column), observations carried no novel unexpected information. However, he is not sure what is the high level episode (the middle column). After observing B uncertainty over the first two steps disappears. At the same time Bob will be surprised by what he observed and this surprise will propagate back in time as this observation revealed the true episode in the last three time steps (surprise in $t = 2$). Also note that there is a small amount of surprise for the future steps ($t = 3, 4, 5$). This is because now episode Y is more probable than it was in $t = 0, 1$. A similar situation repeats for the next three steps.

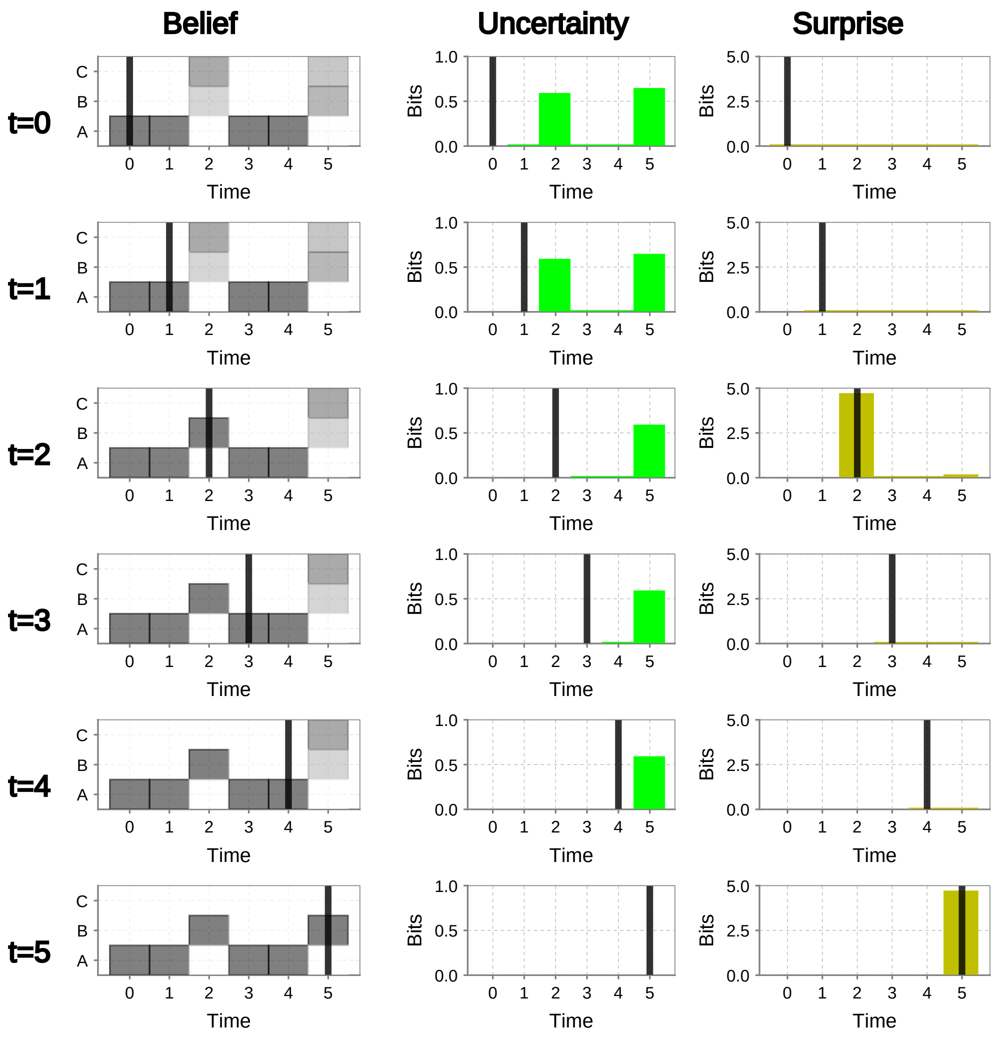


Figure 5.19: Evolution of belief, extended observation uncertainty and extended observation surprise in AHM*EM* for the sequence *AABAAB*. The difference compared to the level of episodes (Fig. 5.18) is that initially Bob is unsure only about observations at $t = 2$ and 5, he knows that the rest of the observations will be *A*. The same applies for surprise measure. The legend is the same as in Fig. 5.18.

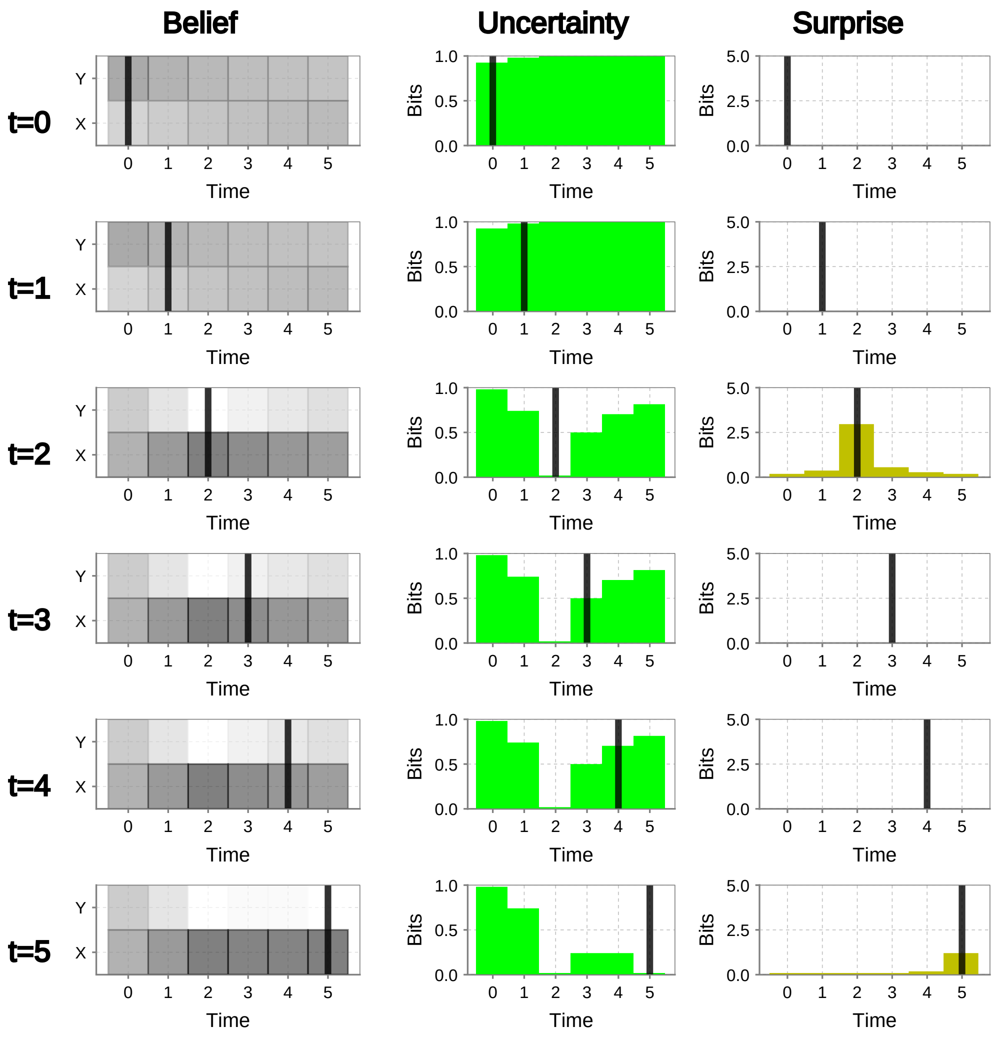


Figure 5.20: Evolution of belief, extended episode uncertainty and extended behavior surprise in CHMM for the sequence $AABAAB$. Compared to the same case in AHM$EM$ shown in Fig. 5.18 it can be seen that CHMM is not able to correctly compute certainty and surprise. Even after observing all symbols the model is unsure what the high level episodes were. The legend is the same as in Fig. 5.18.

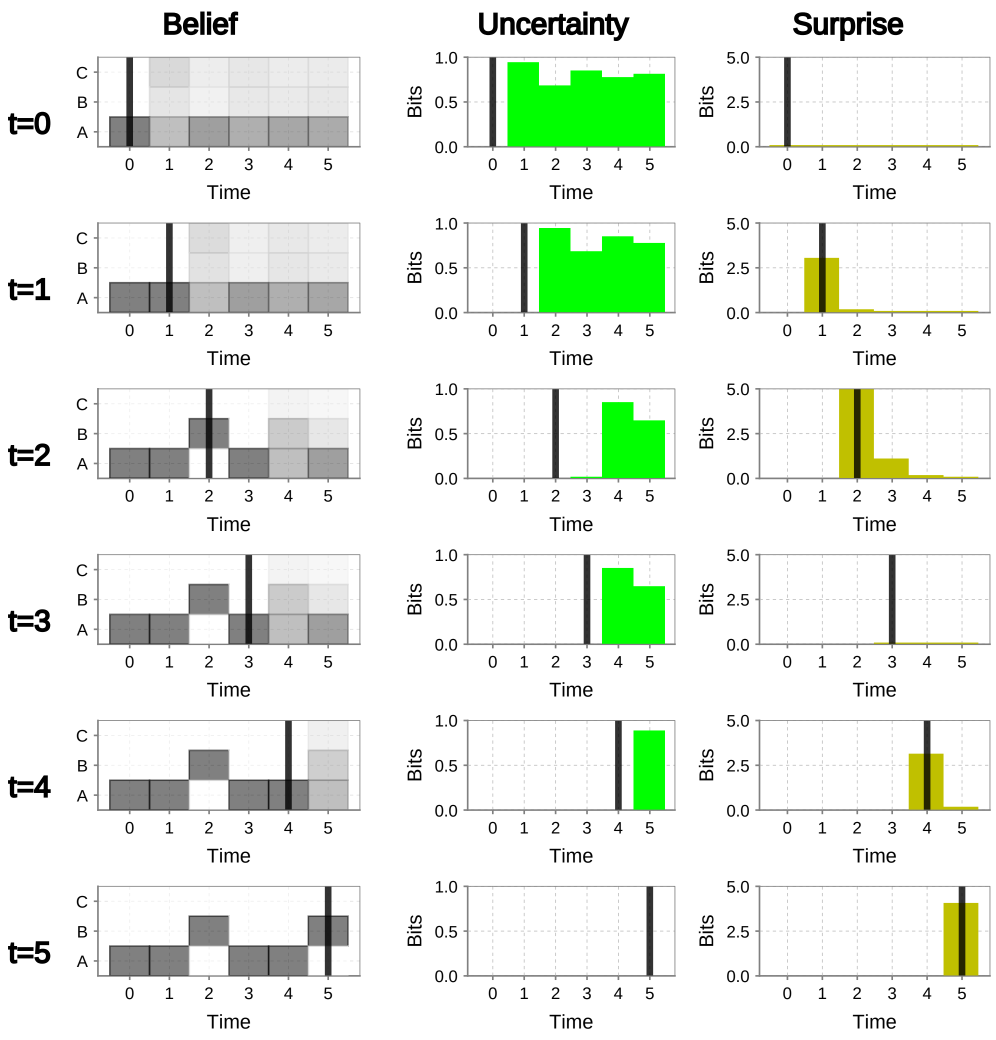


Figure 5.21: Evolution of belief, extended observation uncertainty and extended observation surprise in CHMM for the sequence $AABAAB$. Compare this figure to the same case in AHMEM shown in Fig. 5.19. The legend is the same as in Fig. 5.18.

5.6 Summary

In this chapter we demonstrated algorithms and definitions from Chapter 4 on the toy domain. We showed the importance of the probabilistic models and we highlighted important differences between the two representative examples AHMEM and CHMM. In general AHMEM proved to be a better probabilistic model than the simple CHMM, since it captured more regularities of the toy domain. We have also shown that different mem picking strategies might return different results. The next chapter will evaluate the model on more realistic datasets.

Chapter 6

Experiments

The first step in evaluation of every new technology is a "proof of concept" scenario that should verify its applicability. Chapter 5 has shown how DyBaNeM works in a trivial toy domain. Its purpose was to introduce the reader to the concepts defined in the previous sections through a set of examples. In this chapter we will describe two experiments that test several variants of DyBaNeM on two distinct more realistic domains. The first experiment tests differences between encoding strategies and different probabilistic models on a domain generated by an HTN planner[1]. The second experiment demonstrates that DyBaNeM can be used in a domain of memory modeling for IVAs[2]. Behavior of the agent in the second domain was generated by a version of a BT.

We will not try to quantitatively fit results of the experiments on human data. That is we will not try to replicate the exact outcome of experiments performed with human subjects. Instead of this we will try to show a qualitative match. We consider this approach sufficient at this stage. That is we will require that the recall "resembles" recall of humans. Nevertheless quantitative fitting human data might be an interesting future work. Our two test corpora are computer generated. The first corpus is generated by randomized hierarchical planning algorithm. The second originates from a virtual agent embodied in a 3D environment. In both domains we test a scenario where one NPC observes another NPC. This scenario will be common in computer games.

Every time one uses artificially generated corpora of human activities an important concern should be whether these datasets match the main characteristics of real human activities. We investigated this question in (Kadlec et al., 2013) where we compared several artificial and real corpora based on conditional entropy and compressibility of action sequences. The comparison included the corpus that we will use in an experiment from Section 6.2 and a modified version

[1]Preliminary version of these experiments was published in (Kadlec and Brom, 2013b).

[2]Preliminary version of these results was published in (Kadlec and Brom, 2013a).

of s corpus used in Section 6.1. We found a reasonable match between these two types of corpora. This suggests that the artificial corpora might be used as an approximation of real corpora.

All experiments in this chapter were run on a Win7 64bit machine with 16GB of RAM utilizing a single core of Intel Pentium i5-2520M 2.50GHz CPU. Detailed instructions that describe how to reproduce experiments in this chapter are given in Appendix A.

6.1 Monroe Corpus: Recall Accuracy - RMaxS and RMinOS in CHMM and AHMEM

6.1.1 Aim

In this experiment we evaluate recall accuracy depending on:

- The probabilistic model used to represent schemata. We tested CHMM^1 and AHMEM$_8^1$ models. The upper index denotes a number of episodic levels represented in the model, the lower index denotes a number of states of the internal memory ($|D(H_i)|$) in the AHMEM. This parameter is not applicable to the CHMM since it does not have the random variables H_i.
- The mem creation strategy, namely RMinOS and RMaxS.
- Number of mems used to aid the recall. We will use 1, 2 and 3 mems. This will allow us to see the effect of more evidence on the quality of recall.
- Amount of data used to learn the episodic schemata.

Our initial hypotheses are that: AHMEM is better than CHMM (for reasons explained in Section 4.1.4 and empirically demonstrated in Chapter 5); RMinOS is better than RMaxS (since RMinOS tries to minimize surprise associated with all random variables in the model); using more mems used as evidence for recall leads to better accuracy and that more data used to train episodic schemata result in better recall accuracy.

6.1.2 Dataset

We used Monroe plan corpus[3] (Blaylock and Allen, 2005a) since it is similar to real human activity corpus and it was already used by other researchers in activity recognition and episodic memory research. Monroe corpus is an artificially

[3]Monroe corpus is downloadable from http://www.cs.rochester.edu/research/speech/monroe-plan/ [23.1.2015]

generated corpus of hierarchical activities created by a randomized HTN planner. The domain describes rescue operations like cutting down a fallen tree to clear a road, transporting wounded to a hospital, etc. The corpus contains up to 9 levels of episodes' hierarchy, we shortened these episodic traces to contain only the atomic actions and the highest level episodes. The corpus features 28 atomic actions and 43 episodes. Only 10 episodes appear on the highest level. For a complete list of atomic action and the highest level episodes see Appendix B.

6.1.3 Method

For the purpose of the experiment we split the stream of episodic traces into sequences of 10. The agent that tries to remember the activity sees only 10 consecutive atomic actions of an activity that it will later try to recall. This corresponds to a situation where Bob (the observing agent) sees only a short sequence of Alice's actions (the observed agent) and then he stops observing her. This is a common situation in virtual environments where agents usually interact only briefly. We tested accuracy of recall of atomic actions and episodes on 100 sequences that were not contained in the training set. For every probabilistic model, encoding strategy and a number of stored mems that the models recall is compared to the ground truth. This way we obtain accuracy on the level of observations which is a percentage of correctly recalled atomic actions. That is:

$$Accuracy(o_{0:9}^{truth}, o_{0:9}^{recall}) = \frac{\sum_{i=0}^{9} \mathbb{1}(o_i^{truth}, o_i^{recall})}{10},$$

where $\mathbb{1}$ is the identity function. Accuracy on the level of episodes is computed analogically.

In experiments where we measure influence of the amount of training data we started with just 25 training sequences and in each step we added other 25 sequences up to the total of 725 sequences.

Inference over the DBN unrolled to 10 time steps was performed by exact clustering algorithm (Huang and Darwiche, 1996).

Steps needed to re-run these experiments are described in Appendix A.3.

6.1.4 Results

Table 6.1 summarizes recall performances of the models on the level of observation and on the level of episodes when 725 sequences were used for training the episodic schemata. We use a simple crude accuracy measure that measures how many atomic actions/high level episodes were correctly recalled at the correct time. Only the most probable action at each step was used to measure the accuracy.

	arch \ mems	RMaxS 1	RMaxS 2	RMaxS 3	RMinOS 1	RMinOS 2	RMinOS 3
O	CHMM^1	55 ± 17	74 ± 16	87 ± 12	60 ± 15	76 ± 15	88 ± 11
	AHMEM$^1_8$	63 ± 18	86 ± 17	95 ± 12	64 ± 17	86 ± 14	94 ± 10
E^0	CHMM^1	65 ± 35	79 ± 26	79 ± 27	71 ± 31	78 ± 29	80 ± 27
	AHMEM$^1_8$	67 ± 34	83 ± 28	85 ± 26	77 ± 27	83 ± 27	85 ± 26

Table 6.1: Results of the recall experiment for all tested models, encoding strategies and the number of stored mems and level of hierarchy (observations O and first level of episodes E^0). The table reports mean recall accuracy in percentage and its standard deviation on the 100 testing sequences when the schemata were learnt on 725 sequences.

arch. \ strat.	RMaxS	RMinOS
AHMEM$^1_8$	122 ± 6	551661 ± 136062
CHMM^1	103 ± 89	20242 ± 2572

Table 6.2: Average time $\pm$ SD needed to compute one mem measured in microseconds. The average is over the time needed to compute the first, second and third mem in 100 testing sequences.

That is we measure 1st-best accuracy. Still the model provides more information, for instance one can measure accuracy of the second or third best hypothesis. However, the 1st-best accuracy is of the highest interest since it measures the accuracy of the first answer that would be recalled by the agent.

Table 6.2 shows the average time needed to compute a single mem in each model. We can see that RMaxS has significantly lower time requirements than RMinOS.

Effect of more mems available for recall

Now we test whether increase in recall accuracy when using more mems is statistically significant. All statistical tests are performed by paired one sided t-tests where p values were adjusted with Bonferroni correction (Bonferroni, 1935) to compensate for family-wise error rate ($p = p_{orig} \cdot 40$, since there are 40 tests in total reported in tables 6.3, 6.4 and 6.5).

The results for all pairwise differences (two mems versus one mem, three mems versus two mems) for all levels, encoding strategies and probabilistic models are shown in Table 6.3. All results are statistically significant with the exception

Model		2 mems > 1 mem		3 mems > 2 mems	
		p	Cohen's d	p	Cohen's d
$CHMM^1$ + $RMaxS$	O	$< .001$	1.141	$< .001$	0.932
	E^1	$< .001$	0.462	$> .1$	-0.008
$CHMM^1$ + $RMinOS$	O	$< .001$	1.099	$< .001$	0.906
	E^1	.042	0.226	$> .1$	0.068
$AHMEM^1$+ $RMaxS$	O	$< .001$	1.312	$< .001$	0.617
	E^1	$< .001$	0.500	$> .1$	0.075
$AHMEM^1$+ $RMinOS$	O	$< .001$	1.454	$< .001$	0.652
	E^1	.017	0.237	$> .1$	0.060

Table 6.3: Results of testing the hypothesis that having n mems is better than $n-1$ mems. The tests were performed using paired sided t-test. The table shows p values and effect sizes measured using Cohen's d (Cohen, 1988). The first two columns show results for the hypothesis that having two mems is better than having just one mem. The second two columns show the same for comparison of three versus two mems. The results that are not significant ($\alpha = 0.05$) are highlighted. The table reports results from 100 testing sequences when the schemata were learnt on 725 sequences.

of the case of three versus two mems on the level of episodes in all models and encoding strategies. Table 6.4 shows results for a test of the hypothesis that AHMEM$^1_8$ is better than CHMM^1. The results on the level of episodes are not significant. On the other hand five out of six result on the level of observations are significant. Table 6.5 shows result for test of the hypothesis that RMinOS is better strategy than RMaxS. In this particular dataset RMinOS outperforms RMaxS significantly only in recall on the level of episodes when one mem is used.

Effect of training data

The next series of graphs shows how the amount of training data influences recall accuracy. Table 6.1 shows the results when all 725 sequences were used to train the episodic schemata. The Figures 6.1 to 6.8 show how the models behave when we use less training data.

The first two figures show performance of AHMEM$^1_8$ with RMinOS strategy on the level of observations (later denoted as AHMEM$^1_8 + RMinOS + O$) (shown in Figure 6.1) and on the level of episodes (Figure 6.2). RMaxS strategy in the same model shows a similar trend on the level of observations, therefore we omit this figure. However, AHMEM$^1_8 + RMaxS + E^0$ shows a different trend when the recall with just one mem is consistently better with RMinOS, see Figure 6.3.

Now we will focus on CHMM^1. Figure 6.4 shows CHMM^1 + $RMinOS$ +

Strategy	Level	Mems	p	Cohen's d
RMaxS	O	1	$< .001$	0.412
		2	$< .001$	0.698
		3	$< .001$	0.614
	E^1	1	$> .1$	0.063
		2	$> .1$	0.130
		3	$> .1$	0.218
RMinOS	O	1	$> .1$	0.262
		2	$< .001$	0.700
		3	$< .001$	0.618
	E^1	1	$> .1$	0.185
		2	$> .1$	0.178
		3	$> .1$	0.177

Table 6.4: Results of testing the hypothesis that AHMEM is a better probabilistic model than CHMM. The legend is the same as in Table 6.3.

Arch.	Level	Mems	p	Cohen's d
$CHMM^1$	O	1	.061	0.266
		2	$> .1$	0.129
		3	$> .1$	0.053
	E^1	1	$> .1$	0.195
		2	$> .1$	-0.036
		3	$> .1$	0.041
$AHMEM^1$	O	1	$> .1$	0.068
		2	$> .1$	0.045
		3	$> .1$	-0.019
	E^1	1	.012	0.315
		2	$> .1$	0.018
		3	$> .1$	0.004

Table 6.5: Results of testing the hypothesis that RMinOS encoding strategy is better than RMaxS. The legend is the same as in Table 6.3.

O, Figure 6.5 shows recall in the same model on the level of episodes (that is CHMM^1 + $RMinOS$ + E^0). As in AHMEM_8^1 RMaxS strategy again shows a similar trend and therefore we omit the related graphs.

Direct comparison of recall accuracy of AHMEM_8^1 + $RMinOS$ and CHMM^1 + $RMinOS$ is shown in Figure 6.6. We can see that AHMEM_8^1 performs consistently better than CHMM^1. The same applies also for the level of episodes (Figure 6.7) when the models have enough training data. When the schemata are trained on less than 100 sequences CHMM^1 performs better. We omit the same graphs with RMaxS strategy since the general trend is the same.

When comparing memory encoding strategies RMinOS is in most cases at least as good as RMaxS. The only exception is AHMEM_8^1 + O with three mems shown in Figure 6.8.

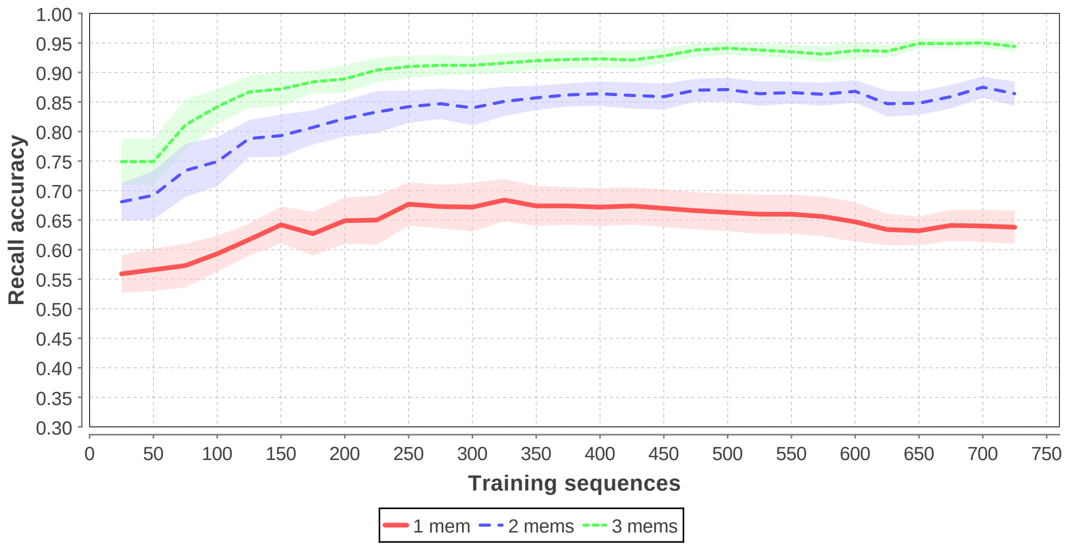


Figure 6.1: Influence of the amount of the training data on 1st-best recall accuracy in AHMEM_8^1 using the RMinOS strategy on the level of observations. We will denote this combination as AHMEM_8^1 + $RMinOS$ + O. The ribbon shows a variance of the data. Note that more mems lead to better recall. Adding more training data helps universally when three mems are used as evidence. However, the training data between sequences 625 and 650 seem to harm performance of the recall with one and two mems. Also note that the variance of recall accuracy with three mems decreases with more training data.

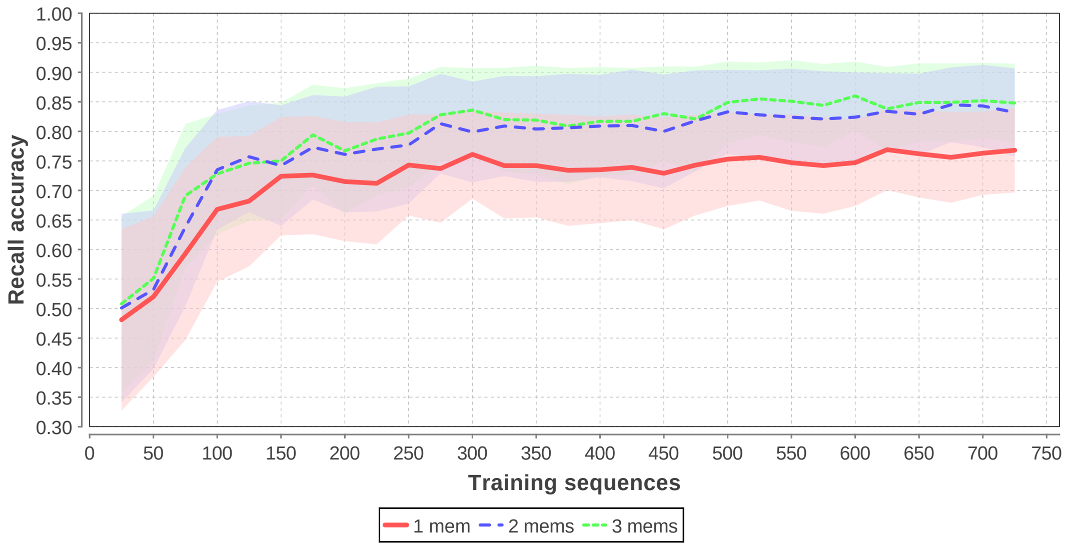


Figure 6.2: Influence of the amount of the training data on 1st-best recall accuracy in $\text{AHM}E\text{M}_8^1 + RMinOS + E^0$. Note higher variance compared to recall in the same model on the level of observations (shown in Figure 6.1).

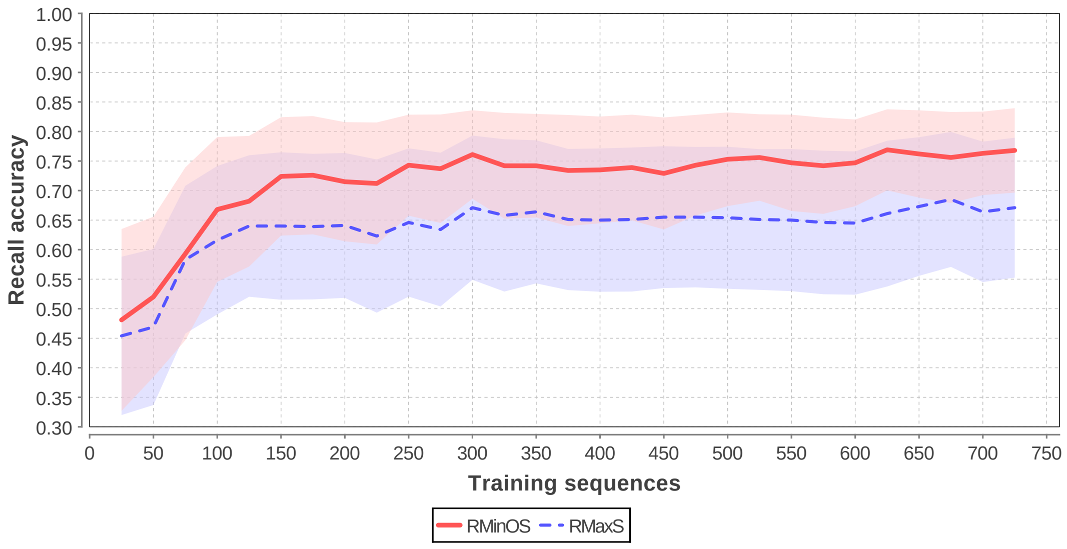


Figure 6.3: Comparison of recall accuracy on the level of episodes in $\text{AHM}E\text{M}_8^1$ when using RMinOS and RMaxS with just one mem.

6.1.5 Discussion

First of all we will summarize the most important experimental results. Then we show how the recall is affected by a number of mems on one example day. The

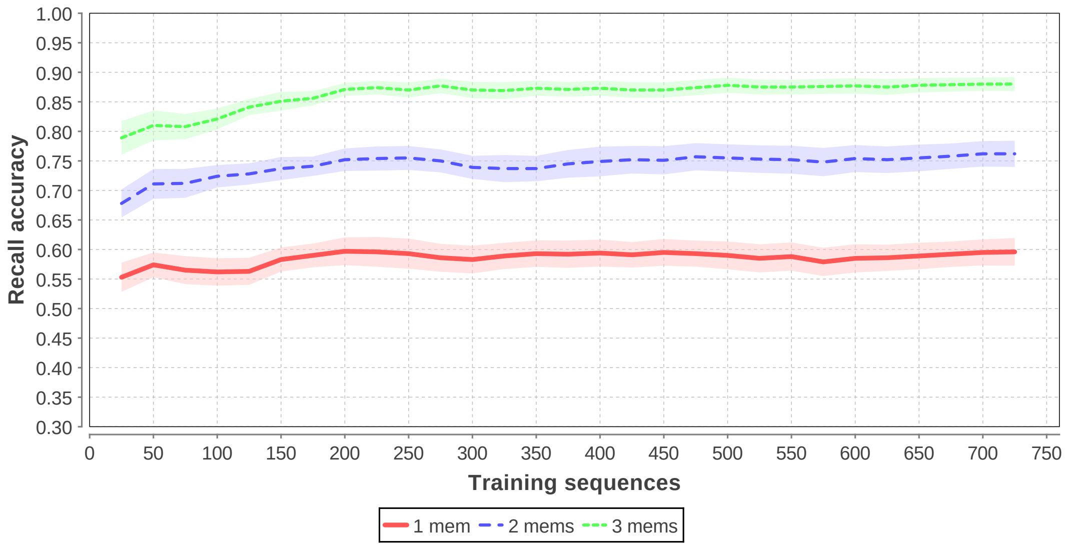


Figure 6.4: Recall in $\text{CHMM}^1_8 + RMinOS + O$. Approximately after using 200 sequences for training the model does not improve. Compare this figure with Figure 6.1 showing the same situation in $\text{AHM}E\text{M}^1_8$, which can make use of more training data.

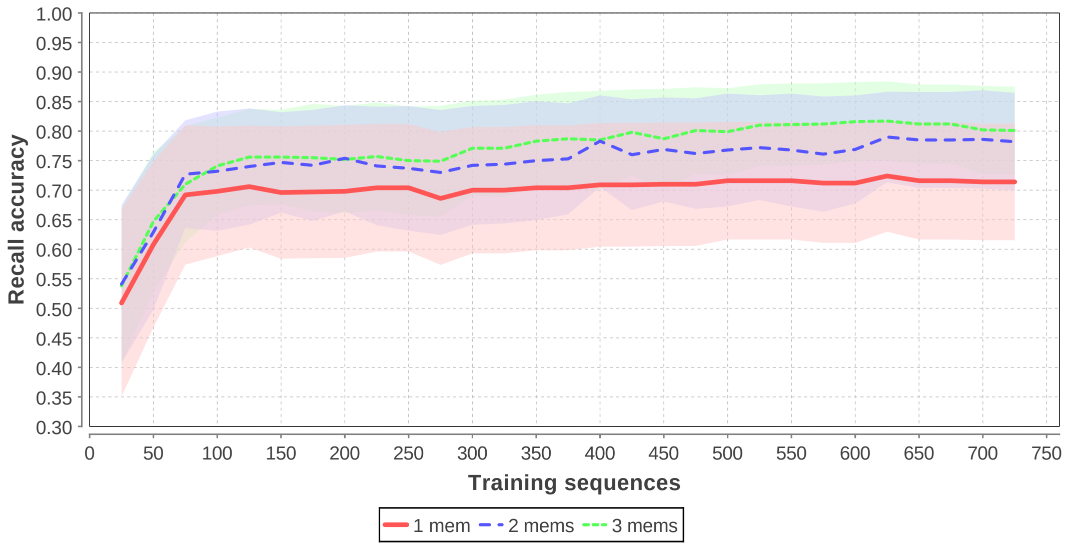
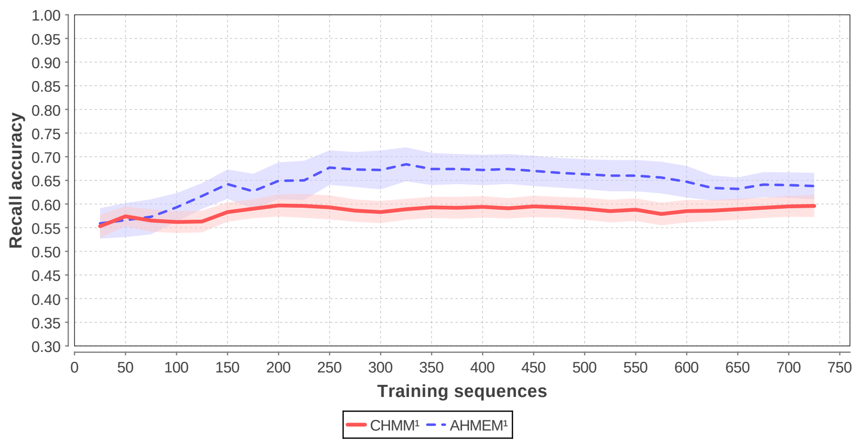
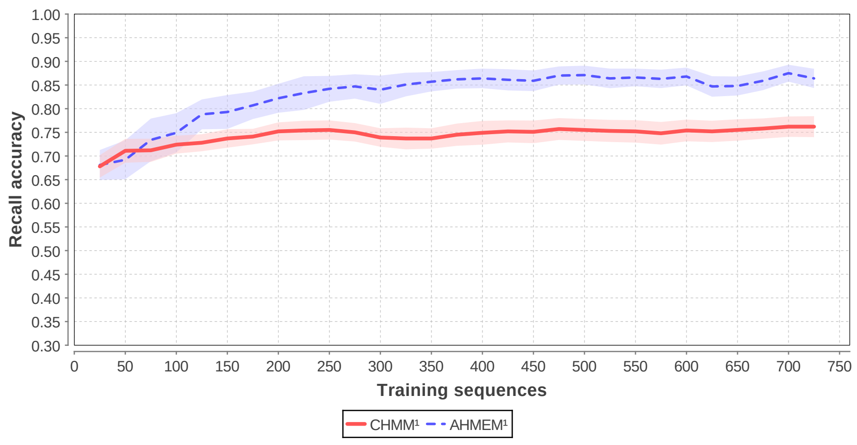


Figure 6.5: Recall in $\text{CHMM}^1_8 + RMinOS + E^0$. Compare this figure with Figure 6.1 showing the same situation in $\text{AHM}E\text{M}^1_8$, which can make use of more training data.

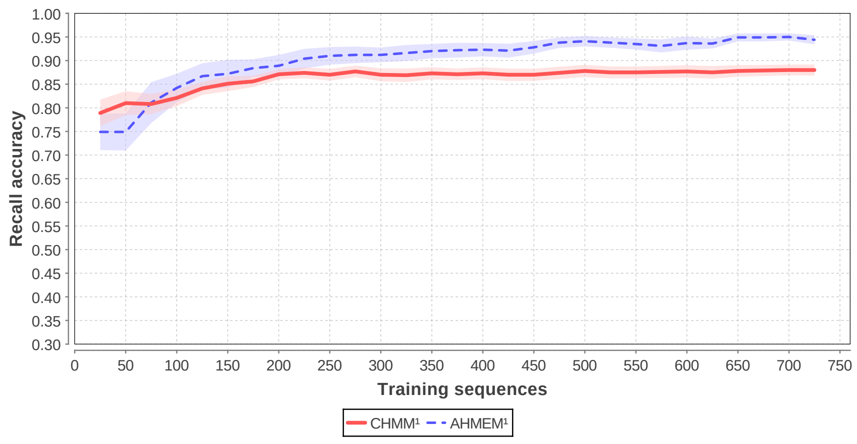
most important results are:



(a) One mem

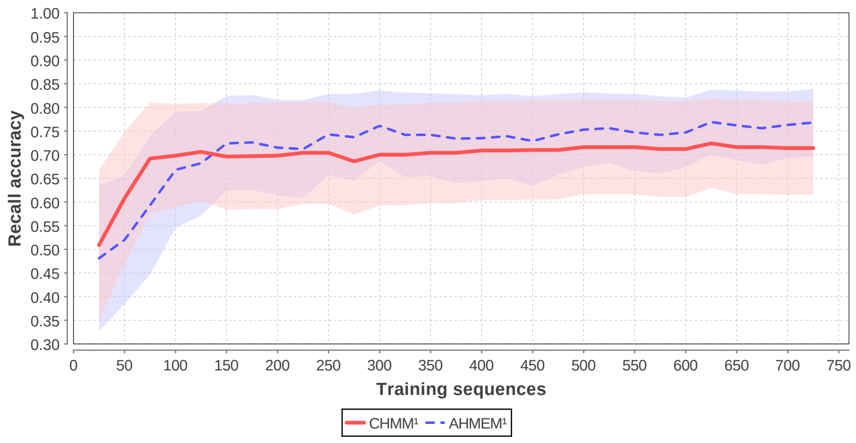


(b) Two mems

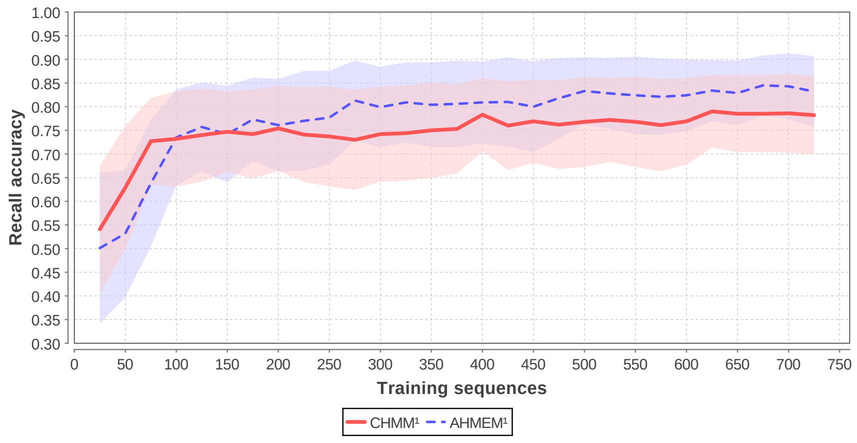


(c) Three mems

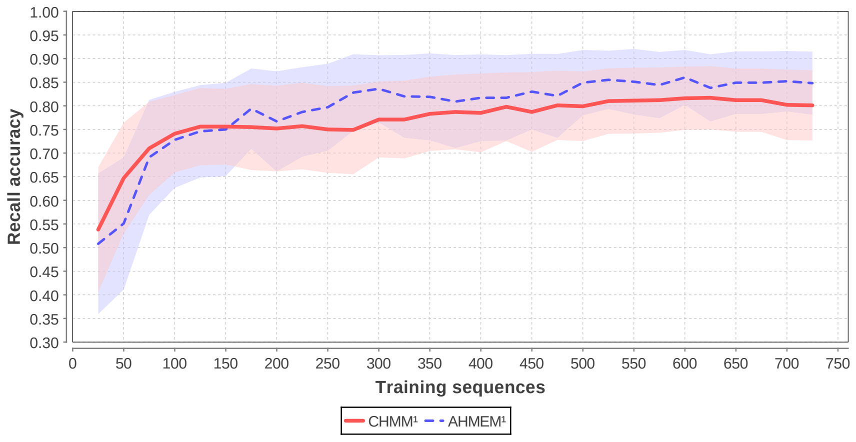
Figure 6.6: Comparison of recall accuracy between AHMEM$_8^1$ and CHMM^1 on the level of observations when using RMinOS. These figures are superpositions of the graphs from Figures 6.1 and 6.4.



(a) One mem



(b) Two mems



(c) Three mems

Figure 6.7: Comparison of recall accuracy between $\mathrm{AHM}E\mathrm{M}_8^1$ and CHMM^1 on the level of episodes when using RMinOS. These figures are superpositions of the graphs from Figures 6.2 and 6.5.

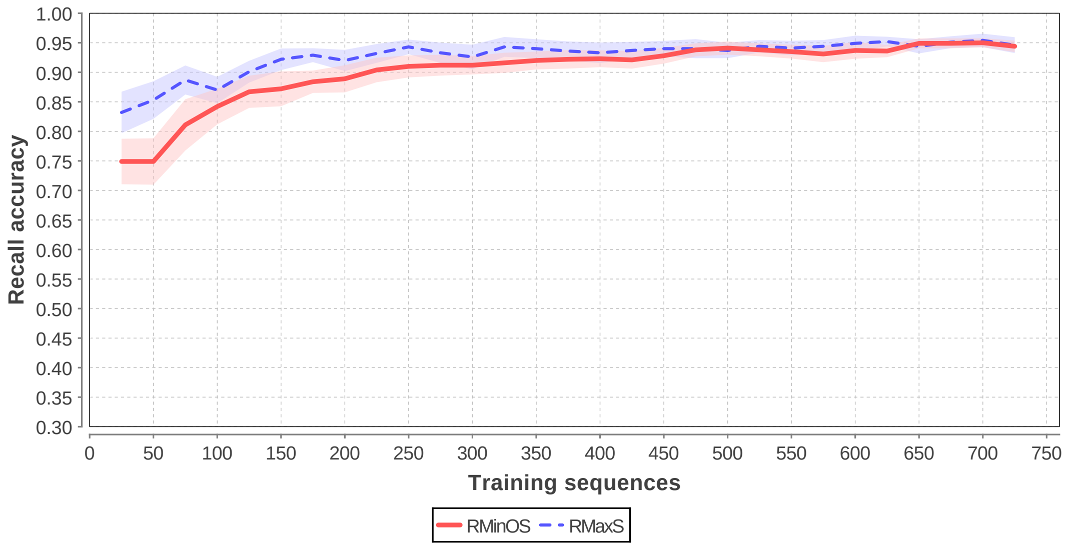


Figure 6.8: Comparison of RMinOS and RMaxS strategies on recall accuracy in $\mathrm{AHM}E\mathrm{M}_8^1 + O$ when three mems are used as evidence for recall. This is the only case where RMaxS performs better than RMinOS.

- Using more mems helps to achieve better recall accuracy.

- $\mathrm{AHM}E\mathrm{M}_8^1$ is generally a better probabilistic model than CHMM^1 when the training dataset is large.

- However, when the training dataset is limited it might be better to use CHMM since it seems to generalize better in this domain, see Figure 6.7 when using only fewer than 100 sequences for training.

- RMinOS is significantly better than RMaxS only in AHMEM on the level of episodes when the recall uses only one mem. See Figure 6.3).

- RMaxS is much faster to compute than RMinOS. On this domain RMaxS would be suitable for online use even with exact inference algorithm.

- Learning episodic schemata on more training examples helps more in AHMEM. We may say that AHMEM has higher “capacity” than CHMM. Performance of CHMM stops improving after using approximately 200 training sequences, whereas AHMEM can make use of more training data. See Figure 6.6.

- AHMEM is prone to over-fit to the training data. A possible example of over-fitting is shown in Figure 6.1 where the accuracy of recall on the level of observations slowly degrades after once the model is learnt on more than 350 sequences. However, AHMEM is still better than CHMM in this case, see Figure 6.6a.

Based on the results of these experiments we can suggest using AHMEM as the primary probabilistic model of choice. RMaxS is slightly worse in recall accuracy than RMinOS, however, it is much less computationally expensive. Therefore the decision of which strategy to use depends on the time budget that can be spent in the encoding phase.

However, note that these results hold only on the Monroe corpus. When applying DyBaNeM to new domain the same set of experiments should be performed to evaluate performance of different models and memory encoding strategies.

Example of Recall

We now demonstrate how the number of stored mems affects the recalled sequence on an example of one sequence. In an encoded example sequence only direct observations (values of O_t) ended stored in the mems. However, this does not have to be true in general. Fig. 6.9 shows probability distributions when considering a different number of mems for recall of activities from the example sequence. The mems are sorted according to the order in which they were created by the encoding algorithm. Hence we can visualize how the forgetting would affect recall since the third mem is the least significant one and it will be forgotten as the first, whereas the first mem will be forgotten as the last. After forgetting all the mems the model would return $NAVIGATE\_VEHICLE$ for each time point giving us 30% recall accuracy because this action appears three times in this particular sequence. With one mem a remembered episodic schema is activated and accuracy grows to 50%. The second mem further specifies the activated schema and changes the most probable action not only in $t = 9$, which is the mem itself, but also in $t = 3, 5, 6$ and 8, and the recall accuracy on the level of observations rises to 100%. The third mem removes the last uncertainty at $t = 4$. That is, the recall accuracy remains the same. However, calibration of probabilities is even better than in the case with only two mems.

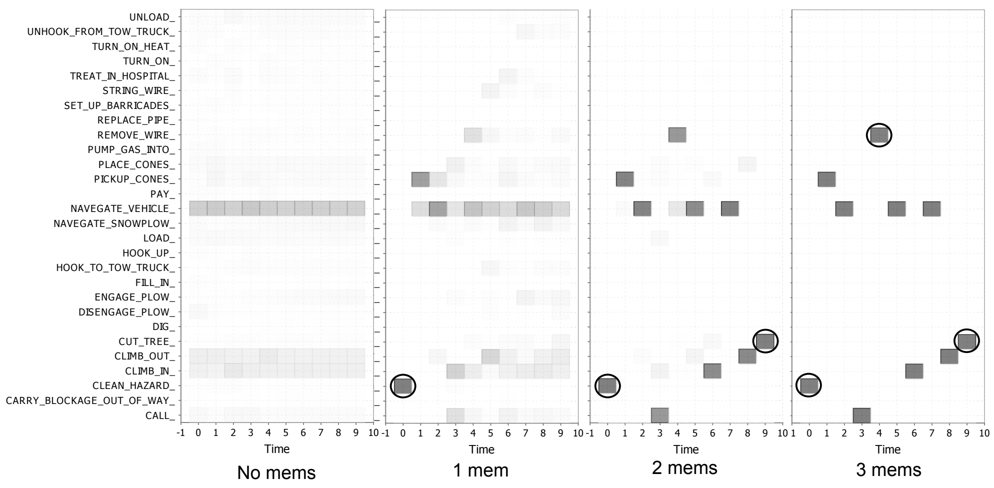


Figure 6.9: Recall of observation probabilities for 10 time steps in AHMEM$^1_8$ + $RMaxS$ model with an increasing number of mems used to reconstruct the sequence. Level of gray indicates probability of each atomic action at that time step. The darker the color is the more probable the action is. The first figure shows $P(O_t)$ for each time step in the remembered sequence when only schemata are used, this is what the model would answer after forgetting all mems; the second shows $P(O_t|O_0 = CLEAN\_HAZARD)$, recall with one mem; the third $P(O_t|O_0 = CLEAN\_HAZARD, O_9 = CUT\_TREE)$ and the fourth $P(O_t|O_0 = CLEAN\_HAZARD, O_9 = CUT\_TREE, O_4 = REMOVE\_WIRE)$. The mems are marked by circles, all the other values are derived from the schema that is most probable given the recalled mems. Probability distribution for all 28 possible atomic actions is shown, even though some have zero probability at every time step.

6.2 Prototype DyBaNeM Connection to an IVA

6.2.1 Aim

In this section we demonstrate DyBaNeM's applicability to the domain of IVAs. We connected DyBaNeM to an IVA whose behavior resembles a background character from a MMORPG. The domain in this section is more limited than the Monroe corpus used in the previous section. Therefore we will not perform detailed tests as in the previous section. Instead we will analyze in detail how DyBaNeM behaves in recall of one example day. We will show:

- That DyBaNeM can learn meaningful episodic schemata even in this domain. For instance, compared to the Monroe corpus beginnings of "days" in this corpus are much more deterministic and episodic schemata encode this knowledge.
- How DyBaNeM stores and recalls one day of the IVA's activity. We will compare differences in recall with two probabilistic models, CHMM^1 and AHMEM$^1_2$.
- How DyBaNeM supports the dialog enhancing features discussed in Sec. 4.7.
- That the method has reasonable computational time requirements given domains of moderate complexity, even though the problem of exact inference in Bayesian Network is exponential in the network's treewidth.

6.2.2 Dataset

As an input for the EM model we used corpora generated by Čermák (2013). The corpora contains 23 "days" of simulated IVA hierarchical activity. The simulated days consist of up to 36 activities which will be used as observations in our model. The IVA was controlled by hierarchical DMS based on AND-OR trees formalism. An AND-OR tree describes decomposition of an IVA's behavior into goals and subgoals with possible alternatives of accomplishing each goal. The IVA's DMS also includes nondeterministic scheduling algorithm that creates a desired schedule for each day. However, external interruptions from the 3D simulation may block completion of some goals. For instance, the IVA may miss lunch even though he initially planned it. Therefore there are two sources of nondeterminism, the scheduling algorithm and the 3D simulation.

The IVA is connected to a 3D virtual environment of Unreal Tournament 2004 (Epic Games, 2004). The simulation is run in a modified version of a map DM-UnrealVille[4]. The agent was implemented in Java and the Pogamut plat-

[4]DM-UnrealVille homepage, URL: http://www.gamemaps.com/details/100 [9.5.2015]

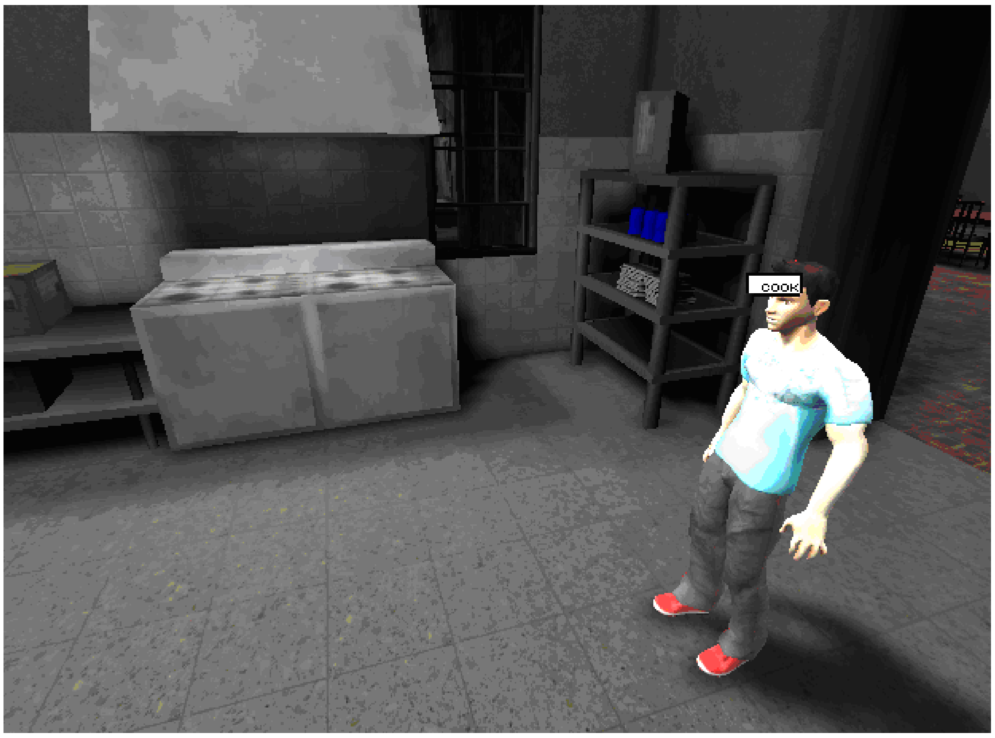


Figure 6.10: Screenshot of a simulation showing the IVA performing an action *COOK* in a virtual kitchen. The action is not animated in detail in our simulation. The screenshot is rendered using UT2004 (Epic Games, 2004).

form (Gemrot et al., 2009) was used as a middleware for interfacing the IVA with the environment.

Details of the IVA's DMS are provided in (Kadlec et al., 2013) (sec 3.2) and in (Čermák, 2013). This DMS is an extension of a work previously done by Burkert (2009).

Every simulated day has a similar structure, the IVA gets up at home, he brushes his teeth, washes his face, goes to the toilet; then he usually goes to work; in the evening he may go to a theater or to a pub. He may also do the shopping, clean the house and other activities resembling a normal life. In total the simulation contains 37 different types of atomic actions and 19 types of first level episodes (for a complete list of actions and episodes see the labels in Fig. 6.11). This matches the daily routine of a background NPC from a MMORPG.

The generated stream of actions contains more levels of episodes but for this evaluation we use only the first level of episodes which is sufficient for demonstrating all the features discussed in Sec. 4.7[5]. There are different plans for work-

[5]Experiments with multiple levels of episodes are described in (Kadlec and Brom, 2013b).

ing days and for weekends, which increases variability of the the IVA's episodic schemata. Not all days contained the same number of the atomic actions, the longest one has 36 actions. To make all days equal in size we added a sufficient number of padding actions DAY_END to the end of each day. One downside of this representation is that we lose mapping from indexes of time slices of the DBN to real time. For instance, the 11th time slice can represent 11:30 am in one day and 12:45 pm in another day. An advantage of this representation is that day descriptions are more compressed, thus allowing for faster inference.

An alternative approach might be to have fixed mapping from real time to time slices of the DBN. In this approach all days will be represented by the same number of time steps. Therefore it will not be necessary to pad the days with the extra symbol. Since the dataset used in this experiment does not contain information about duration of actions measured in real time we have to adopt the previous time representation.

Note that actions in this dataset correspond to well defined activities like walking or washing hands. We do not use granularity of individual animations. For instance, our action $WALK$ might be further decomposed into a sequence of steps interleaved by short stops. However, this would dramatically increase the number of actions representing each day. Therefore we choose granularity that is meaningful and still computationally inexpensive.

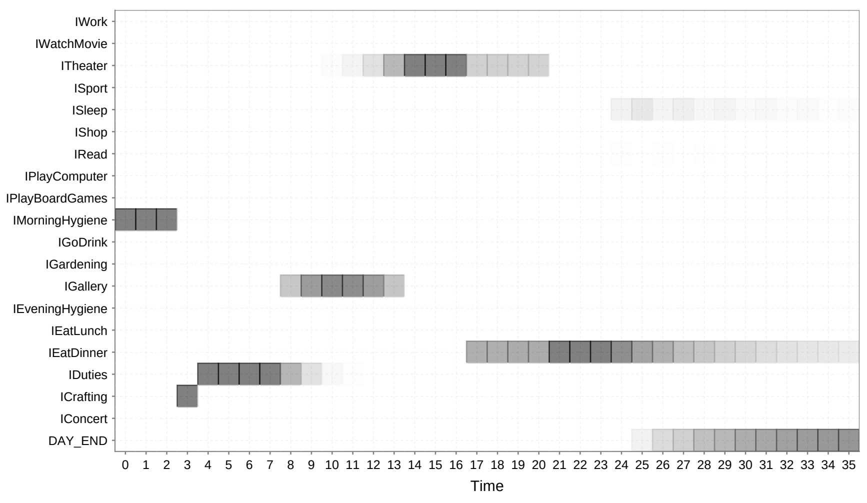
6.2.3 Method

Since the size of this corpora is relatively limited we use different testing schema than in the previous experiment. Here we use all 23 days to learn the episodic schemata. After that we encode one day of activity with respect to this schemata. Note that the same day was already used to train the schemata (together with another 22 days).

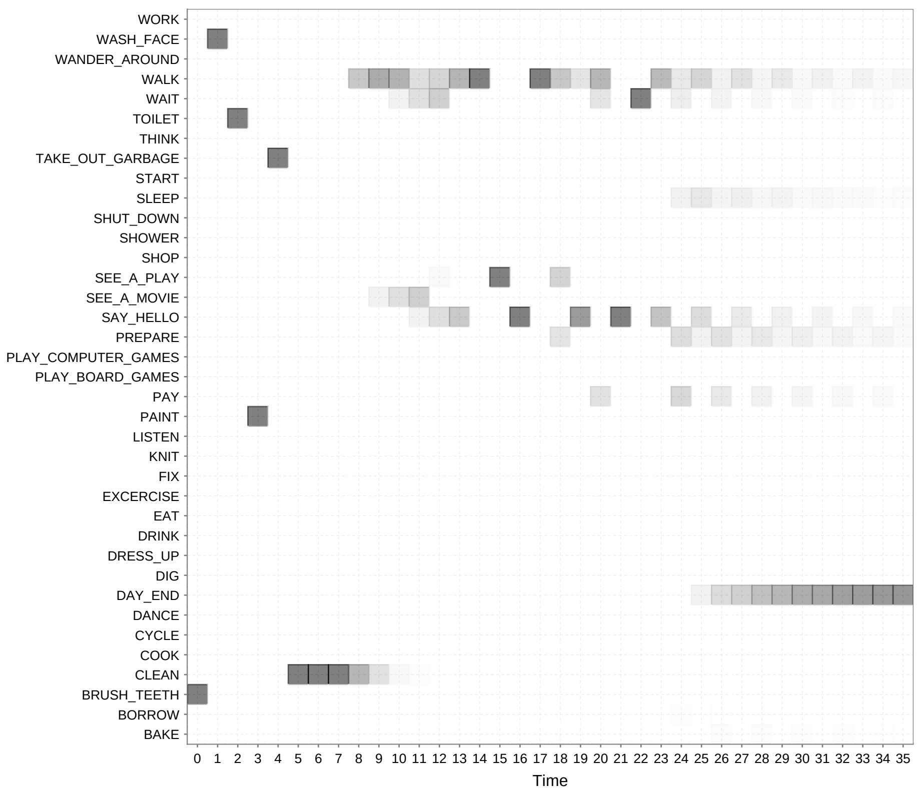
This setup assumes that all 23 days were stored in detail in some temporary store. Then DyBaNeM learns the episodic schemata for all days and it computes mems for all days with respect to these schemata. After that the temporary store might be erased. This setup has also its justification, it might be viewed as a lossy compression for episodes. Or in other words, we assume that the agent creates a stand alone episodic schemata for those 23 days and he encodes the days relative to his schemata.

We will test both CHMM^1 and AHMEM$_2^1$ on this task. However, we will report only results with the RMaxS since it is better suited for real time use as was demonstrated in the previous experiment. We will compute three mems for the stored day as in the case of the Monroe experiment.

Steps needed to re-run these experiments are described in Appendix A.4.

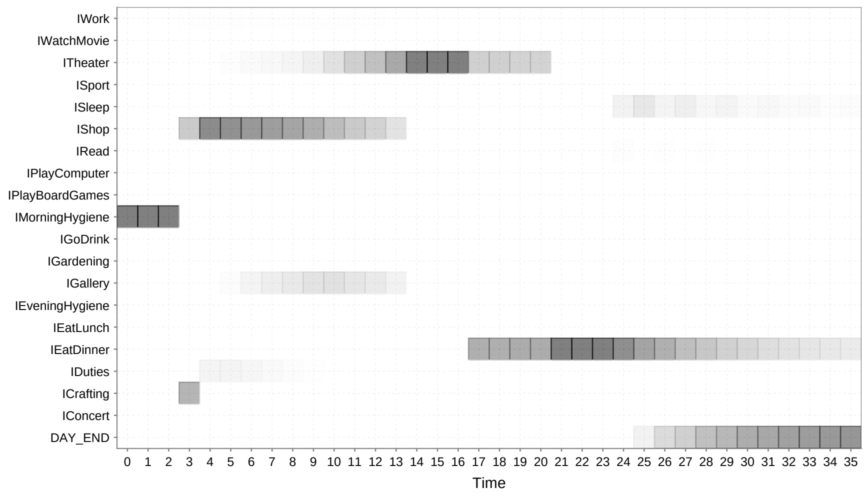


(a) Level of episodes

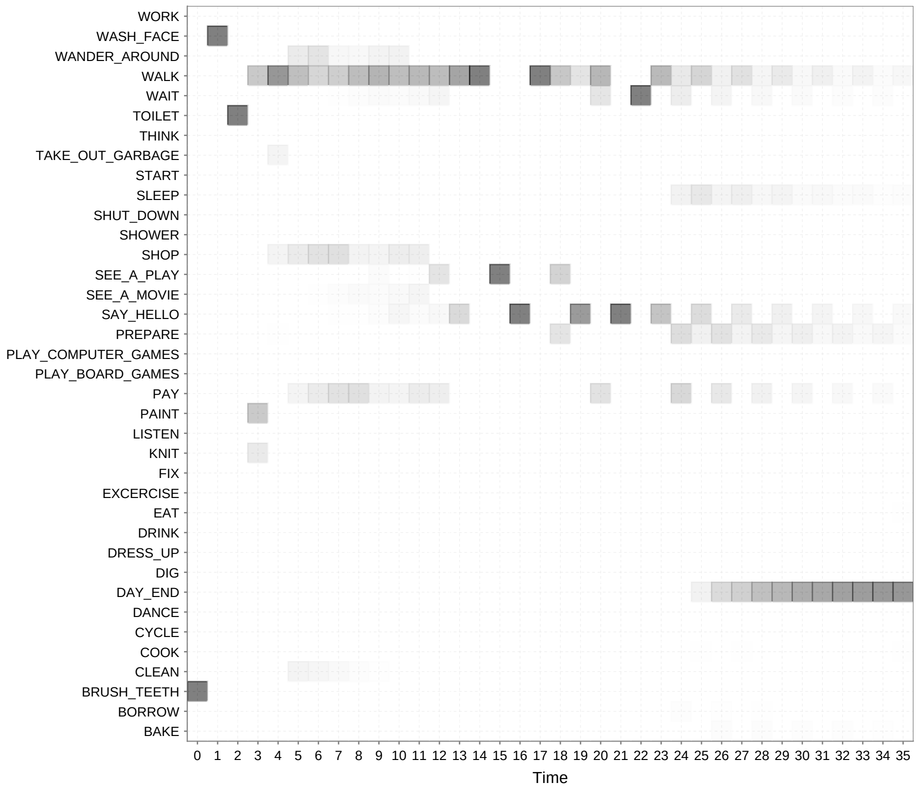


(b) Level of observations

Figure 6.11: Recall of the stored day in AHMEM when three mems are used for reconstruction.

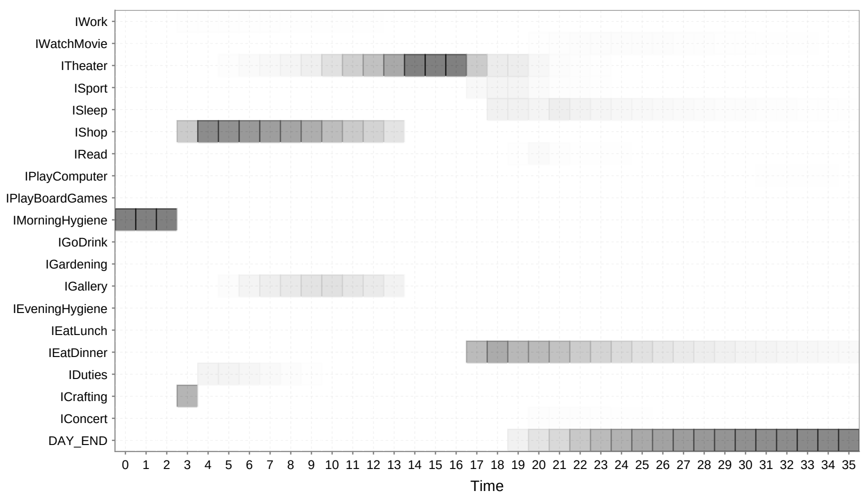


(a) Level of episodes

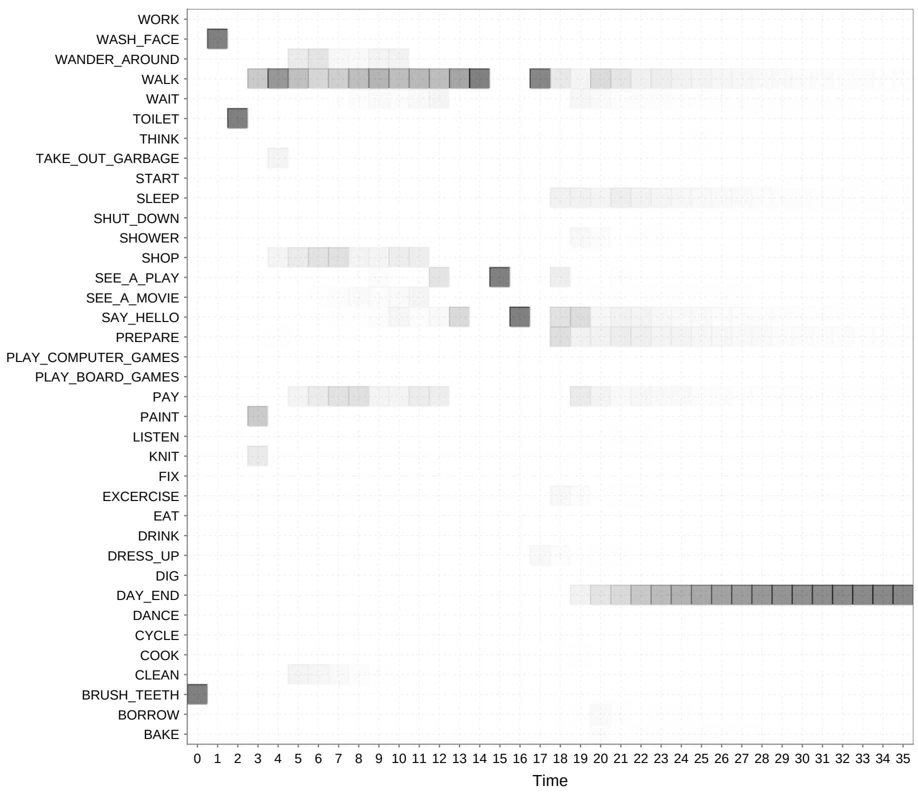


(b) Level of observations

Figure 6.12: Recall of the stored day in AHMEM when two mems are used for reconstruction.

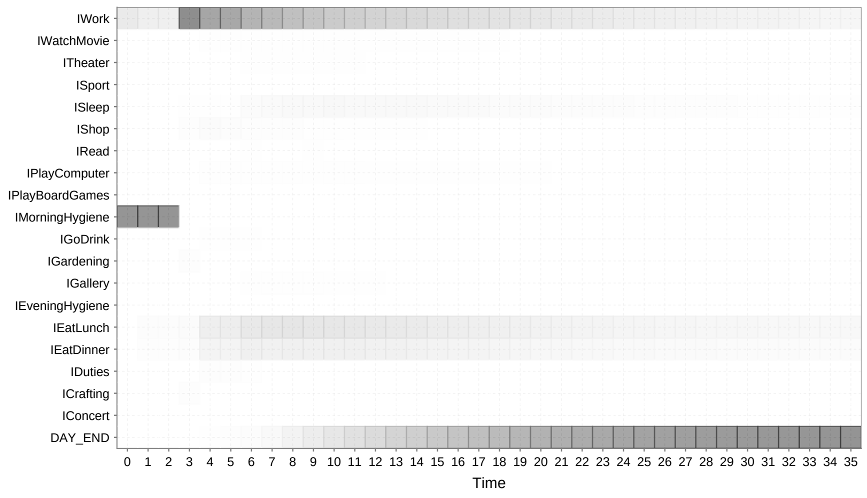


(a) Level of episodes

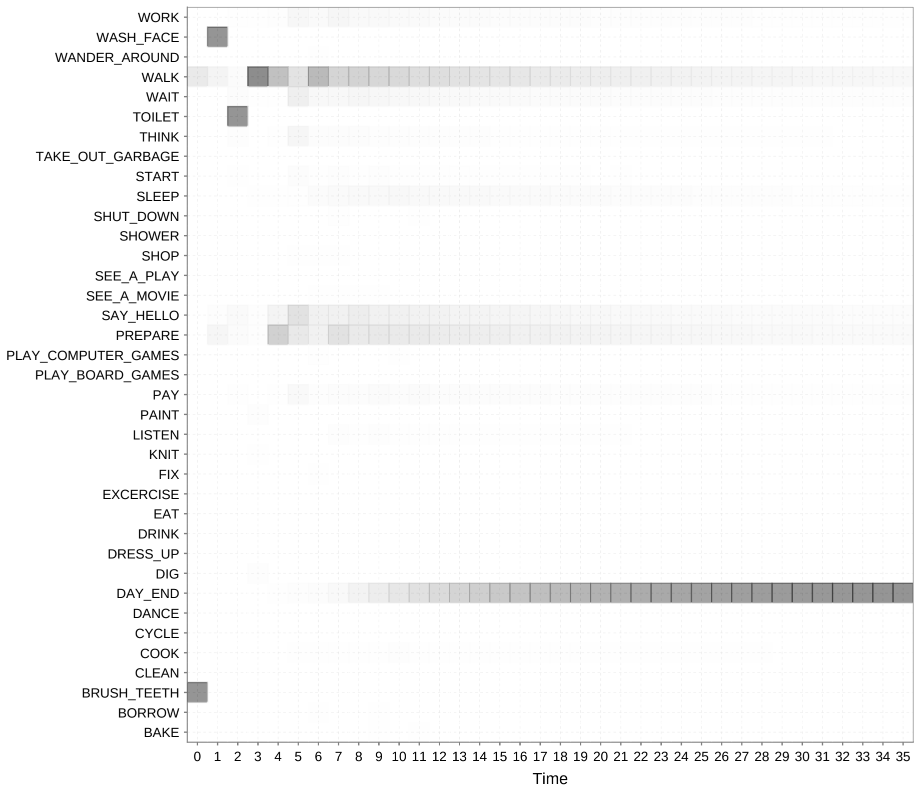


(b) Level of observations

Figure 6.13: Recall of the stored day in AHM*E*M when one mem is used for reconstruction.

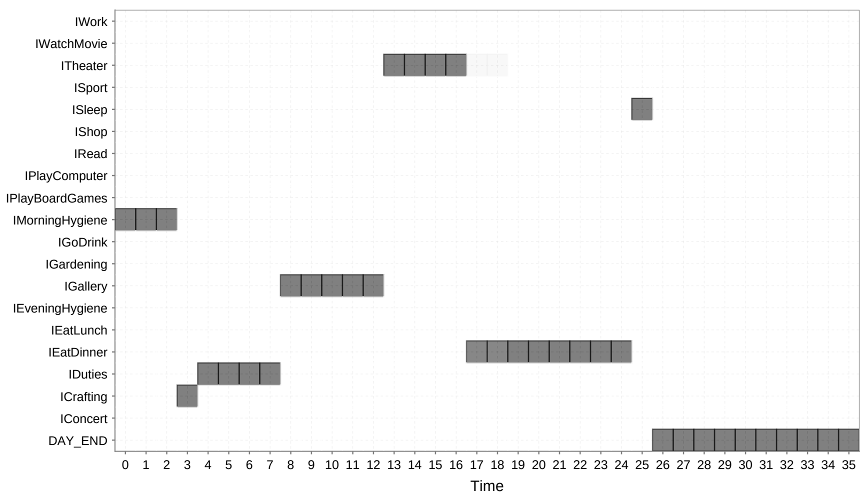


(a) Level of episodes

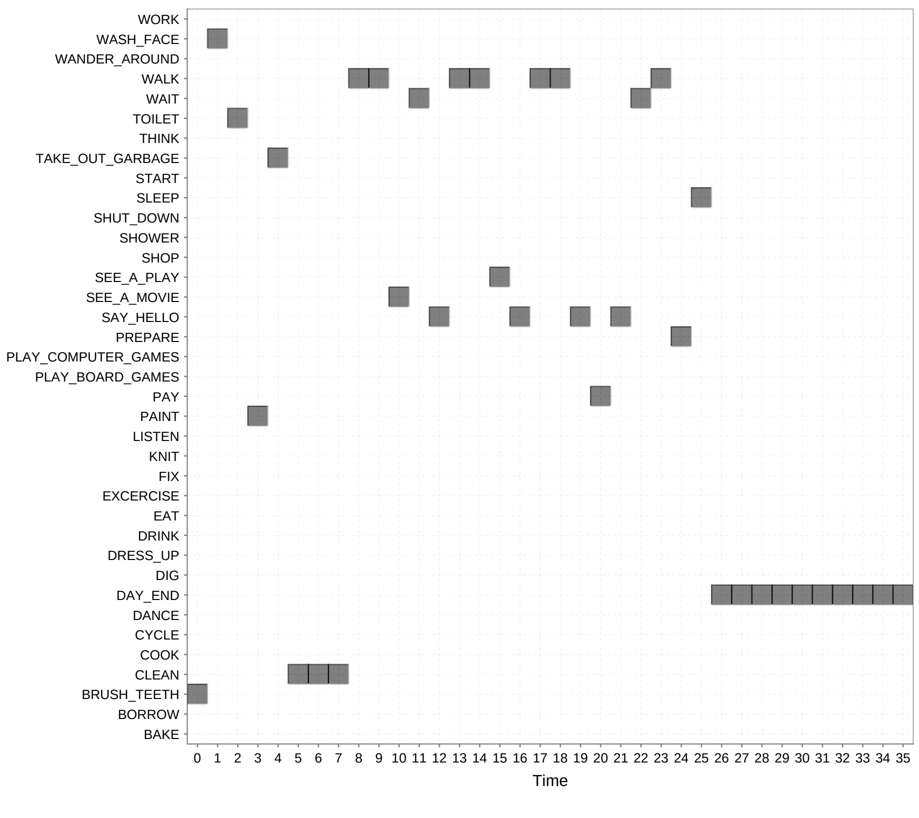


(b) Level of observations

Figure 6.14: Recall of the stored day in AHM*E*M when no mem is used for reconstruction.

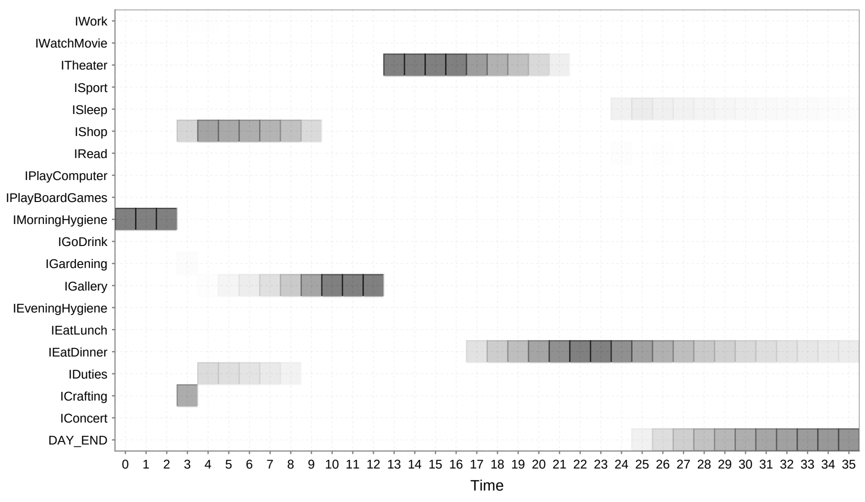


(a) Level of episodes

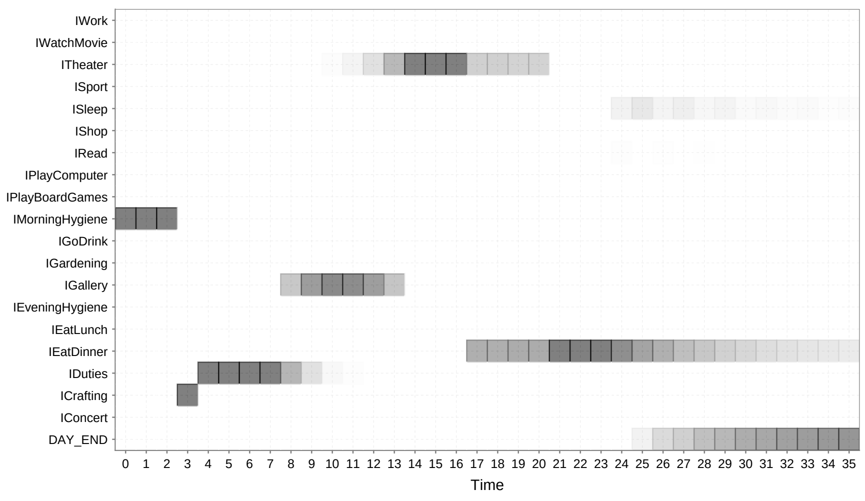


(b) Level of observations

Figure 6.15: Ground truth of the stored day.



(a) Level of episodes in CHMM^1



(b) Level of episodes in AHMEM$_2^1$

Figure 6.16: Comparison of recall on the level of episodes when three mems are used. AHMEM$_2^1$ correctly recalls the *IDuties* episode whereas CHMM^1 forgets it.

6.2.4 Results

Learning the schemata in AHMEM$_2^1$ took around 15 seconds. Computing each of the three mems took at most 0.1 of a second. Learning the schemata in CHMM^1 took around 1 second. The time needed to compute each mem was roughly the same as in AHMEM$_2^1$.

On the level of observation the accuracy of both models is the same. When using only one mem to reconstruct the whole day, 58% of atomic actions were correctly recalled, with two mems it was 69%, and with three mems 80%. However, on the level of episodes, AHMEM$_2^1$ performed better than CHMM, see Table 6.6. Three mems computed by AHMEM$_2^1$ are $O_{15} = SEE\_A\_PLAY$, $O_{22} = WAIT$, $O_{5:7} = CLEAN$. The mems computed in CHMM^1 are $O_{15} = SEE\_A\_PLAY$, $O_{10} = SEE\_A\_MOVIE$, $O_{22} = WAIT$.

Figures 6.11 — 6.14 show recall on the level of episodes and observation in AHMEM$_2^1$ when three mems (6.11), two mems (6.12), one mem (6.13) or no mem (6.14) were used to aid the recall. Since on the level of observations both models perform the same we omit the same graphs for CHMM^1. However, we compare recall in AHMEM$_2^1$ and CHMM^1 in the case when three mems are used since here the models differ on the level of episodes, see Figure 6.16. Text summary of the recall in AHMEM$_2^1$ is in Table 6.7.

Figure 6.15 shows ground truth of the stored day.

	1 mem	2 mems	3 mems
AHMEM$_2^1$	63.9%	66.7%	88.9%
CHMM^1	58.3%	69.4%	72.2%

Table 6.6: Accuracy on the level of episodes for recall of the 23rd day in the two compared probabilistic models.

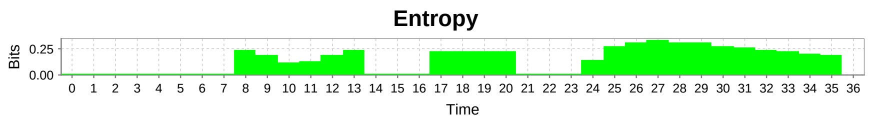
6.2.5 Discussion

The evaluation indicates that computational cost is reasonable. Learning the schemata is done only once off-line and time necessary for computing three mems (0.3 s) is also acceptable. Note that this time might be decreased by using approximate inference techniques. Figures 6.11 and 6.17 illustrate how DyBaNeM fulfills the requirements from Section 1.3 that were further discussed in Section 4.7. These requirements are: **1** — Summarization; **2** — Clarifying questions; **3** — Expressing degree of certainty; **4** — Believable mistakes in recall; **5** — Recall of interesting details first.

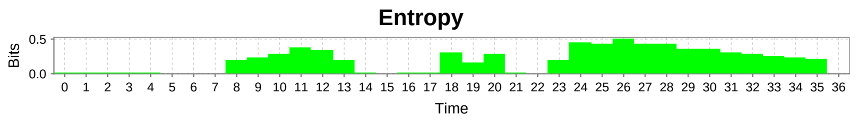
Figure 6.11a shows all high level episodes recalled for the day of interest. Level of gray indicates probability of each atomic action/episode at that time

Mems	Fig.	Verbalized recall
3	6.11	Morning routine. Crafting. Household duties. Gallery visit. Theater. Dinner.
2	6.12	Morning routine. *Maybe crafting. After that shopping and possibly gallery visit.* Theater. Dinner.
1	6.13	Morning routine. Maybe crafting. After that shopping and possibly gallery visit. Theater. *Dinner, but I do not remember any details about it.*
0	6.14	Morning routine. *Work. Some time during the day there was lunch and dinner.*
Truth	6.15	Morning routine. Crafting. Household duties. Gallery visit. Theater. Dinner.

Table 6.7: Hypothetical verbalized recall of high level events based on Figures 6.11 – 6.14. Text in italics highlights where the recall differs from recall when using one more mem. The list line shows ground truth that is the same as recall with three mems. The modal verb "maybe" is used whenever there are multiple possible episodes with high probability.



(a) Level of episodes



(b) Level of observations

Figure 6.17: Entropy of recall in AHMEM$_2^1$ with three mems.

step. The darker the color is the more probable the action/episode is. This summarization capability fulfills requirement **1**. Entropy depicted in Figure 6.17 shows how certain the IVA is about his memory for those events (requirement **3**). The more alternative recalled actions/episodes there are for each time step the higher the entropy is. Figure 6.11b shows probability of all atomic actions. This is the second level of hierarchy that allows for clarifying questions (requirement **2**). Requirement **4** is met, e.g., by "fuzzy" transition around time 9: the model is not sure when exactly the switch from household duties to a gallery visit happened. Fulfilling requirement **5** follows from the design of the encoding algorithm. It is manifested by the fact that none of the created mems is in the beginning of the day (recall that the three mems for this sequence computed in $\text{AHM}E\text{M}_2^1$ are $O_{15} = SEE\_A\_PLAY$, $O_{22} = WAIT$, $O_{5:7} = CLEAN$). All mornings are the same (brush teeth, wash face and go to the toilet). Therefore there is no surprise associated with observing this sequence and they can be reconstructed from episodic schemata. Now suppose that an observing agent (Bob) should recall only the single most interesting detail about an observed agent's day (Alice). Bob would tell us that Alice went to see a play, which is the content of the first mem. Details about Alice's morning routine would not be considered interesting by Bob. However, he would still be able to guess what Alice did in the morning.

Similarly to the experiment with the Monroe dataset we can inspect the effect of forgetting for recall of this particular day. Figures 6.12 — 6.14 show how recall in $\text{AHM}E\text{M}_2^1$ degrades as fewer mems are used as evidence, Table 6.7 summarizes the recall in text.

We can see how the details are gradually forgotten. When only two mems are used for recall the household duties episode ($IDuties$) would be completely forgotten. Instead of this the IVA would incorrectly recall the shopping episode (similarly to CHMM^1 with three mems, see Fig. 6.16).

Using only one mem results in forgetting details of the evening. Predictions of atomic actions around $t = 20$ are now much more "blurred".

When using no mems at all the model has to base its recall solely on the episodic schemata. Since the agent which generated the dataset used to train the schemata was going to work five days a week the schema for an average day consists mainly of getting up, working and having lunch and dinner during the day. In this case only the morning routine would be recalled correctly.

Note that when CHMM^1 is used the $IDuties$ episode starting at $t = 4$ would not be recalled when only three mems are used for recall, see Figure 6.16. CHMM^1 would incorrectly output $IShop$ as the most probable episode, $IDuties$ would be the second most probable guess. $\text{AHM}E\text{M}_2^1$ does a better job in this case, see Fig. 6.16.

6.3 Summary

Both the experiments presented in this chapter have proved applicability of DyBaNeM framework to the domain of EM modeling in IVAs. The framework addresses the initial requirements and it has acceptable computational requirements even in the prototype implementation that is not optimized for speed. The first experiment thoroughly compared several DyBaNeM variants on relatively large activity corpora. It also examined the effect of using increasingly more training data. In the second experiment we tested applicability of DyBaNeM in a different domain with only a limited amount of training data. We also performed qualitative analysis of gradual forgetting of one day.

Chapter 7

Discussion

This chapter discusses how DyBaNeM should be used by a third party developer willing to extend his application with EM capabilities. Further we discuss that it is not clear whether Bayesian statistics is the right theoretical basis for explaining human behavior. We also show how DyBaNeM can be interpreted as a lossy compression algorithm for episodes and what class of probabilistic models can be used in DyBaNeM instead of AHMEM and CHMM.

7.1 Third Party Developer's Perspective of DyBaNeM

This section describes steps that have to be taken by a third party developer to extend his application (e.g., an IVA) with DyBaNeM. It is a cookbook that can be used by the developer without deep knowledge of DyBaNeM's internals.

The first question is whether DyBaNeM is the right tool for the developer's needs. Section 2.3 lists several models that have already been applied to virtual agents and they might be reasonable alternatives to DyBaNeM. DyBaNeM's strength is that it 1) accounts for uncertainty in observations; 2) automatically recognizes high level behavior of the other agents; 3) reconstructively recalls the past episodes as a mixture of exact knowledge about the past stored in mems and a general schema of activities, thus some events may be filled from the schema and this can create false memories (Loftus and Pickrell, 1995; Brainerd and Reyna, 2005).

Now we will sketch the steps than a developer has to take in order to connect DyBaNeM with his own agent.

1. Decide encoding of the inputs:
   (a) First, the developer has to specify what properties of the environment will be made available for the encoding and on what granularity they

will be sampled. Observable properties are bound to the O_t random variables in the probabilistic model. Thus this step includes specification of the domain for O_t. The domain can contain only actions of an observed agent as in the experiments in Sections 6.1 and 6.2. However, observations can have more complex internal structure. For instance, each observation can be tuple $\langle action, used\ object\rangle$ and the structure of the graphical model might be adjusted accordingly. There might be two variables O_t^{action} and O_t^{object} and possibly a probabilistic model $P(O_t^{action}, O_t^{object})$ binding those two. Discussion relevant to this topic is in Section 4.1.5. Note that extending the probabilistic model might increase its computational complexity. Concerning granularity of the represented observations the best practice is to omit the uninteresting details and represent only actions that might be important for the IVA's behavior. For instance, representing every single step as an atomic action is probably unnecessary. The whole sequence of steps might be replaced by a single $WALK$ action. In general this leads to a discussion on how to map time, which is continuous, to a set of finite observations that are used as inputs of the probabilistic model. In the experiment from Section 6.1 we mapped each "day" to a fixed length sequence of observations. In the second experiment (see Section 6.2.2) we used a different strategy where every day was represented by a variable number of observations and the sequences were padded by a special symbol to have the same length. Another important decision is how many levels of episodes will be stored. A single level of episodes is a reasonable starting point since it introduces a simple hierarchy in behavior and the model is still relatively fast to compute.

(b) Second, decide how long the sequence of actions to be stored in the model will be. The sequence might contain a whole day or only several hours. The encoding algorithm uses the assumption that all observations are stored in an STM and then they are all encoded at once. The advantage of using a longer time window (say one day) is that all events that occurred in this day will be considered at the same time in the encoding algorithm. It is possible that an observation made in the morning (i.e. $PHONE\_CALL\_WITH\_FRIEND$) has an impact on events in the evening ($CINEMA\_VISIT$). Even though capturing these long term dependencies in the probabilistic model might be difficult it is still possible (AHMEM can in principle express this dependency). When one uses the whole day as a time window then both the encoding and retrieval processes can take advantage of this dependency. However, if the dependent events fall into two different time windows this ability disappears. The disadvantage of using a longer time window is that exact Bayesian inference over longer sequences quickly becomes intractable. Sometimes several observations can be grouped together thus effectively shortening the sequence. Still the developer has to weigh the pros and cons of these conflicting tendencies.

2. Implement the storage mechanism. Once the mems for the last time window are computed they have to be stored somewhere and indexed with cues used in later retrieval. DyBaNeM does not come with any storage mechanism thus it is the developer's decision whether to use in memory data structures like hash maps or some persistent store implemented as, e.g., an SQL database. DyBaNeM does not cover this phase since it may vary considerably depending on the application's needs. IVAs running for a long time may need a complex database system, whereas for simpler agents in memory hash maps might be sufficient. It should be noted that forgetting what takes place during storage must be also implemented by the end developer. For instance, the developer can decide that no forgetting dependent on time passed is desirable. The agent simply computes N mems for each time window and those mems remain intact. A different option might be exponential forgetting, as discussed in Section 4.3. An efficient way of implementing this type of forgetting is discussed by Derbinsky and Laird (2012).
3. Decide which cues will be used for indexing of the created mems in the permanent storage. This can be a time when the episode happened, external context, etc. The cues should be determined by requirements on the retrieval. For instance, when the agent will mostly answer questions like: "What were you doing X time units ago?" then the mems have to be indexed by time.
4. Obtain episodic schemata. The schemata can be either directly specified by an expert (the expert specifies a probabilist model by hand) or preferably they can be learned from annotated example episodes. Details of learning in Bayesian models are discussed in Section 4.1.3. Annotating the example episodes by hand might be a time consuming task. Fortunately there is one convenient alternative unsupervised method for obtaining the annotated data. One can exploit the agent's own DMS as long as it uses hierarchy to express the agent's behavior (popular hierarchical DMSs are listed in Section 2.6). For every time step one can log trace through the agent's currently executed hierarchical goal (e.g., $COMMUTE \rightarrow WAIT\_FOR\_BUS \rightarrow READ\_NEWSPAPERS$) and also store all the necessary observations from the environment (e.g., $\{central\_station, the\_times\}$). Then the episodic schemata can be learned from this log created by the agent itself. A similar

approach was already used in (Berzan and Scheutz, 2012). The downside is that the agent can only recognize episodes that are coded in his DMS.

5. Pick the probabilistic model used in DyBaNeM and the encoding strategy (RMinOS or RMaxS). The models used in the experiments are CHMM and AHMEM. One can start with the more complex AHMEM. If it later shows that the computation is too demanding (e.g., it does not meet real-time requirements) then one should try the less expressive CHMM. Another reason for picking CHMM might be that it generalizes better when there are only limited training data. See results and discussion of the experiment from Section 6.1. However, DyBaNeM can work with other types of probabilistic models as well, for review of suitable models and deeper discussion of this topic see Chapter 7.4. Based on the results of our experiments, using AHMEM with RMaxS seems to be a reasonable starting point.
6. Decide when the storage and retrieval take place. For instance, an IVA can buffer all observations from one day in a short term store. Then during the "night" encoding algorithm computes mems describing the day. Retrieval can be initiated when the IVA is asked to provide details of a specific day or when the IVA needs to get some information for its internal decision making.
7. If the model is too slow go back to step 5 and try a simpler probabilistic model or use some approximate technique for inference in DBNs. Luckily the inference engine SMILE used by DyBaNeM supports several exact and approximate inference algorithms out of the box. Namely these are junction tree algorithm (Huang and Darwiche, 1996), Pearl's algorithm (Pearl, 1986), logic sampling (Henrion, 1988), likelihood sampling (Fung and Chang, 1989), backward sampling (Fung and Favero, 1994), AIS (Cheng and Druzdzel, 2000) and EPIS (Yuan and Druzdzel, 2003). However, accuracy of the approximation algorithms has to be carefully assessed in each domain.

Summary of the overall system and modules provided by DyBaNeM and those that have to be implemented by the developer is in Fig. 7.1.

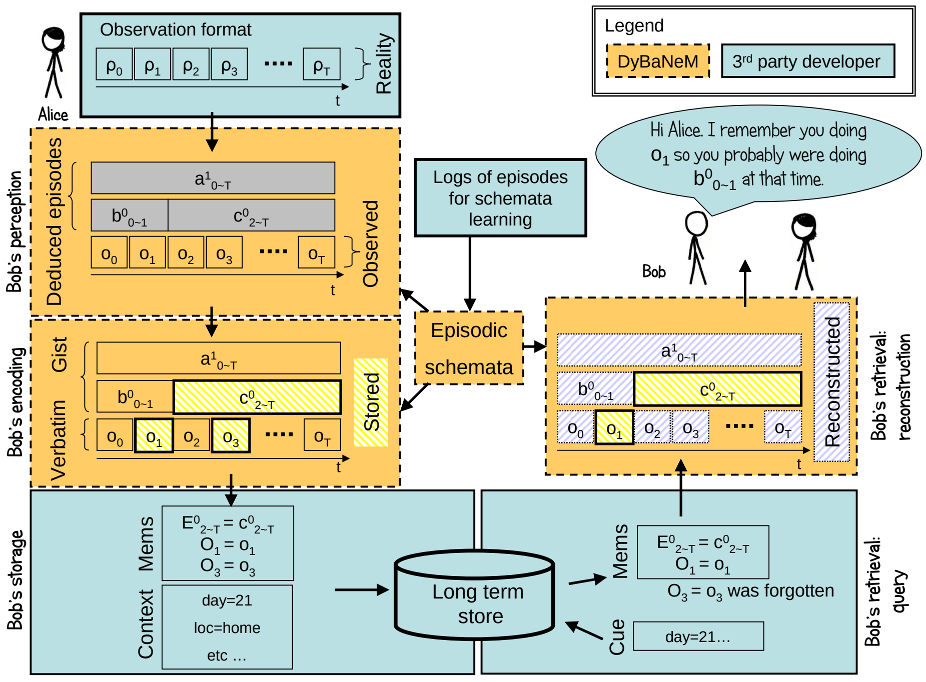


Figure 7.1: Third party developer's perspective of DyBaNeM. Dashed orange modules are provided by DyBaNeM, solid light blue components have to be implemented by the third party developer. This figure is an extension of Fig. 3.1 where you can find additional details.

7.2 Is Memory Bayesian?

One of the core assumptions of DyBaNeM is that the memory encoding and retrieval processes follow laws of Bayesian statistics. It is questionable whether human memory works this way. Even though Bayesian models proved to be useful in the cognitive science community (for a selection of papers inspired by Bayesian statistics see, e.g., (Doya et al., 2007; Chater and Oaksford, 2008), or see FTT discussed in Section 1.4) there are still phenomena that are better described by different theories. If we take analogy from human decision making, it is well known that humans do not act rationally (that is, following Bayesian rules), they rather use some heuristics (Kahneman et al., 1982). A prominent example of this general approach might be the prospect theory (Kahneman and Tversky, 1979). In recent years there have been attempts to find more coherent explanations of these heuristics, e.g., by using mathematical theories applied in quantum physics (Busemeyer et al., 2011).

Similarly, the RMaxS and RMinOS strategies that try to model rational rules for mem selection are only approximations of reality. Additionally, even the reconstructive process employed in activity recognition and memory retrieval (computing $P(episodes|observations)$ and $P(episodes|mems)$ respectively) might be in fact driven by some heuristics and not by Bayesian computations as is assumed in DyBaNeM.

7.3 Compression Algorithms Analogy

As already noted in Section 1.4 the Fuzzy-Trace Theory (FTT) (Brainerd and Reyna, 2005) states that human memory uses two distinct modes for encoding episodic memories. The vivid detailed memories are encoded as verbatim, whereas less detailed memories take the form of gist, a general schema of the encoded episode. For instance, gist might be *professor's office* and verbatim can be *photography on professor's table*. This leads to the idea that EM may be viewed as a natural type of lossy compression algorithm for events. There are events that we can recall in great detail, e.g., the terrorist attacks on 9/11. This kind of experience is referred to as a *flashbulb memory* (Brown and Kulik, 1977). On the other hand other more recent but probably less important events are forgotten much faster. One probably recalls what he/she had for dinner yesterday, but recalling a menu from last week might be a much harder task. This gradual forgetting guided by time passed and importance of events might be a mechanism that humans have developed throughout evolution and which helps us retain the important events and at least reasonably guess about the others ("I probably had pizza, I used to have pizza on Tuesdays."). This forgetting process is closely related to the lossy compression algorithms from computer science.

In this section the link between compression algorithms and DyBaNeM will be outlined. First, types of compression algorithms for different types of data will be reviewed, second, DyBaNeM will be presented as a type of lossy compression algorithm for stories.

7.3.1 Types of compression

Generally, a compression algorithm works as follows: an input I is transformed by the encoder to a compressed representation C, that is later decoded to output O. The goal of the compression is to create a compressed representation such that $Size(I) > Size(C)$. When $I = O$, we call the method lossless compression. On the other hand, when $I \neq O$ but O "is close" to I, we call it lossy compression. Some details of the input may be lost during the compression but the output O should resemble the original.

There are lossless compression algorithms that can compress any type of data, be it a text, audio, an image or video. This category includes basic algorithms

Data type	Algorithms
Image	JPEG, PNG
Video	MPEG
Audio	MP3
Text/Stories	No standard method

Table 7.1: Lossy compression algorithms for different types of data.

like run-length encoding (RLE)[1] or dictionary based LZF (Ziv and Lempel, 1978). A different family of lossless algorithms are those based on the arithmetic coding (Moffat et al., 1998) like prediction by partial match (PPM) (Bell et al., 1989) and its various recent extensions, e.g., PAQ6 (Mahoney, 2005). These are among the best performing lossless algorithms when focusing on the compression ratio.

On the other hand, there is no general purpose lossy compression algorithm. Lossy algorithms use characteristics of the compressed data and the way humans perceive this particular type of data. For instance, algorithms for video compression may use facts that: 1) two consecutive frames will be probably similar; 2) slight changes in color will be hardly perceived. Similarly, in audio compression, cropping high frequencies has only a little impact on the decompressed output. Table 7.1 shows examples of algorithms for different types of data. The interesting fact is that there is no standard method for lossy compression of episodes.

If we think of a hypothetical episode lossy compression algorithm, one may see a connection to text summarization. Its goal is to either rephrase the given text or just pick the most important sentences (e.g., TextSum algorithm (Erkan and Radev, 2004), for review see (Das and Martins, 2007)).

7.3.2 DyBaNeM as a Lossy Compression Algorithm for Activities

In this section, we argue that DyBaNeM can be seen as a type of lossy compression algorithm for episodes. DyBaNeM gets an observed episode on the input and transforms the episode into a list of mems that is shorter than the original episode. With the use of the episodic schemata the mems can be used to reconstruct the episode. However, some details of the episode might be changed due to forgetting and imperfect schemata. The core assumption is that the surprising events have more information than the less surprising ones. For instance, if every day begins with the same routine, it makes sense to store the first event that deviates from the most probable sequence of events.[2]

[1]RLE: URL: `http://en.wikipedia.org/wiki/Run-length_encoding` [31.12.2013]

[2]This is a similar usecase as featured in "5. Measuring interestingness of events" from Section 4.7.

The difficulty in interpreting DyBaNeM as a compression algorithm is that not only mems but also the episodic schemata θ have to be stored (or transmitted). Episodic schemata can be seen as a counterpart of a dictionary in dictionary based algorithms like LZF. Even though the exact content is different, they play the same role. Since storage of θ requires far more space than storage of one mem this approach is feasible only if a large number of episodes will have to be stored. The method can be considered as compression only if the following statement holds:

$$Size(original\ episodes) > Size(episodic\ schemata) + n * Size(one\ mem) \tag{7.1}$$

Where n is a number of stored mems. Now imagine that Bob wants to re-tell the story to Alice. Bob can tell Alice every even uninteresting detail. Or he can tell her only the interesting details and hope that Alice has the same episodic schemata as him and therefore she can reconstruct the whole story. This can be true in the real world where people from the same culture can have almost the same schemata due to cultural influences[3]. Thus when Bob encodes the episode with respect to his own schemata and subsequently he tells her only the content of these mems Alice can use her schemata to fill in the missing events. Only a small number of falsely reconstructed events should emerge. Bob picks only the interesting details and he knows that the rest of the episode follows from his own schemata and thus also from Alice's ones.

In some usecases the limitation of storing the schemata might be only a minor problem. Imagine a MMORPG with tens of thousands of IVAs. Among the IVAs there might be one thousand lumberjacks that can share the same episodic schemata. The schema will consist of getting up, going to a wood, chopping down trees, having lunch in the wood, getting the trees to a sawmill and going back home. Memories of all lumberjacks will be encoded with respect to the same schemata thus its size will be minimal compared to the aggregated size of memories of all lumberjacks. This way DyBaNeM can enhance not only the players' experience, but it can also provide benefits in the form of reduced storage space needed for IVAs' EMs.

7.4 Activity Recognition Probabilistic Model

The activity recognition module is one of DyBaNeM's main subprocesses. Its current implementation, as presented in Section 4, uses two DBN topologies: CHMM and AHMEM.

[3]For examples of cross cultural differences see, e.g., (Bartlett, 1932).

In this section we will discuss what other types of probabilistic activity recognition models can be used in DyBaNeM.

As can be seen from the review in Section 2.5 there is a wide range of algorithms applicable to activity recognition. Any of the reviewed activity recognition algorithms can be used for the activity recognition/perception phase in DyBaNeM. That is the phase where Bob observes Alice's behavior and he infers what her high level goals are (Bob computes $P(\text{episodes hierarchy} \mid \text{observation})$). However, this is not the only place where the probabilistic model is used in DyBaNeM. Besides perception, it is also used in encoding where the difference between what Bob has seen and what he would recall with mems computed so far is repeatedly measured, i.e., the difference between $P(\text{episodes hierarchy} \mid \text{observation})$ and $P(\text{episodes hierarchy and observations} \mid \text{mems})$. In the end, the probabilistic model is also used in reconstructive recall to once again compute $P(\text{episodes hierarchy and observations} \mid \text{mems})$.

The last two use-cases require usage of a model that is able not only to recognize episode hierarchy given the observation, but that is also able to regenerate the observations given incomplete evidence, i.e., a list of mems. This leads to the concept of generative and discriminative models in machine learning (Ng and Jordan, 2001).

The distinction between generative and discriminative models can be explained as follows. When translated to the probabilistic setting, generative models learn joint probability $P(X, Y)$ of inputs X and the target classes Y. Using the Bayes rule, both $P(Y|X)$ and $P(X|Y)$ can be computed. The generative models make no distinction between inputs and target classes. On the other hand, so called discriminative models learn only the distribution $P(Y|X)$. Thus for any given input x they can compute probability of the target class y. A generative model can do the same and additionally generate a random example of input x that belongs to y. In general, discriminative models have asymptotically lower prediction error, however, generative models perform better when only a limited number of training examples are available (Ng and Jordan, 2001).

In the context of activity recognition popular generative models are, e.g., DBNs, whereas discriminative models are, e.g., CRFs or neural networks.

Therefore DyBaNeM can work with any generative model presented in section 2.5. For instance, any model expressed as DBN can be used. AHMEM and CHMM used in experimental evaluation are two examples of such models, however, they can be replaced with HHMM, HSMM, CxHSMM or any other similar model.

7.5 Summary

In this chapter we have discussed steps that have to be taken by the third party developer in order to apply DyBaNeM in a new application. We touched on representation of observations and granularity of time. We emphasized that it is important to choose a model that is efficient from a computational perspective. For instance, it is advised to use the simplest possible model that fulfills all requirements of the application.

We also showed that under some assumptions DyBaNeM can be considered as a form of lossy compression for episodes. Mems store only interesting details that deviate from known schemata and the rest of the episode can be probably easily reconstructed. In the end we discussed possible alternative probabilistic models that might be used to represent the episodic schemata.

Chapter 8

Future Work

Future work on the DyBaNeM can come along several axes. These are:

1. Obtaining realistic data for schema learning.
2. Extending the core functionality.
3. Applying the model to different domains.
4. Testing the model with human users.
5. Implementing a game prototype.
6. Integrating the model in a broader agent architecture.

8.1 Data for Schemata Learning

8.1.1 Learning episode schemata from natural activity datasets

In the current experiments the episodic schemata were learned on data generated by the simulation. Streams of actions generated in these simulations are inherently restricted by the nature of behavior/plans implemented by the programmer. It would be better if the schemata could be learned on real activity corpora. However, there are not that many hierarchically annotated corpora of human activities. There are annotated corpora that come from the field of activity recognition, for instance, PlaceLab (Intille et al., 2006)[1] or Huynh's dataset (Huynh et al., 2008)[2]. The disadvantage of these datasets is that they usually contain only relatively low level activities (walking, sitting, etc.) over a limited time period. A dataset with higher level activity annotation was created in our previous work (Kadlec and Brom, 2011), this dataset contains nearly three months of data collected on a mobile phone with hierarchical annotation that uses on average

[1]PlaceLab URL: http://architecture.mit.edu/house_n/data/PlaceLab/PlaceLab.htm [15.11.2013]

[2]URL: http://www.ess.tu-darmstadt.de/datasets/tud-ubicomp08 [15.11.2013]

three levels of abstraction. An interesting source of data might be annotated videos. A recent EU funded project Codmos (Cataldi et al., 2011)[3] aimed at creating tools for semantic annotation of videos. Unfortunately besides creating the annotation tools and methodology only a few hours of video were annotated in this project.

A different option might be to use unannotated activity datasets (i.e., Nokia's Lausanne Data Collection Campaign (Kiukkonen et al., 2010)[4]) and employ some semi-supervised method for activity recognition. Thus the annotation would be the result of joint work of a computer program and a human. This way we can obtain several years of annotated activities.

8.1.2 Common sense reasoning

Common sense reasoning provides a qualitatively different approach that tries to mine activity schemata from common sense data sources available on the Internet. One of the first examples of this approach is LifeNet (Singh, 2003), which tries to translate a common sense corpora to a Markov network which is a type of undirected graphical model. Mined activity models were also used to bootstrap activity recognition in (Wyatt et al., 2005). There are general purpose engines mining relations from natural language sources like Wikipedia (i.e., ReVerb (Fader et al., 2011)[5]). The mined relations include frequency counts, thus they can be easily translated into probabilities. The final step is to figure out how to translate general relations into episodic schemata, that is, to identify relations that describe time precedence of activities and their hierarchical structure. Alternatively one could use some already existing activity ontologies (Akdemir et al., 2008; Chen and Nugent, 2009) and translate them into parameters of the probabilistic model.

8.2 Extending the Core of DyBaNeM

This section summarizes possible extensions of DyBaNeM's computational core.

8.2.1 Connecting different probabilistic models

As was already discussed CHMM and AHMEM are not the only possible DBN architectures that can be used in DyBaNeM. Thus extending DyBaNeM with any

[3]URL: `http://cadmos.di.unito.it` [12.11.2013]

[4]Lausanne Data Collection Campaign URL: `https://research.nokia.com/page/11367` [15.11.2013]

[5]URL: `http://openie.cs.washington.edu` [12.11.2013]

of the models reviewed in Chapter 7.4 might be a future work. Or even better, the structure of DBN itself can be learned from the data (Boyen et al., 1999; Abbeel et al., 2006; Campos and Ji, 2011).

8.2.2 Unsupervised schemata learning

The current experiments supposed that the training data contained not only observations but also the high level episodes. That is, all observations were hierarchically labeled with episodes. In an unsupervised setting both the labels for episodes and their starting and ending times are missing. Thus when Bob watches Alice he has to figure out when she probably finished some episode, how many episode schemata are needed to describe her activity and what are the instances of the same episode schema. There already exist algorithms for unsupervised sequence segmentation (Seldin et al., 2001) based on statistical regularities of the sequences. A different approach to segmentation in the context of activities might be to use heuristics from psychological literature. For instance, boundaries between episodes often correlate with features like change in movement direction or change in the object of the behavior (see (Zacks and Tversky, 2001) for in depth discussion). One can imagine that this heuristic event segmentation might be used in conjunction with the AHM$EM$ model. Values of H_t^i will be set from the heuristics and EM algorithm will be used to compute the most probable episode labels[6]. One assumption of the previous approach is that the number of possible activities is known in advance ($|D(E^i)|$). A further extension could be the estimation of the optimal number of episodic schemata from the training data via nonparametric Bayesian methods (Orbanz and Teh, 2010). More specifically an infinite HMM (Beal et al., 2001) might be a promising option.

8.2.3 Different recall strategies

The current model uses marginal probabilities at each time step as the basis of its recall. As already discussed in Section 4.4 there are other options, for instance MAP estimate. Alternatively one can use only random walk (Pearson, 1905) that matches constraints given by mems and it is drawn from the probabilistic distribution given by the episodic schemata. This way the agent's recall can be slightly different each time but arguably still believable. The problem of generating Markov chain random walk with additional constraints is addressed by Pachet et al. (2011).

[6]Note the learned labels will be purely abstract in this case (i.e. $EPISODE\_7$). Mapping of these abstract labels to human readable has to be done later.

8.2.4 Fuzzy time index of mems

Currently each mem has precise time information. However, this does not match our expectation of the human memory (Larsen et al., 1996). Thus time of each mem might be specified by a probabilistic distribution (for instance Gaussian) rather than an exact time index. One of our previous EM models already addressed this issue (Brom et al., 2010).

8.2.5 Transfer of episodic schemata across domains

It might also be interesting to employ techniques of transfer learning (Pan and Yang, 2010) and try to transfer schemata learned in one domain to another domain with minimum of training examples. This would decrease development time needed to connect DyBaNeM equipped IVA to a new environment. The general question is if there are some "meta activity schemata" that encode structure common to similar episodic schemata in different domains. Parameters of those meta schemata could be learned in virtually any domain with human activity. Then they could just be transferred to the target domain where fewer training examples should be needed since the schemata will be already pre-trained on a different domain. Transfer learning can help to deal with the fact that different domains use different vocabulary to describe the activities. Each domain can have a different number of atomic actions and episodes and even semantically equivalent actions might use different names (e.g., $WALK$ and GO). Transfer learning can identify the common structure in both domains (e.g., $WALK$ in domain A is the same action as GO in domain B) and thus speedup learning in the new domain.

8.3 Testing the Model With Human Users

There are two main possible approaches to testing the model with humans.

8.3.1 Fitting human data

In this approach the research task will be to compare recall of humans and recall of DyBaNeM when both are provided with the same input real life data. The model's parameters will be fitted on a training dataset and accuracy of its predictions will be assessed on the testing dataset. This would be an ideal scenario for testing any episodic memory model. However, there is one inherent drawback to this approach. Humans will have some episodic schemata that they have acquired throughout their lives prior to the beginning of the experiment. But the computational model will have to learn the schemata only on the input data. This will clearly disadvantage the computational model. One way of overcoming this limitation may be to learn the schemata on external data as discussed in Section 8.1.

To eliminate this discrepancy one could design a completely artificial environment similar to the abstract grammar learning task used by Knowlton and Squire (1996). The grammar would be initially unknown to both humans and the model, thus they would both build the schemata from the same input data. Transfer of knowledge from normal life would be impossible. The disadvantage of using an artificial domain is that humans would probably use a different memory system than the EM that they employ for long term memories from real life. This type of evaluation might be interesting for the community of computational cognitive science.

8.3.2 Measuring believability

A different experiment setup can use a modified Turing test (Turing, 1950) where human subjects and the memory model would have the same inputs and then humans will judge whether DyBaNeM's recall seems believable to them. Compared to the previous setup this one focuses more on believability of the recall than on its exact match to human data. However, believability is exactly what is required by the majority of applications utilizing IVAs.

8.3.3 Personalized memory for social services

A promising research direction could be providing the memory model with personal histories that users of social media like Facebook, Google+ or Twitter generate. Feeds of profile updates can be connected with a person's location tracked via a mobile phone, with e-mail communication, instant messaging, check-ins on Foursquare[7], etc. When these sources are combined together they create a digital diary of one's lives. More and more people are now "living in the network" (Lazer et al., 2009), meaning that they either consciously (i.e. Facebook updates) or unconsciously (i.e. logs of phone calls) create digital footprints of their life. Thus experiments similar to Wagenaar's pioneering autobiographic memory study (Wagenaar, 1986) can be now conducted on a significantly larger scale.

8.4 Applying the Model to Different Domains

Even though DyBaNeM was formulated as a model for IVAs it can be applied to any relational domain.

[7]URL: http://foursquare.com [15.11.2013]

8.4.1 Music

One particularly interesting domain is music. In this domain episodic schemata would change to schemata of different musical genres. These can be trained on a library of MIDI files. The trained DyBaNeM can be used to memorize a song and replay it after some time. Hopefully the replay will capture important aspects of the song and guess the rest from the inferred music genre. This ability would be an extension of systems like Sony's The Continuator (Pachet, 2003). Modeling various types of constraints needed to generate music is a well studied problem, see (Pachet et al., 2011; Boulanger-Lewandowski et al., 2012; Roy and Pachet, 2013)

8.4.2 Generic relational domains

Domains like episodes or music are inherently temporal. However, DyBaNeM can be applied to any domain that can be expressed in a BN (note that DBN is a only a sub-type of generic BN). The key idea of measuring discrepancy between recall given mems computed so far and the observation will work equally well. Thus the model can be used not only for episodic memories but also for storing semantic facts.

8.5 Implementing a Game Prototype

This work has shown how DyBaNeM can be applied to activity streams of IVAs. Creating a more complete virtual environment where the human user can interact with the IVAs and ask them about their memories for past events would be a natural extension of this initial work. Ideally the environment should resemble a small fraction of a MMORPG inhabited by multiple types of agents (i.e., woodcutter, soldier, priest, merchant, etc.) each with a slightly different set of goals and daily schedule. Thus each agent type would have its own episodic schemata influenced by its daily habits. Most of the simulated days will be alike. However, from time to time an unexpected event like a siege of the town will happen.

This environment would serve two purposes at once. First, it will be suitable for conducting thorough user studies where human subjects would judge believability of an IVA's recall. Second, it will be a standardized testbed where one would be able to study how different types of agents interpret actions of the other agents. As well as how they remember their own activity and activity of observed agents. The difficulty in interpreting activity of other agents is that observations will be usually sparse, thus there will be more possible episodes explaining the observations.

8.6 Integrating the Model in a Broader Agent Architecture

An interesting future work might be integration of DyBaNeM in a more generic cognitive architecture like Soar (Laird, 2012) or Sigma (Rosenbloom, 2013, 2014) which also uses the formalism of probabilistic graphical models. Knowledge stored in the episodic schemata, that is a generic model of the world, can be reused for an agent's DMS. When the agent has some hypothesis about his current state he might use episodic schemata to predict the effect of his own actions on the world state. This type of reasoning was shown in Section 5.5.

8.7 Summary

In this chapter we discussed how DyBaNeM can be extended in the future. This includes enhancements of the core functionality like integration of new probabilistic models or new encoding strategies. Another possible direction of work would be a series of empirical evaluations with human subjects. An interesting option would also be to integrate DyBaNeM to a complex agent architecture and then measure the impact of the EM module on an agent's overall performance.

Chapter 9

Conclusion

This thesis has developed a novel computational framework for EM modeling designed with the needs of IVAs in mind. The framework called DyBaNeM builds on top of probabilistic graphical models used for activity recognition and it is inspired by findings from psychology. Thorough review of related work has shown that none of the existing EM models matches the features of DyBaNeM.

After describing the model on a formal basis its functions were demonstrated on a simplistic toy domain. Two strategies for encoding memories were formulated and two probabilistic models were empirically evaluated in the more complex domain of the Monroe corpus. The framework was also applied to the activity stream generated by an IVA embodied in a 3D environment, demonstrating its applicability to virtual worlds. Both corpora contained sequences of atomic actions together with high level activities that were used as ground truth to measure accuracy of recall. The evaluations show that DyBaNeM fulfills all the requirements specified in Section 1.3.

DyBaNeM was designed as a generic framework that should be applicable to a wide range of applications including E-memory, computer games and virtual companions. In many cases development of complex software consists of integrating libraries implementing the needed sub-functionality. For instance, most current 3D computer games use some generic 3D engine, they integrate it with a physics engine and possibly with a separate engine implementing AI functionality like path finding and NPCs' DMS. However, there is no generic purpose EM library that could be integrated to multiple applications. DyBaNeM tries to bridge this gap. It provides a framework that should enable NPCs to answer questions about their own past activity and about activity of observed agents and players. Empirical evaluation of DyBaNeM in two domains shows promising results in this direction.

Bibliography

Abbeel, P., Koller, D., and Ng, A. Y. Learning Factor Graphs in Polynomial Time and Sample Complexity. *The Journal of Machine Learning Research*, 7: 1743–1788, 2006.

Akdemir, U., Turaga, P., and Chellappa, R. An ontology based approach for activity recognition from video. *Proceeding of the 16th ACM international conference on Multimedia - MM '08*, pages 709–712, 2008. doi: 10.1145/1459359.1459466. URL http://portal.acm.org/citation.cfm?doid=1459359.1459466.

Anderson, J. A Spreading Activation Theory of Memory. *Journal of Verbal Learning and Verbal Behavior*, 22:261–295, 1983. URL http://www.sciencedirect.com/science/article/pii/S0022537183902013.

Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., and Qin, Y. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

Anwar, A. and Franklin, S. S parse distributed memory for 'conscious' software agents. *Cognitive Systems Research*, 4:339–354, 2003. doi: 10.1016/S1389-0417(03)00015-9.

Atkinson, R. C. and Shiffrin, R. M. The control processes of short-term memory. Technical report, nstitute for Mathematical Studies in the Social Sciences, Stanford University, 1971.

Atkinson, R. C. and Shiffrin, R. M. Human memory: A proposed system and its control processes. *The psychology of learning and motivation*, 2:89–195, 1968.

Averell, L. and Heathcote, A. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25–35, February 2011. ISSN 00222496. doi: 10.1016/j.jmp.2010.08.009. URL http://linkinghub.elsevier.com/retrieve/pii/S0022249610001100.

Aylett, R., Louchart, S., Dias, J., and Paiva, A. FearNot ! - an experiment in emergent narrative. *Intelligent Virtual Agents*, 3661:305–316, 2005.

doi: 10.1007/11550617_26. URL http://www.springerlink.com/index/awvfvlemwnmbuh5e.pdf.

Baars, B. J. and Franklin, S. Consciousness Is Computational: the Lida Model of Global Workspace Theory. *International Journal of Machine Consciousness*, 1 (1):23–32, June 2009. ISSN 1793-8430. doi: 10.1142/S1793843009000050. URL http://www.worldscientific.com/doi/abs/10.1142/S1793843009000050.

Baddeley, A. The episodic buffer: a new component of working memory? *Trends in cognitive sciences*, 4(11):417–423, November 2000. ISSN 1879-307X. URL http://www.ncbi.nlm.nih.gov/pubmed/11058819.

Bartlett, F. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, Cambridge, England, 1932.

Bates, J. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, 1994.

Beal, M., Ghahramani, Z., and Rasmussen, C. The infinite hidden Markov model. *Advances in neural information processing systems*, pages 577–584, 2001. URL http://machinelearning.wustl.edu/mlpapers/paper_files/nips02-AA01.pdf.

Bell, C. and Gemmell, J. *Total recall: how the E-memory revolution will change everything*. Dutton, 2009. ISBN 0525951342.

Bell, T., Witten, I., and Cleary, J. Modeling for text compression. *ACM Computing Surveys (CSUR)*, 21(4), 1989. URL http://dl.acm.org/citation.cfm?id=76896.

Bellman, R. A Markovian decision process. *Indiana Univ. Math. J*, 6(4):679–684, 1957. URL http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD0606367.

Berzan, C. and Scheutz, M. What am I doing ? Automatic Construction of an Agent ' s State-Transition Diagram through Introspection. In *Proceedings of AAMAS 2012*, 2012.

Bickmore, T., Vardoulakis, L., Jack, B., and Paasche-Orlow, M. Automated Promotion of Technology Acceptance by Clinicians Using Relational Agents. In *Intelligent Virtual Agents*, pages 68—-78. Springer, 2013. URL http://link.springer.com/chapter/10.1007/978-3-642-40415-3_6.

Bilmes, J. Electrical Engineering On Virtual Evidence and Soft Evidence in Bayesian Networks. Technical Report 206, University of Washington, Dept. of Electrical Engineering, UWEETR-2004-0016, 2004. URL https://www.ee.washington.edu/techsite/papers/refer/UWEETR-2004-0016.html.

Bird, C. M. and Burgess, N. The hippocampus and memory: insights from spatial processing. *Nature reviews. Neuroscience*, 9(3):182–94, March 2008. ISSN 1471-0048. doi: 10.1038/nrn2335. URL http://www.ncbi.nlm.nih.gov/pubmed/18270514.

Blaylock, N. and Allen, J. Generating Artificial Corpora for Plan Recognition. In Ardissono, L., Brna, P., and Mitrovic, A., editors, *Proceedings of the 10th international conference on User Modeling (UM'05)*, pages 179–188, Edinburgh, 2005a. Springer. Corpus is downloadable from http://www.cs.rochester.edu/research/speech/monroe-plan/. doi: 10.1007/11527886_24.

Blaylock, N. and Allen, J. Recognizing instantiated goals using statistical methods. *IJCAI Workshop on Modeling Others from Observations (MOO-2005)*, pages 79–86, 2005b. URL https://www.cs.rochester.edu/research/cisd/pubs/2005/blaylock-allen-moo2005.pdf.

Blaylock, N. and Allen, J. Fast Hierarchical Goal Schema Recognition. *Proceedings of the National Conference on Artificial Intelligence (AAAI 2006)*, pages 796–801, 2006. URL http://www.aaai.org/Papers/AAAI/2006/AAAI06-126.pdf.

Bond, A. H. Representing episodic memory in a system-level model of the brain. *Neurocomputing*, 65-66:261–273, June 2005. ISSN 09252312. doi: 10.1016/j.neucom.2004.10.018. URL http://linkinghub.elsevier.com/retrieve/pii/S0925231204004035.

Bonferroni, C. E. l calcolo delle assicurazioni su gruppi di teste. *Studi in Onore del Professore Salvatore Ortu Carboni*, pages 13–60, 1935.

Boulanger-Lewandowski, N., Vincent, P., and Bengio, Y. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, (Cd):1159–1166, 2012.

Boyen, X., Friedman, N., and Koller, D. Discovering the hidden structure of complex dynamic systems. *Proceedings of the Fifteenth conference on Uncertainty in Artificial iItelligence*, pages 91–100, 1999. URL http://dl.acm.org/citation.cfm?id=2073807.

Bradski, G., Carpenter, G. a., and Grossberg, S. STORE working memory networks for storage and recall of arbitrary temporal sequences. *Biological Cybernetics*, 71(6):469–480, October 1994. ISSN 0340-1200. doi: 10.1007/BF00198465. URL http://www.springerlink.com/index/10.1007/BF00198465.

Brainerd, C. and Reyna, V. *The science of false memory.* Oxford University Press, 2005.

Brom, C. and Lukavský, J. Towards Virtual Characters with a Full Episodic Memory II : The Episodic Memory Strikes Back. *Proc. Empathic Agents, AAMAS workshop*, pages 1–9, 2009.

Brom, C., Pešková, K., and Lukavský, J. What Does Your Actor Remember? Towards Characters with a Full Episodic Memory. In *International Conference on Virtual Storytelling, LNCS 4871*, pages 89–101. Springer, 2007. URL http://www.springerlink.com/index/h833006228768183.pdf.

Brom, C., Burkert, O., and Kadlec, R. Timing in Episodic Memory for Virtual Characters. *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 305–312, 2010. doi: 10.1109/ITW.2010.5593339. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5593339.

Brom, C., Vyhnanek, J., Lukavsky, J., Waller, D., and Kadlec, R. A computational model of the allocentric and egocentric spatial memory by means of virtual agents, or how simple virtual agents can help to build complex computational models. *Cognitive Systems Research*, 17-18:1–24, 2012. doi: http://dx.doi.org/10.1016/j.cogsys.2011.09.001.

Brown, R. and Kulik, J. Flashbulb memories. *Cognition*, 5(1):73 – 99, 1977. doi: doi:10.1016/0010-0277(77)90018-X.

Bryson, J. *Intelligence by Design.* Ph.d., MIT, 2001.

Bryson, J. and Stein, L. Modularity and design in reactive intelligence. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1115—-1120, 2001. URL ftp://www.ai.mit.edu/people/joanna/ijcai01.pdf.

Bui, H. A general model for online probabilistic plan recognition. *International Joint Conference on Artificial Intelligence*, pages 1309–1318, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.1428&rep=rep1&type=pdf.

Bui, H., Venkatesh, S., and West, G. Policy recognition in the abstract hidden Markov model. *Journal of Artificial Intelligence Research*, 17:451—-499, 2002. URL http://arxiv.org/abs/1106.0672.

Bui, H., Phung, D., and Venkatesh, S. Hierarchical hidden Markov models with general state hierarchy. *Proceedings of the National Conference on Artificial Intelligence*, pages 324—-329, 2004. URL http://www.aaai.org/Papers/AAAI/2004/AAAI04-052.pdf.

Burgess, N. and Hitch, G. J. Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, 106(3):551–581, 1999. ISSN 0033-295X. doi: 10.1037//0033-295X.106.3.551. URL http://doi.apa.org/getdoi.cfm?doi=10.1037/0033-295X.106.3.551.

Burgess, N., Maguire, E. a., and O'Keefe, J. The human hippocampus and spatial and episodic memory. *Neuron*, 35(4):625–41, August 2002. ISSN 0896-6273. URL http://www.ncbi.nlm.nih.gov/pubmed/12194864.

Burkert, O. Connectionist model of episodic memory for virtual humans. Master's thesis, Charles University in Prague, 2009.

Burkert, O., Brom, C., Kadlec, R., and Lukavský, J. Timing in episodic memory: Virtual characters in action. *Proceedings of AISB workshop Remembering Who We Are-Human Memory For Artificial Agents*, pages 1–8, 2010. URL http://artemis.ms.mff.cuni.cz/main/papers/AISB10-Burkert.pdf.

Busemeyer, J., Pothos, E., Franco, R., and Trueblood, J. S. A quantum theoretical explanation for probability judgment errors. *Psychological Review*, 118 (2):193–218, 2011. doi: 10.1037/a0022542. URL http://psycnet.apa.org/journals/rev/118/2/193/.

Byrne, P., Becker, S., and Burgess, N. Remembering the past and imagining the future: a neural model of spatial memory and imagery. *Psychological review*, 114(2):340–75, April 2007. ISSN 0033-295X. doi: 10.1037/0033-295X.114.2.340. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2678675&tool=pmcentrez&rendertype=abstract.

Campos, C. and Ji, Q. Efficient Structure Learning of Bayesian Networks using Constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

Carpenter, G. and Grossberg, S. Adaptive resonance theory. In *The Handbook of Brain Theory and Neural Networks*, number 617, pages 87–90. MIT Press, 2003. URL http://digilib.bu.edu/ojs/index.php/trs/article/view/92.

Cataldi, M., Damiano, R., Lombardo, V., Pizzo, A., and Sergi, D. Integrating Commonsense Knowledge into the Semantic Annotation of Narrative Media Objects. In *AI* IA 2011: Artificial Intelligence Around Man and Beyond*, pages 312–323. Springer, 2011.

Cha, S. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300—-307, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.154.8446.

Charniak, E. and Goldman, R. A Bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993. URL http://www.sciencedirect.com/science/article/pii/000437029390060O.

Chater, N. and Oaksford, M., editors. *The Probabilistic Mind: Prospects for a Bayesian Cognitive Science*. Oxford University Press Oxford, 2008. ISBN 9780199216093. doi: 10.1093/acprof:oso/9780199216093.001.0001. URL http://dx.doi.org/10.1093/acprof:oso/9780199216093.001.0001.

Chen, L. and Nugent, C. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 5 (4):410–430, 2009. ISSN 1744-0084. doi: 10.1108/17440080911006199. URL http://www.emeraldinsight.com/10.1108/17440080911006199.

Cheng, J. and Druzdzel, M. J. AIS-BN : An Adaptive Importance Sampling Algorithm for Evidential Reasoning in Large Bayesian Networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000.

Cohen, J. Statistical power analysis for the behavioral sciences, 1988. ISSN 01621459. URL http://books.google.com/books?id=Tl0N2lRAO9oC&pgis=1.

Cohen, N. and Squire, L. Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210(4466):207–210, 1980. URL http://www.sciencemag.org/content/210/4466/207.short.

Colledanchise, M. and Ogren, P. How Behavior Trees modularize robustness and safety in hybrid systems. *Intelligent Robots and Systems (IROS 2014)*, pages 1482–1488, 2014. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6942752.

Conway, M. Sensory-perceptual episodic memory and its context: autobiographical memory. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 356(1413):1375–1384, September 2001. ISSN 0962-8436. doi: 10.1098/rstb.2001.0940. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1088521&tool=pmcentrez&rendertype=abstract.

Cooper, G. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2):393–405, 1990. URL http://www.sciencedirect.com/science/article/pii/000437029090060D.

Cowan, N. What are the differences between long-term, short-term, and working memory? *Progress in brain research*, 169:323–338, 2008. doi: 10.1016/S0079-6123(07)00020-9.What.

Cox, M. T., Munoz-avila, H., and Bergmann, R. Case-based planning. *The Knowledge Engineering Review*, 20(03):283–287, 2005. doi: 10.1017/S000000000000000.

Dagum, P. and Luby, M. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993. URL http://www.sciencedirect.com/science/article/pii/000437029390036B.

Dagum, P. and Luby, M. An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence*, 93(1-2):1–27, 1997.

Das, D. and Martins, A. F. T. A Survey on Automatic Text Summarization Single-Document Summarization. Technical report, Literature Survey for the Language and Statistics II course at Carnegie Mellon University, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.5100&amp;rep=rep1&amp;type=pdf.

Dawson, M. *Understanding cognitive science*. Wiley-Blackwell, 1998.

Dechter, R. *Constraint processing*. Morgan Kaufmann, 2003.

Dechter, R. and Mateescu, R. Mixtures of deterministic-probabilistic networks and their AND/OR search space. *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 120–129, 2004. URL http://dl.acm.org/citation.cfm?id=1036858.

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series*

B (Methodological), 39:1–38, 1977. URL http://www.jstor.org/stable/10.2307/2984875.

Derbinsky, N. and Laird, J. Efficiently implementing episodic memory. *Case-Based Reasoning Research and Development*, pages 403–417, 2009. URL http://www.springerlink.com/index/N20U703857R77110.pdf.

Derbinsky, N. and Laird, J. Computationally Efficient Forgetting via Base-Level Activation. *The 11th International Conference on Cognitive Modelling (ICCM)*, pages 109–110, 2012.

Derbinsky, N., Li, J., and Laird, J. A multi-domain evaluation of scaling in a general episodic memory. In *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 193–199, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/download/4852/5138.

Deutsch, T., Gruber, A., Lang, R., and Velik, R. Episodic memory for autonomous agents. *Human System Interactions (HSI 2008)*, pages 621 – 626, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4581512.

Dias, J., Ho, W. C., Vogt, T., Beeckman, N., Paiva, A., and Andr, E. I Know What I Did Last Summer: Autobiographic Memory in Synthetic Characters. *Affective Computing and Intelligent Interaction, LNCS 4738*, pages 606–617, 2007. doi: 10.1007/978-3-540-74889-2_53. URL http://www.springerlink.com/index/M05Q5024587T523H.pdf.

Dodd, W. and Gutierrez, R. The Role of Episodic Memory and Emotion in a Cognitive Robot. *ROMAN 2005, IEEE International Workshop on Robot and Human Interactive Communication*, pages 692–697, 2005. doi: 10.1109/ROMAN.2005.1513860. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1513860.

Doya, K., Ishii, S., Pouget, A., and Rao, R., editors. *Bayesian Brain: Probabilistic Approaches to Neural Coding*. 2007. ISBN 9780262042383. doi: 10.7551/mitpress/9780262042383.001.0001. URL http://books.google.com/books?id=bsQMWXXHzrYC&pgis=1.

Duong, T., Phung, D., Bui, H., and Venkatesh, S. Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 173(7-8): 830–856, May 2009. ISSN 00043702. doi: 10.1016/j.artint.2008.12.005. URL http://linkinghub.elsevier.com/retrieve/pii/S0004370208002142.

Epic Games. Unreal Tournament 2004, 2004. URL http://web.archive.org/web/20060615184746/http://www.unrealtournament2003.com/.

Erkan, G. and Radev, D. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479, 2004.

Erol, K., Hendler, J., and Nau, D. HTN planning: Complexity and expressivity. *AAAI*, pages 1123–1128, 1994. URL http://www.aaai.org/Papers/AAAI/1994/AAAI94-173.pdf.

Fader, A., Soderland, S., and Etzioni, O. Identifying relations for open information extraction. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535—-1545, 2011. URL http://dl.acm.org/citation.cfm?id=2145596.

Fagan, M. and Cunningham, P. Case-based plan recognition in computer games. In *Case-Based Reasoning Research and Development*, pages 161—-170. 2003. URL http://link.springer.com/chapter/10.1007/3-540-45006-8_15.

Faltersack, Z., Burns, B., and Nuxoll, A. Ziggurat: Steps Toward a General Episodic Memory. *2011 AAAI Fall Symposium*, 2011. URL http://www.aaai.org/ocs/index.php/FSS/FSS11/paper/viewFile/4138/4542.

Fine, S., Singer, Y., and Tishby, N. The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 62:41–62, 1998. URL http://www.springerlink.com/index/h7630r4u78j0xhu1.pdf.

Franklin, S. and Patterson Jr, F. The LIDA architecture: Adding new modes of learning to an intelligent, autonomous, software agent. *Integrated Design and Process Technology,IDPT-2006*, 2006.

Fung, R. and Chang, K.-c. Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks. *Uncertainty in Artificial Intelligence 5*, pages 112—-117, 1989.

Fung, R. and Favero, B. D. Backward Simulation in Bayesian Networks. pages 227–234, 1994.

Gallo, D. *Associative illusions of memory: False memory research in DRM and related tasks*. Psychology Press, 2006. ISBN 1841694142.

Geib, C. W. and Goldman, R. P. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, July 2009. ISSN 00043702. doi: 10.1016/j.artint.2009.01.003. URL http://linkinghub.elsevier.com/retrieve/pii/S0004370209000459.

Gemrot, J., Kadlec, R., Bída, M., Burkert, O., Píbil, R., Havlíček, J., Zemčák, L., Šimlovič, J., Vansa, R., Štolba, M., Plch, T., and Brom, C. Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. *Agents for Games and Simulations, LNCS 5920*, pages 1–15, 2009. URL http://link.springer.com/chapter/10.1007/978-3-642-11198-3_1.

Ghallab, M., Nau, D., and Traverso, P. *Automated Planning: theory and practice.* Morgan Kaufmann, 2004.

Gogate, V., Dechter, R., and Bidyuk, B. Modeling transportation routines using hybrid dynamic mixed networks. In *Conference on Uncertainty in Artificial Intelligence (UAI2005)*, 2005. URL http://arxiv.org/abs/1207.1384.

Gomes, P. F., Martinho, C., and Paiva, A. I've Been Here Before ! Location and Appraisal in Memory Retrieval. *AAMAS*, pages 1039–1046, 2011.

Guo, H. A Survey of Algorithms for Real-Time Bayesian Network Inference. In *In the joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, number 1, 2002.

Ha, E. Y., Rowe, J. P., Mott, B. W., and Lester, J. C. Goal Recognition with Markov Logic Networks for Player-Adaptive Games. *AIIDE*, 2011.

Harel, D. Statecharts: a visual complex systems. *Science of Computer Programming*, 8:231–274, 1987.

Hebb, D. *The Organization of Behavior.* New York: Wiley & Sons., 1949.

Hemmer, P. and Steyvers, M. Integrating Episodic and Semantic Information in Memory for Natural Scenes. *CogSci*, pages 1557–1562, 2009. URL http://csjarchive.cogsci.rpi.edu/proceedings/2009/papers/337/paper337.pdf.

Henrion, M. Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. In *Uncertainty in Artificial Intelligence 2*, pages 149–163. North Holland, 1988.

Ho, W. C., Dautenhahn, K., and Nehaniv, C. Computational Memory Architectures for Autobiographic Agents Interacting in a Complex Virtual Environment: A Working Model. *Connection Science*, 20(1):21–65, 2008. ISSN 0954-0091. doi: 10.1080/09540090801889469. URL http://www.informaworld.com/openurl?genre=article&doi=10.1080/09540090801889469&magic=crossref||D404A21C5BB053405B1A640AFFD44AE3.

Ho, W., Dias, J. a., and Figueiredo, R. Agents that remember can tell stories: integrating autobiographic memory into emotional agents. *AAMAS*, pages 35–37, 2007. URL http://dl.acm.org/citation.cfm?id=1329138.

Holland, O. and Marques, H. G. Functional Embodied Imagination and Episodic Memory. *International Journal of Machine Consciousness*, 2(2):245–259, 2010. doi: 10.1142/S1793843010000473. URL http://www.worldscinet.com/ijmc/02/preserved-docs/0202/S1793843010000473.pdf.

Hopfield, J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8): 2554, 1982.

Horvitz, E., Dumais, S., and Koch, P. Learning predictive models of memory landmarks. *Proceedings of the CogSci 2004*, 2004. URL http://candy.yonsei.ac.kr/courses/06mobile/10-1.pdf.

Houlette, R. and Fu, D. The Ultimate Guide to FSMs in Games. In *AI Game Programming Wisdom 2*. Charles River Media, 2003.

Howard, M. and Kahana, M. J. A Distributed Representation of Temporal Context. *Journal of Mathematical Psychology*, 46(3):269–299, June 2002. ISSN 00222496. doi: 10.1006/jmps.2001.1388. URL http://linkinghub.elsevier.com/retrieve/pii/S0022249601913884.

Huang, C. and Darwiche, A. Inference in belief networks: A procedural guide. *International journal of approximate reasoning*, (15):225–263, 1996. URL http://www.sciencedirect.com/science/article/pii/S0888613X96000692.

Huynh, T., Fritz, M., and Schiele, B. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*, pages 10–19, New York, New York, USA, 2008. ACM Press. ISBN 9781605581361. doi: 10.1145/1409635.1409638. URL http://portal.acm.org/citation.cfm?doid=1409635.1409638.

Intille, S. S., Larson, K., Tapia, E. M., Beaudin, J. S., Kaushik, P., Nawyn, J., and Rockinson, R. Using a live-in laboratory for ubiquitous computing research. In *Pervasive Computing*, pages 349–365. Springer, 2006. URL http://link.springer.com/chapter/10.1007/11748625_22.

Ito, Y., Ueno, T., Kitagami, S., and Kawaguchi, J. A computational exploration on the role of semantic memory in episodic future thinking. *CogSci*, pages 2626–2631, 2013. URL http://csjarchive.cogsci.rpi.edu/Proceedings/2013/papers/0474/paper0474.pdf.

Itti, L. and Baldi, P. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–1306, 2009. ISSN 1878-5646. doi: 10.1016/j.visres.2008.09.007. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2782645&tool=pmcentrez&rendertype=abstract.

Kabanza, F., Filion, J., Benaskeur, A., and Irandoust, H. Controlling the Hypothesis Space in Probabilistic Plan Recognition. *IJCAI*, 2013. URL http://planiart.usherbrooke.ca/~julien/files/kabanza-etal-ijcai13.pdf.

Kadlec, R. and Brom, C. Towards an Automatic Aiary : an Activity Recognition From Data Collected by a Mobile Phone. In *IJCAI Workshop on Space, Time and Ambient Intelligence*, pages 56–61, 2011.

Kadlec, R. and Brom, C. DyBaNeM : Bayesian Episodic Memory Framework for Intelligent Virtual Agents. *Intelligent Virtual Agents 2013, LNCS 8108*, pages 15–28, 2013a.

Kadlec, R. and Brom, C. DyBaNeM : Bayesian Framework for Episodic Memory Modelling. *The 12th International Conference on Cognitive Modelling (ICCM)*, 2013b.

Kadlec, R., Čermák, M., Behan, Z., and Brom, C. Generating Corpora of Activities of Daily Living and Towards Measuring the Corpora's Complexity. *Cognitive Agents for Virtual Environments - First International Workshop, LNCS 7764*, pages 149–166, 2013. URL http://artemis.ms.mff.cuni.cz/main/papers/KadlecCAVE2012.pdf.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, May 1998. ISSN 00043702. doi: 10.1016/S0004-3702(98)00023-X. URL http://linkinghub.elsevier.com/retrieve/pii/S000437029800023X.

Kahana, M. J. Associative retrieval processes in free recall. *Memory & cognition*, 24(1):103–9, January 1996. ISSN 0090-502X. URL http://www.ncbi.nlm.nih.gov/pubmed/8822162.

Kahneman, D. and Tversky, A. Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, 47(March):263–291, 1979. URL http://www.jstor.org/stable/10.2307/1914185.

Kahneman, D., Slovic, P., and Tversky, A. *Judgment under uncertainty: Heuristics and biases.* Cambridge University Press, 1982.

Kamar, E. and Horvitz, E. Jogger : Models for Context-Sensitive Reminding. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pages 1089–1090, 2011.

Kanerva, P. *Sparse distributed memory.* The MIT Press, 1988.

Kasap, Z., Moussa, M. B., Chaudhuri, P., and Magnenat-Thalmann, N. Making Them Remember—Emotional Virtual Characters with Memory. *IEEE Computer Graphics and Applications*, 29(2):20–29, March 2009. ISSN 0272-1716. doi: 10.1109/MCG.2009.26. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4797513.

Kautz, H. and Allen, J. Generalized plan recognition. *Proceedings of the fifth national conference on artificial intelligence (AAAI-86)*, pages 32–37, 1986. URL http://www.aaai.org/Papers/AAAI/1986/AAAI86-006.pdf.

Kautz, H., Etzioni, O., and Fox, D. Foundations of assisted cognition systems. Technical report, University of Washington, 2003. URL https://ftp.cs.rochester.edu/~kautz/papers/ac03tr.pdf.

Kerr, W., Tran, A., and Cohen, P. Activity Recognition with Finite State Machines. In *International Joint Conference on Artificial Intelligence*, pages 1348–1353, 2011.

Kiukkonen, N., Blom, J., Dousse, O., Gatica-perez, D., and Laurila, J. Towards rich mobile phone datasets : Lausanne data collection campaign. In *Proceedings of the ACM International Conference on Pervasive Services (ICPS)*, 2010.

Knowlton, B. J. and Squire, L. R. Artificial grammar learning depends on implicit acquisition of both abstract and exemplar-specific information. *Journal of experimental psychology. Learning, memory, and cognition*, 22(1):169–81, January 1996. ISSN 0278-7393. URL http://www.ncbi.nlm.nih.gov/pubmed/8648284.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques.* The MIT Press, 2009.

Kope, A., Rose, C., and Katchabaw, M. Modeling Autobiographical Memory for Believable Agents. *AIIDE*, pages 23–29, 2013. URL http://publish.uwo.ca/~akope2/conferences/AKope_AIIDE.pdf.

Kopp, S., Gesellensetter, L., Kramer, N., and Wachsmuth, I. A Conversational Agent as Museum Guide–Design and Evaluation of a Real-World Application. *Intelligent Virtual Agents, LNCS 3661*, pages 1–14, 2005. URL http://link.springer.com/chapter/10.1007/11550617_28.

Kullback, S. Statistics and information theory, 1959.

Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)*, volume 2001, pages 282–289, 2001. URL http://repository.upenn.edu/cis_papers/159/.

Laird, J. E. *The Soar Cognitive Architecture*. MIT Press, 2012.

Langley, P., Laird, J. E., and Rogers, S. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, June 2009. ISSN 13890417. doi: 10.1016/j.cogsys.2006.07.004. URL http://linkinghub.elsevier.com/retrieve/pii/S1389041708000557.

Larsen, S. F., Thompson, C. P., and Hansen, T. Time in autobiographical memory. *Remembering our past: Studies in autobiographical memory*, pages 129–156, 1996.

Lazer, D., Pentland, A. S., Adamic, L., Aral, S., Barabasi, A. L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., and Others. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721, 2009. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2745217/.

Lehman, M. and Malmberg, K. J. A global theory of remembering and forgetting from multiple lists. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 35(4):970–988, 2009. doi: 10.1037/a0015728. URL http://psycnet.apa.org/journals/xlm/35/4/970/.

Lesh, N. and Etzioni, O. A Sound and Fast Goal Recognizer. In *International Joint Conference on Artificial Intelligence*, pages 1704–1710, 1995.

Li, J. and Laird, J. Preliminary Evaluation of Long-Term Memories for Fulfilling Delayed Intentions. In *2011 AAAI Fall Symposium Series*, 2011. URL http://www.aaai.org/ocs/index.php/FSS/FSS11/paper/viewPDFInterstitial/4146/4546.

Li, W. P., Balint, T., and Allbeck, J. M. Using a Parameterized Memory Model to Modulate NPC AI. *Intelligent Virtual Agents 2013, LNCS 8108*, pages 1–14, 2013.

Liao, L., Fox, D., and Kautz, H. Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields. *The International Journal of Robotics Research*, 26(1):119–134, January 2007a. ISSN 0278-3649. doi: 10.1177/0278364907073775. URL http://ijr.sagepub.com/cgi/doi/10.1177/0278364907073775.

Liao, L., Patterson, D., Fox, D., and Kautz, H. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, April 2007b. ISSN 00043702. doi: 10.1016/j.artint.2007.01.006. URL http://linkinghub.elsevier.com/retrieve/pii/S0004370207000380.

Lim, M. Y. Memory Models for Intelligent Social Companions. In *Human-Computer Interaction: The Agency Perspective, Studies in Computational Intelligence Volume 396*, pages 241–262. Springer, 2012.

Lim, M. Y., Aylett, R., Vargas, P. A., Ho, W. C., and Dias, J. a. Human-like Memory Retrieval Mechanisms for Social Companions. *AAMAS*, pages 1117–1118, 2011.

Lim, M., Aylett, R., Ho, W., Enz, S., and Vargas, P. A socially-aware memory for companion agents. *Intelligent Virtual Agents*, 2009. URL http://link.springer.com/chapter/10.1007/978-3-642-04380-2_5.

Lisman, J. E. Relating hippocampal circuitry to function: recall of memory sequences by reciprocal dentate-CA3 interactions. *Neuron*, 22(2):233–42, February 1999. ISSN 0896-6273. URL http://www.ncbi.nlm.nih.gov/pubmed/10069330.

Lisý, V., Pıbil, R., Stiborek, J., Bošanský, B., and Pechoucek, M. Game-theoretic Approach to Adversarial Plan Recognition. In *ECAI 2012 - 20th European Conference on Artificial Intelligence*, pages 546–551, 2012. ISBN 9781614990987. doi: 10.3233/978-1-61499-098-7-546. URL http://agents.felk.cvut.cz/cgi-bin/docarc/public.pl/document/396/FAIA242-0546.pdf.

Liu, H. and Maes, P. What would they think?: a computational model of attitudes. *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 38–45, 2004. URL http://dl.acm.org/citation.cfm?id=964451.

Loftus, E. and Pickrell, J. The Formation of False Memories. *Psychiatric Annals*, 25(12):720—-725, 1995.

Lu, H., Yang, J., Liu, Z., Lane, N., Choudhury, T., and Campbell, A. The Jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the*

8th ACM Conference on Embedded Networked Sensor Systems, pages 71–84. ACM, 2010.

Macedo, L. and Cardoso, A. Modeling forms of surprise in an artificial agent. In *CogSci*, 2001. URL http://conferences.inf.ed.ac.uk/cogsci2001/pdf-files/0588.pdf.

Macedo, L., Reisenzein, R., and Cardoso, A. Modeling Forms of Surprise in Artificial Agents : Empirical and Theoretical Study of Surprise Functions. In *CogSci*, 2004.

Mahoney, M. Adaptive weighing of context models for lossless data compression. *Florida Inst. Technol., Melbourne, FL, Tech. Rep. CS-2005-16*, 2005.

Marr, D. *Vision: A computational investigation into the human representation and processing of visual information.* WH Freeman, 1982.

Mattar, N. and Wachsmuth, I. Who Are You? On the Acquisition of Information about People for an Agent that Remembers. *ICAART 2012*, pages 98 – 105, 2012. URL http://pub.uni-bielefeld.de/publication/2440520.

Miller, G. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81–97, 1956. URL http://psycnet.apa.org/journals/rev/63/2/81/.

Moffat, A., Neal, R. M., and Witten, I. H. Arithmetic coding revisited. *ACM Transactions on Information Systems*, 16(3):256–294, July 1998. ISSN 10468188. doi: 10.1145/290159.290162. URL http://portal.acm.org/citation.cfm?doid=290159.290162.

Murdock, J., Bennet B. The serial position effect of free recall. *Journal of Experimental Psychology*, 64(5):482–488, 1962. ISSN 0022-1015. doi: 10.1037/h0045106. URL http://content.apa.org/journals/xge/64/5/482.

Murphy, K. P. *Dynamic Bayesian Networks: Representation, Inference and Learning.* Ph.d., University of California, Berkeley, 2002.

Murphy, K. and Paskin, M. Linear-time inference in hierarchical HMMs. *Advances in Neural Information Processing Systems*, 2:833—-840, 2002.

Ng, A. and Jordan, M. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, pages 841–848, 2001.

Nilsson, N. J. and Fikes, R. E. STRIPS : A New Approach to the Application of Theorem Proving to. *Artificial Intelligence*, 8:189–208, 1971.

Nodelman, U. *Continuous Time Bayesian Networks*. Ph.d., Stanford, 2007.

Nodelman, U., Shelton, C., and Koller, D. Continuous time Bayesian networks. *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 378—-387, 2002. URL http://dl.acm.org/citation.cfm?id=2073921.

Nooraei, B., Rich, C., and Sidner, C. A Real-Time Architecture for Embodied Conversational Agents: Beyond Turn-Taking. In *Conf. on Advances in Computer-Human Interactions*, volume 1, 2014. URL http://web.cs.wpi.edu/~rich/always/pub/NooraeiRichSidner2014_ACHI.pdf.

Norman, K., Detre, G., and Polyn, S. Computational models of episodic memory. *The Cambridge handbook of computational psychology*, pages 189—-224, 2008. URL http://compmem.princeton.edu/publications/Norman_CambridgeHandbook.pdf.

Nuxoll, A., Laird, J., and James, M. Comprehensive working memory activation in Soar. *International Conference on Cognitive Modeling*, pages 226–230, 2004. URL http://pdf.aminer.org/000/686/815/comprehensive_working_memory_activation_in_soar.pdf.

Nuxoll, A., Tecuci, D., Ho, W. C., and Wang, N. Comparing forgetting algorithms for artificial episodic memory systems. *Proc. of the Symposium on Human Memory for Artificial Agents. AISB*, pages 14—-20, 2010. URL http://dl.lirec.eu/papers/Nuxoll_Tecuci_Ho_Wang_AISB2010.pdf.

Nuxoll, A. M. *Enhancing Intelligent Agents with Episodic Memory*. Phd, University of Michigan, 2007.

Nuxoll, A. M. and Laird, J. E. Enhancing intelligent agents with episodic memory. *Cognitive Systems Research*, October 2011. ISSN 13890417. doi: 10.1016/j.cogsys.2011.10.002. URL http://linkinghub.elsevier.com/retrieve/pii/S1389041711000428.

Nuxoll, A. M. and Laird, J. E. Enhancing intelligent agents with episodic memory. *Cognitive Systems Research*, 17-18:34–48, 2012. ISSN 13890417. doi: 10.1016/j.cogsys.2011.10.002. URL http://linkinghub.elsevier.com/retrieve/pii/S1389041711000428.

O'Hara, K., Morris, R., Shadbolt, N., Hitch, G. J., Hall, W., and Beagrie, N. Memories for life: a review of the science and technology. *Journal of the Royal Society, Interface / the Royal Society*, 3(8):351–65, June 2006. ISSN 1742-5662. doi: 10.1098/rsif.2006.0125. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1578756&tool=pmcentrez&rendertype=abstract.

Oliver, N., Garg, A., and Horvitz, E. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96(2):163–180, November 2004. ISSN 10773142. doi: 10.1016/j.cviu.2004.02.004. URL http://linkinghub.elsevier.com/retrieve/pii/S1077314204000724.

Oliver, N. and Horvitz, E. A Comparison of HMMs and Dynamic Bayesian Networks for Recognizing Office Activities. In *User Modeling 2005*, pages 199—-209. Springer, 2005.

Orbanz, P. and Teh, Y. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*, pages 81–89. Springer, 2010. URL http://link.springer.com/content/pdf/10.1007/978-0-387-30164-8_66.pdf.

O'Reilly, R. and Rudy, J. Conjunctive representations in learning and memory: Principles of cortical and hippocampal function. *Psychological review*, 108(2):311, 2001.

Ortony, A., Clore, G., and Collins, A. *The cognitive structure of emotions*. Cambridge University Press, 1990.

Pachet, F. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003. URL http://www.tandfonline.com/doi/abs/10.1076/jnmr.32.3.333.16861.

Pachet, F., Roy, P., and Barbieri, G. Finite-Length Markov Processes with Constraints. *IJCAI*, pages 635–642, 2011.

Pan, S. J. and Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5288526.

Pearl, J. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(1986):241–288, 1986. URL http://www.sciencedirect.com/science/article/pii/000437028690072X.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988. URL http://books.google.com/books?hl=en&lr=&id=AvNID7LyMusC&oi=fnd&pg=PA1&dq=Probabilistic+reasoning+in+intelligent+systems:+networks+of+plausible+inference&ots=FZWSYjtt36&sig=HmMLeNhXc03uc9TX9DjZAaeL998.

Pearson, K. The problem of the random walk. *Nature*, 72(1865):294, 1905.

Pelechano, N., Allbeck, J. M., and Badler, N. I. Virtual crowds: Methods, simulation, and control. *Synthesis Lectures on Computer Graphics and Animation*, 3(1):1–176, 2008.

Pollack, J. Recursive distributed representations. *Artificial Intelligence*, 46(1-2): 77–105, 1990.

Polyn, S. and Kahana, M. Memory search and the neural representation of context. *Trends in cognitive sciences*, 12(1):24–30, 2008.

Pynadath, D. and Wellman, M. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence (UAI 2000)*, pages 507—-514. Morgan Kaufmann Publishers Inc., 2000. URL http://dl.acm.org/citation.cfm?id=2074005.

Rabe, F. and Wachsmuth, I. Cognitively motivated episodic memory for a virtual guide. In Filipe, J. and Fred, A. L. N., editors, *ICAART (1)*, pages 524–527. SciTePress, 2012. ISBN 978-989-8425-95-9.

Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. ISSN 00189219. doi: 10.1109/5.18626. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=18626.

Ramamaurthy, U., D'Mello, S., and Franklin, S. Modified sparse distributed memory as transient episodic memory for cognitive software agents. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 6, pages 5858–5863. IEEE, 2004.

Ramamurthy, U., D'mello, S., and Franklin, S. Realizing forgetting in a modified sparse distributed memory system. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, pages 1992–1997, 2006.

Ramirez, M. and Geffner, H. Plan recognition as planning. In *Proc. IJCAI*, pages 1778–1783, 2009.

Ramirez, M. and Geffner, H. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. *Proc. IJCAI*, pages 2009–2014, 2011.

Reilly, R. C. O. and Rudy, J. W. Conjunctive Representations in Learning and Memory : Principles of Cortical and Hippocampal Function. *Psychological Review*, 108(2):311–345, 2001. doi: 10.1037//0033-295X.

Richardson, M. and Domingos, P. Markov logic networks. *Machine Learning*, 62(1-2)):107–136, 2006. doi: 10.1007/s10994-006-5833-1. URL `http://link.springer.com/article/10.1007/s10994-006-5833-1`.

Rickel, J. and Johnson, W. L. Animated Agents for Procedural Training in Virtual Reality : Perception , Cognition , and Motor Control. *Applied Artificial Intelligence*, 4-5(13):343–382, 1999.

Rosenbloom, P. S. The Sigma Cognitive Architecture and System. *AISB Quarterly*, 136:4–13, 2013.

Rosenbloom, P. Deconstructing Episodic Memory and Learning in Sigma. *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, pages 1317–1322, 2014. URL `http://pollux.usc.edu/~rosenblo/Pubs/ELS-CogSci14finalD.pdf`.

Roy, P. and Pachet, F. Enforcing Meter in Finite-Length Markov Sequences. *AAAI*, 2013. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6422`.

Ryoo, M. S. and Matthies, L. First-person activity recognition: What are they doing to me? *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2730–2737, 2013. ISSN 10636919. doi: 10.1109/CVPR.2013.352.

Sadilek, A. and Kautz, H. Recognizing Multi-Agent Activities from GPS Data. *AAAI*, pages 1134–1139, 2010.

Sapouna, M., Wolke, D., Vannini, N., Woods, S., Schneider, W., Enz, S., Hall, L., Piava, A., Andre, E., Dautenhahn, K., and Aylett, R. Virtual learning intervention to reduce bullying victimization in primary school: a controlled trial. *Journal of Child Psychology and Psychiatry*, 51(1):104—-112, 2009. doi: 10.1111/j.1469-7610.2009.02137.x.

Schacter, D. L. Implicit memory: History and current status. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13(3):501–518, 1987. ISSN 0278-7393. doi: 10.1037/0278-7393.13.3.501.

Schacter, D. *The seven sins of memory.* Houghton Mifflin New York, 2001.

Schank, R. C. *Dynamic memory revisited.* Cambridge University Press, 1999.

Schmidt, C., Sridharan, N., and Goodson, J. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11 (1-2):45–83, 1978. URL http://www.sciencedirect.com/science/article/pii/0004370278900127.

Schultheis, H., Lile, S., and Barkowsky, T. Extending act-r's memory capabilities. *Proc. of EuroCogSci*, 7:758–763, 2007.

Seldin, Y., Bejerano, G., and Tishby, N. Unsupervised sequence segmentation by a mixture of switching variable memory Markov sources. *ICML*, pages 513–520, 2001. URL http://www.soe.ucsc.edu/~jill/papers/icml01.pdf.

Shastri, L. Episodic memory and cortico-hippocampal interactions. *Trends in Cognitive Sciences*, 6(4):162–168, 2002.

Simonyan, K. and Zisserman, A. Two-Stream Convolutional Networks for Action Recognition in Videos. *Advances in Neural Information Processing Systems 27*, pages 568—-576, 2014. URL http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf.

Sindlar, M. In the eye of the beholder: explaining behavior through mental state attribution. *SIKS dissertation series*, 2011-42, 2011.

Singh, P. LifeNet: a propositional model of ordinary human activity. *Proceedings of the Workshop on Distributed*, 2003. URL http://www.media.mit.edu/~push/LifeNet.pdf.

Snaider, J. and Franklin, S. Extended Sparse Distributed Memory and Sequence Storage. *Cognitive Computation*, 4(2):172–180, February 2012. ISSN 1866-9956. doi: 10.1007/s12559-012-9125-8. URL http://www.springerlink.com/index/10.1007/s12559-012-9125-8.

Song, Y. C., Kautz, H., Allen, J., Swift, M., Li, Y., and Luo, J. A Markov Logic Framework for Recognizing Complex Events from Multimodal Data Categories and Subject Descriptors. *15th ACM International Conference on Multimodal Interaction (ICMI 2013)*, 2013.

Stikic, M. and Schiele, B. Activity recognition from sparsely labeled data using multi-instance learning. *Location and Context Awareness*, pages 156–173, 2009.

Storck, J., Hochreiter, S., and Schmidhuber, J. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 159–164, 1995.

Subagdja, B., Wang, W., Tan, A.-H., Tan, Y.-S., and Teow, L.-N. Memory Formation, Consolidation, and Forgetting in Learning Agents. *AAMAS 2012*, pages 1007–1014, 2012. URL http://dl.acm.org/citation.cfm?id=2343841.

Sutton, C., McCallum, A., and Rohanimanesh, K. Dynamic Conditional Random Fields : Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. *Journal of Machine Learning Research*, 8:693–723, 2007.

Takac, M. and Knott, A. A neural network model of working memory for episodes. *CogSci*, pages 1432–1437, 2013.

Tan, A.-h., Carpenter, G., and Grossberg, S. Intelligence through interaction: Towards a unified theory for learning. *Advances in Neural Networks–ISNN*, pages 1094–1103, 2007. URL http://www.springerlink.com/index/A73110U544T20743.pdf.

Tecuci, D. and Porter, B. Memory based goal schema recognition. In *Proceedings of the 22nd Florida Artificial Intelligence Research Society Conference (FLAIRS-22)*, 2009. URL http://www.aaai.org/ocs/index.php/FLAIRS/2009/paper/download/114/254.

Tecuci, D. and Porter, B. A generic memory module for events. In *Proceedings to the 20th Florida Artificial Intelligence Research Society Conference (FLAIRS 200)*, volume 68, 2007. URL http://www.aaai.org/Papers/FLAIRS/2007/Flairs07-028.pdf.

Toda, M. *Man, robot, and society: Models and speculations*. Kluwer Academic Pub, 1982.

Tulving, E. Episodic and semantic memory. In *Organization of Memory*, pages 381–403. New York: Academic Press, 1972. URL http://web.media.mit.edu/~jorkin/generals/papers/Tulving_memory.pdf.

Tulving, E. *Elements Of Episodic Memory*. Clarendon Press Oxford, 1983.

Turaga, P., Chellappa, R., Subrahmanian, V., and Udrea, O. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4633644.

Turing, A. M. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. URL http://cogprints.org/499/1/turing.html.

Vail, D. L., Veloso, M. M., and Lafferty, J. D. Conditional Random Fields for Activity Recognition Categories and Subject Descriptors. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent systems (AAMAS 2007)*, 2007. doi: 10.1145/1329125.1329409.

van Dongen, E. V., Thielen, J.-W., Takashima, A., Barth, M., and Fernández, G. Sleep supports selective retention of associative memories based on relevance for future utilization. *PLoS ONE*, 7(8): e43426, January 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0043426. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3420871&tool=pmcentrez&rendertype=abstract.

Čermák, M. *Episodic Memory for Virtual Agents: False memories.* M.sc., Charles University in Prague, 2013.

Čermák, M., Kadlec, R., and Brom, C. Towards modeling false memory using virtual characters: a position paper. In *Symposium on Human Memory for Artificial Agents, AISB*, pages 20–24, 2011.

Wagenaar, W. My memory: A study of autobiographical memory over six years. *Cognitive psychology*, 18(2):225–252, 1986. ISSN 0010-0285.

Wikipedia. List of memory biases — Wikipedia, The Free Encyclopedia, 2015a. URL http://en.wikipedia.org/wiki/List_of_memory_biases.

Wikipedia. Forgetting Curve, 2015b. URL https://en.wikipedia.org/wiki/File:ForgettingCurve.svg.

Wood, R., Baxter, P., and Belpaeme, T. A review of long-term memory in natural and synthetic systems. *Adaptive Behavior*, 0(0):1–23, December 2011. ISSN 1059-7123. doi: 10.1177/1059712311421219. URL http://adb.sagepub.com/cgi/doi/10.1177/1059712311421219.

Wooldridge, M. *Reasoning about rational agents.* MIT press, 2000.

Wu, C. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983. URL http://projecteuclid.org/euclid.aos/1176346060.

Wyatt, D., Philipose, M., and Choudhury, T. Unsupervised Activity Recognition Using Automatically Mined Common Sense. In *AAAI-05*, pages 21–27, 2005.

Yin, J., Chai, X., and Yang, Q. High-level goal recognition in a wireless LAN. In *Proceedings of the national conference on artificial intelligence*, pages 578–584. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

Yuan, C. and Druzdzel, M. J. An Importance Sampling Algorithm Based on Evidence Pre-propagation. *Uncertainty in Artificial Intelligence 19*, pages 624–631, 2003.

Zacks, J. M. and Tversky, B. Event structure in perception and conception. *Psychological bulletin*, 127(1):3–21, 2001. doi: 10.1037/0033-2909.127.1.3. URL http://psycnet.apa.org/journals/bul/127/1/3/.

Zacks, J., Speer, N., Swallow, K., Braver, T., and Reynolds, J. Event perception: a mind-brain perspective. *Psychological Bulletin*, 133(2):273–293, 2007. doi: 10.1037/0033-2909.133.2.273.Event. URL http://psycnet.apa.org/psycinfo/2007-02367-005.

Ziv, J. and Lempel, A. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530—-536, 1978. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1055934.

Appendices

Appendix A

Running the experiments

A.1 Installation

The source code of the DyBaNeM framework together with the code of all experiments can be downloaded from http://dybanem.googlecode.com/svn/trunk/. Issue the following command to download the repository.

```
> svn checkout http://dybanem.googlecode.com/svn/trunk/
```

DyBaNeM requires:

1. Java 1.8 or newer, it is recommended to use 32bit version of Java.
2. The experiments were run on Windows 7 system, however, it should be possible to run the experiments on Linux since most of the program is written in Java and the mandatory dependencies are available in both Windows and Linux versions.
3. Maven[1] as a build tool.
4. jSMILE library[2], the experiments were run with a version from April 10, 2014. After downloading jsmile_win32.zip unpack it and place jsmile.dll in a location that is included in the system path variable (%PATH% on Windows, $PATH on Linux). The jsmile.dll library will be loaded by the Java program during runtime. In the end move smile.jar file to ${svnCheckoutDir}\projects\jsmileMvnInstall and execute *installJsmileJar.bat*

[1]Available from URL: http://maven.apache.org/

[2]Proprietal licence of SMILE prohibits redistribution of the library with third party software. The binaries have to be downloaded from the project's homepage. URL: https://dslpitt.org/genie/index.php/downloads where you can find them under the section "SMILE Wrappers for Java and .NET" as Java Win32 binaries. At the time of writing the file is available under this link, URL: https://dslpitt.org/genie/download/jsmile_win32.zip.

script that resides in the same folder. This will install jSMILE as Maven artifact into the local Maven repository.

5. (OPTIONAL) To edit and compile source code from an IDE you will need to add install project Lombok to your IDE. All the necessary steps are described at `http://jnb.ociweb.com/jnb/jnbJan2010.html#installation`.

Now you can build DyBaNeM and all required sub-projects by the following commands:

```
> cd ${svnCheckoutDir}\projects
> mvn install
```

A.2 Toy Domain Examples

After building the project you can run the experiment by executing:

```
> cd ${svnCheckoutDir}\projects\dybanem
> target\appasembler\bin\ToyDomain.bat
```

Alternatively you can execute class:
`cz.cuni.amis.episodic.bayes.experiment.Experiment_2a.java` from Java IDE of your choice. All the results and graphs will be written to a folder `target\experiments\2` relative to the place of execution.

A.3 Monroe Corpus Examples

Script Experiment_4c_monroe_effectOfTrainingDataLarge.java generates all the results presented in Section 6.1. It outputs all the graphs shown in Figures 6.1 — 6.8 together with other graphs that were not included in the thesis. The body of Tables 6.1, 6.3, 6.4 and 6.5 will be printed to the standard output.

```
> cd ${svnCheckoutDir}\projects\dybanem
> target\appasembler\bin\MonroeDomain.bat
```

After executing this script the generated graphs will be stored in a folder `target\experiments\4_monroe_effectOfTrainingDataLarge_2` relative to the place of execution.

A.4 Pogamut Corpus Examples

Experiments with the Pogamut corpus presented in Section 6.2 can be replicated by executing a script Experiment_9_cermak.java.

You can use the following command to run this script:

```
> cd ${svnCheckoutDir}\projects\dybanem
> target\appasembler\bin\PogamutDomain.bat
```

After executing this script the graphs with example recall will be stored in a folder `target\experiments\9_cermak` relative to the place of execution.

Appendix B

Monroe plan corpus

The highest level episodes in the Monroe corpus are:

```
CLEAR_ROAD_HAZARD
CLEAR_ROAD_TREE
CLEAR_ROAD_WRECK
FIX_POWER_LINE
FIX_WATER_MAIN
PLOW_ROAD
PROVIDE_MEDICAL_ATTENTION
PROVIDE_TEMP_HEAT
QUELL_RIOT
SET_UP_SHELTER
```

The atomic action in the Monroe corpus are:

```
CALL
CARRY_BLOCKAGE_OUT_OF_WAY
CLEAN_HAZARD
CLIMB_IN
CLIMB_OUT
CUT_TREE
DIG
DISENGAGE_PLOW
ENGAGE_PLOW
FILL_IN
HOOK_TO_TOW_TRUCK
HOOK_UP
LOAD
NAVEGATE_SNOWPLOW
```

```
NAVEGATE_VEHICLE
PAY
PICKUP_CONES
PLACE_CONES
PUMP_GAS_INTO
REMOVE_WIRE
REPLACE_PIPE
SET_UP_BARRICADES
STRING_WIRE
TREAT_IN_HOSPITAL
TURN_ON
TURN_ON_HEAT
UNHOOK_FROM_TOW_TRUCK
UNLOAD
```

Acronyms

2TBN two-slice temporal Bayes network. 35, 36

AHM*E*M Abstract Hidden Markov Memory Model. 34, 38, 41, 53–73, 77, 78, 80–82, 84, 90, 91, 94, 95, 123, 124

AHMM Abstract Hidden Markov Model. 78

AI artificial intelligence. 2

ART adaptive resonance theory. 15, 26

BN Bayesian Network. 47, 78, 97

CHMM Cascading Hidden Markov Model. 34, 38, 47, 53–60, 64–69, 72, 73, 77, 78, 80, 81, 84, 88, 91, 94, 123, 124

CPMF conditional probability mass function. 37, 38, 55

CRF Condition Random Field. 78, 79

CSP constraint satisfaction programming. 78

CTBN Continuous Time Bayesian Network. 79

CxHSMM Coxian hidden Semi Markov Model. 78, 80

DBN Dynamic Bayesian Network. 1, 3, 28–31, 34–36, 38–40, 43–47, 49, 50, 60, 77–80, 83, 88, 91, 94, 97, 123

DFP Directed Forgetting Paradigm. 10

DMS Decision Making System. 15–18, 22, 39, 85, 91

EM Episodic Memory. 1–12, 15–18, 20–23, 26, 28–31, 33, 51, 74, 76, 85, 88, 96, 123

EM algorithm Expectation-Maximization algorithm. 36–38, 46, 47, 54, 95

EST Event Segmentation Theory. 1, 3

FPS first person shooter. 15

FSM Finite State Machine. 38

FTT Fuzzy-Trace Theory. 1, 3, 28, 30

HFSM hierarchical finite state machine. 28

HHMM Hierarchical Hidden Markov Model. 38, 78, 80

HMM Hidden Markov Model. 38, 78, 95

HSMM Hidden Semi-Markov Model. 78, 80

HTN hierarchical task network. 83

ISC intelligent social companion. 2

IVA intelligent virtual agent. 1–6, 21–24, 47–51, 81, 84–91, 96–98, 124

KL Kullback-Leibler. 35, 42, 46, 50–52, 60, 64, 69

LHMM Layered Hidden Markov Model. 78

LTM long term memory. 19, 21

LTS long term store. 30, 42, 43, 46, 47

MAP maximum a posteriori. 45, 55, 83, 95

MLE maximum likelihood estimate. 37

MLN Markov Logic Network. 79

MMORPG massive multiplayer online role playing game. 4, 76, 98

MN mixed network. 78, 79

NPC non player character. 4, 6, 7, 76

PFSM probabilistic finite state machine. 38, 82

PMF probability mass function. 32, 33, 35, 39, 40, 43, 49–51

POMDP Partially Observable Markov Decision Process. 78

PPM prediction by partial match. 75

PSDG Probabilistic State-Dependant Grammar. 78

RLE run-length encoding. 75

RMaxS *retrospective maximum surprise*. 40, 48, 53, 56, 60–68, 81, 82, 84, 123, 124

RMinOS *retrospective minimum overall surprise*. 41, 48, 53, 56, 60, 64, 66–68, 81, 82, 84, 123, 124

SDM sparse distributed memory. 26, 27

SM Semantic memory. 1, 2

STM short term memory. 19, 90

TCM temporal context model. 24, 25

UT2004 Unreal Tournament 2004. 15

WM Working memory. 1, 2, 17, 21

WMG working memory graph. 18

WSD word sense disambiguation. 18

List of Figures

List of Tables