# Towards an episodic memory for companion dialogue

Gregor Sieber and Brigitte Krenn

Austrian Research Institute for Artificial Intelligence
Freyung 6/6, 1010 Vienna, Austria
http://www.ofai.at

**Abstract.** *We present an episodic memory component for enhancing the dialogue of artificial companions with the capability to refer to, take up and comment on past interactions with the user, and to take into account in the dialogue long-term user preferences and interests. The proposed episodic memory is based on RDF representations of the agent's experiences and is linked to the agent's semantic memory containing the agent's knowledge base of ontological data and information about the interests of the user.*

**Keywords:** dimensions of intelligence, cognition and behavior; autobiographic episodic memory; relating memory and dialogue

## 1 Introduction

Recently, research on artificial companions has come more and more in focus. They are artificial agents (virtual or robotic) that are intended to support the human user in aspects of everyday life. Application areas may range from virtual agents that assist their users in accessing information from the Internet in accordance with the users' interests, preferences and needs , up to assistive robots in home environments that support the elderly in mastering their life at home.

As regards the dialogue capabilities of companions, approaches are required that allow the agent's mental models and memories to be connected to its expressive behaviour [3], and where natural language dialogue is semantically grounded [1]. Companions need to be aware of their own history and past interactions with their individual users, so that the single user can believe that her/his companion knows "what it is talking about". This is particularly important for creating acceptable long–term interactions.

To account for this kind of requirements, we propose a communication component for companions where autobiographic episodic memory, semantic memory and dialogue are closely connected.

## 2 Related Work

Episodic memory (EM) has first been distinguished from other memory types by [9]. Implementations have e.g. been used in artificial life agents [7], in storytelling agents [5], and for non-player characters in games [2]. Since our memory component is realized as an RDF graph, neither nearest–neighbour search nor retrieval by keywords directly apply. The Adaptive Mind Agent by [6] and Gossip Galore [10] describe companion systems able to answer questions on domain data encoded in RDF. Both agents only have limited knowledge of their own past and do not use it for dialogue. Thus they cannot ground dialogue in their own experiences, and are unable to employ knowledge about user preferences for providing more interesting dialogue. [4] describe a companion system for helping users plan a healthier lifestyle. Dialogue is driven by a daily exercise plan. Our system aims at a more open kind of dialogue which does not revolve around a plan model. This leaves the companion in a situation where much less expectations can be made towards the next user utterance.

In the remainder of this contribution, we will concentrate on the interplay between episodic memory and dialogue. In particular, we describe how the episodic memory is represented, how episodes are retrieved (Sec. 3.1), and how episodic memory is used in the dialogue manager (Sec. 4). For an account of how natural language output is generated from memory content, see [8].

## 3 Episodic Memory

An episodic memory component for companion dialogue needs to provide adequate knowledge representation in connection with the cognitive model and the tasks of the agent. RDF-based[1] data stores are widely used for representing domain knowledge as well as common sense knowledge (e.g. the Open Mind Common Sense Database[2], or ConceptNet[3]). Accordingly, we have developed an episodic memory component for artificial companions that stores and operates on episodes as RDF graphs, and that is interfaced with the agent's semantic memory which must be also composed of RDF triples, making both memories interoperable. We employ a Sesame[4] repository for hosting the data stores.

Our implementation of episodic memory is realized using the Elmo[5] framework, which provides Java to RDF mapping and persistence. A persistence framework such as Elmo significantly reduces the amount of plain RDF data that needs to be generated and parsed within the application. The domain data stored within the episodes is independent of the memory implementation.

For episode retrieval, we propose three different mechanisms, each suited to a different function of episodic memory in our companion.

### 3.1 Episodes

Episodes store the time of their creation in epoch time, the actors involved in the episode, and an episode ID. Input episodes additionally store the user input

[1] `http://www.w3.org/RDF/`
[2] `http://commons.media.mit.edu/en/`
[3] `http://conceptnet.media.mit.edu/`
[4] `http://www.openrdf.org/doc/sesame2/2.3.1/users/index.html`
[5] `http://www.openrdf.org/doc/elmo/1.5/`

string and its analysis in the form of a set of triples, along with a label describing the function of the utterance, such as wh-question or assertion. The analysis of user input is composed of concepts, individuals and relations from the domain ontologies.

Action episodes are a subclass of episodes that represent the actions the agent is capable of. These are: AnswerQuestion: the agent uses domain knowledge to answer a user's question. AssertStatement: The companion attempts to assert user input using the domain knowledge. FindSimilar represents deliberate remembering, i.e. actively searching for similar situations. ExecuteModule: this action is stored when the companion uses one of the modules in the dialogue manager. RetrieveContext is generated when the companion retrieves data from a previous episode to add it to the current input, because it could not be processed otherwise. SendOutput can either convey the results of a query, results of remembering, or details about the situation of the agent, which encompasses e.g. reporting errors or the incapability to solve a certain problem.

Output episodes store the domain data sent to the user, and the name of the template used for output generation, if any.

Finally, evaluation episodes represent positive or negative feedback on previous actions. They are crucial for the agent to be able to learn from its past actions. If an evaluation is available, the agent can decide based on its memories whether a past solution should be repeated or not. Not all episodes have an evaluation. Evaluation values can either come from direct user feedback, internal feedback such as empty query results or failure to retrieve a query result, as well as from an emotional model integrated in the agent.

In order to be able to find the right associations and memories, the agent is capable of translating the time stored in the episodes into relative time categories such as morning, noon, afternoon, yesterday, and so on. This allows the agent to relate episodes to human–understandable fuzzy time categories when receiving input, but also when generating output.

As the retrieval of memories is bound to become inefficient once the episode store grows too big, we have implemented a forgetting mechanism. In our case, forgetting means deleting those episodes that have not been retrieved for a long time. This is similar to deleting the least activated memory, as described in [7]. This approach still bears some risk of losing important memories of situations that are rarely encountered. As an alternative, we consider a blending mechanism for combining redundant or similar information into one episode for our future work, and integrating a model of emotions which could help in deciding which episodes are more important than others.

The retrieval of episodes in our RDF–based model of EM is performed using queries on the RDF store that are able to use properties of episodes, the user model and the domain data, as well as the structural information of the data such as the class hierarchy. Since there are different applications of EM in our system, we present three different retrieval mechanisms: retrieval by similarity, retrieval using patterns in memory, and retrieval of parts of the context for resolution of references in the input.

Retrieval of similar episodes allows an agent to avoid past mistakes, to repeat successful strategies, and to connect and refer to past interactions. Input to retrieval by similarity consists of the set of triples representing the current situation, as stored in short–term memory. First, the memory is searched for identical episodes: a query is generated that searches for episodes that contain the same triples as the input. Additionally, queries using combinations and subsets of the instance set and the set of relations present in the user utterance are issued. For instance, given a popular music gossiping scenario, if the user asks a question about Michael Jackson and Janet Jackson, the agent searches its memory for previous episodes involving both artists, but also using the individual artists, in order to connect to and take up previous discussions. Further broadening the search, the structure of the domain data is used to generate queries containing the classes of the individuals in the utterance. The class hierarchy can be applied, as long as the class is in the domain of the property in the query. For example, talking about the birthday of an *Artist*, the companion can relate this input to episodes about birthdays involving its superclass *Person*, but not for episodes involving its superclass *Entity*, since the class *Entity* has no birthday property.

The episodes retrieved by the queries described above are ranked to retrieve the most similar episode. Ranking is performed by temporal distance and the number of overlapping properties, individuals and classes. Evaluations are found by searching the episodes temporally following each of the episodes until either an evaluation or the next user input is found in memory.

In addition to actively searching episodes identical or similar to the current situation, the companion has a second mode of retrieval that matches patterns in memory. This retrieval mode is used by the dialogue manager to generate output based on the content of episodic memory in combination with the user model, domain knowledge, and the current input.

As an example, some patterns allow the companion to talk about when a certain topic or individual was last discussed. Other patterns allow for detecting preferences of the user: for example, if a property – such as in our scenario, the birth place of an artist – appears very regularly in dialogue. Such preferences can also be extended to include specific values, such as a preference for artists born in New York City. Detection of preferences enables the companion to provide information more focused on the interests of the user. For example, the companion may comment on the fact that a preference is being discussed. Or, it may automatically add the kind of information covered by the preference to other answers, if appropriate. Continuing the example from above: a day after being asked about artists born in New York, the companion might notice while talking about the albums recorded by Billy Joel that he was also born in New York, and communicate it to the user. Alternatively, the companion can ask the user whether she would like to know about other individuals that share this property, and provide a list.

Finally, patterns can match data not encoded in the episodes, but in the agent's knowledge base. For example, consider a dialogue about artists that won certain prizes. If some of the last–mentioned artists share a property and its value with an artist included in the current input – such as being born in the same city – a pattern that looks for such properties matches, and the companion is able to provide this information to the user. How these patterns are utilized in the dialogue manager can be seen in Sec. 4.

Additionally, episodic memories are used by the agent to retrieve information necessary to understand utterances from the user that make reference to previous dialogue. Commonly this problem is addressed by maintaining different lists or stacks of entities and topics. In the following part we present an approach that requires no additional external storage mechanisms but relies on episodic memory. While this is by no means a complete resolution mechanism for anaphoric or elliptic expressions, it does show that it is possible to find candidates for such a mechanism using episodic memories. As an example, a method for resolving simple elliptic references in the type of dialogue encountered in our application scenario is described. For selection of candidates, we use the semantic structure of the utterances, i.e. their RDF representations.

Consider the following snippet of dialogue, where the agent would not be able to answer (3) without further inferences, since the information necessary for generating a query is missing from the input string:

```
(1) User: When was Charlie Parker born?
(2) Agent: Charlie Parker was born on August 29th 1920.
(3) User: And what about John Scofield?
```

The agent can extract from the user utterance that some information about the individual John Scofield is required, but not which information exactly. Using its episodic memories, the agent can retrieve the missing information from the context of the dialogue. With context we mean a set of recent episodes that are relevant to the current conversation. The actual number of episodes to consider is determined by a heuristic which selects up to a fixed number of episodes occurring within the previous and current time of day category (morning, noon, etc.).

Context retrieval is handled in the following way: First, the agent needs to determine whether or not a user utterance requires completion. The agent assumes an utterance to be complete if its analysis contains either one or more full triples, or a triple where one of the nodes is a variable. If a single individual or a single property is encountered, the agent needs to complete the utterance. Since a single individual is encountered in step (3) of the example dialogue, the agent has to complete the utterance. Thus from the input string, the agent only knows that the user has some question about John Scofield who is an artist. This artist could either be the subject, or the object of a statement. Thus for further interpretation of the utterance, the episodic memory is called for. The retrieval of the episode used for completion is done by evaluating a SeRQL query that searches for a) the last occurrence of the same class or superclass in case a property is missing (subject position), b) episodes containing a property whose rdfs:range covers the individual under discussion (object position). In our example, the agent looks for the most recent episode that contains the class of the individual (in this case, *Artist*), retrieves the co-occurring property and adds it to the representation of the current input in the agent's short-term memory.

In case no such episode is found within the context, the agent can generalize the query to a superclass of the individual. For example, John Scofield in (3) is an *Artist*, just like Charlie Parker in (1). However, if we substitute Salvador Allende in (1), the agent needs to look for their common superclass, *Person*, to be able to retrieve context information.

## 4 Dialogue Management

Our dialogue manager is a hybrid approach combining rule–based decision and a scoring approach on a set of modules that search memory and user preferences for relevant information. The rule–based system is used to cover situations such as replying to a greeting or feedback, and also for reporting about errors in the system (e.g. failing to connect to the knowledge base).

For each turn in dialogue the dialogue manager generates a set of modules that encode the kind of recency- and preference-driven patterns exemplified in Sec. 3.1. Modules are generated by inserting information from the user model into pre-fabricated queries. Each module has a unique ID and contains the labels of the templates to be used for generating output. The basic scores of modules are set depending on the complexity of the query and the importance of the user preference. Modules are executed in parallel, and those are discarded for which the query does not finish before a certain temporal threshold, in order to keep the response time limited. Penalties are applied to the modules that have successfully matched, one for each recent occurrence of the module, and one for each negative evaluation. The dialogue manager then executes the action contained in the module and sends the output to the user, possibly requiring feedback for further actions.

## 5 Conclusion

We have presented an RDF–based episodic memory component for enhancing dialogue with the user and grounding domain knowledge in interaction experiences. As a result of our model and implementation, the companion is not only able to retrieve situations similar to the current one from its memory. By searching for patterns in memory, it also can detect and comment on preferences of the user, and automatically provide information relevant to the user. In addition, we have shown how episodic memory can be used to find candidates for resolving references in dialogue necessary to understand the user's input.

Retrieval of episodes is accomplished by using a set of SeRQL queries. Our model shows how the contents of past interactions and their relation to current dialogue can be employed by a companion for selecting the next dialogue move and generating dialogue content. In addition, the connections between the episodic memory and the knowledge base by means of RDF graphs allow for a grounding of knowledge in the experiences of each agent.

**Acknowledgments** The work presented is supported by the Austrian Ministry for Transport, Innovation and Technology (BMVIT) under the programme "FEMtech Women in Research and Technology" grant nr. 821855, project C4U. The Austrian Research Institute for Artificial Intelligence (OFAI) is supported by the Austrian ministries BMVIT and BMWF.

## References

1. Benyon, D., Mival, O.: Scenarios for companions. In: Austrian Artificial Intelligence Workshop (2008)
2. Brom, C., Lukavsky, J.: Towards virtual characters with a full episodic memory ii: The episodic memory strikes back. In: Proc. Empathic Agents, AAMAS workshop. pp. 1–9 (2009)
3. Castellano, G., Aylett, R., Dautenhahn, K., Paiva, A., McOwan, P.W., Ho, S.: Long-Term Affect Sensitive and Socially Interactive Companions. In: Proceedings of the 4th International Workshop on Human-Computer Conversation (2008)
4. Cavazza, M., Smith, C., Charlton, D., Zhang, L., Turunen, M., Hakulinen, J.: A 'companion' ECA with planning and activity modelling. In: AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems. pp. 1281–1284 (2008)
5. Ho, W.C., Dautenhahn, K.: Towards a narrative mind: The creation of coherent life stories for believable virtual agents. In: IVA '08: Proceedings of the 8th international conference on Intelligent Virtual Agents. pp. 59–72. Springer, Berlin, Heidelberg (2008)
6. Krenn, B., Skowron, M., Sieber, G., Gstrein, E., Irran, J.: Adaptive mind agent. In: IVA '09: Proceedings of the 9th International Conference on Intelligent Virtual Agents. pp. 519–520. Springer, Berlin, Heidelberg (2009)
7. Nuxoll, A.: Enhancing Intelligent Agents with Episodic Memory. Ph.D. thesis, Univ. of Michigan, Ann Arbor (2007)
8. Sieber, G., Krenn, B.: Episodic memory for companion dialogue. In: Danieli, M., Gambäck, B., Wilks, Y. (eds.) Proceedings of the 2010 Workshop on Companionable Dialogue Systemsi (ACL 2010). Association for Computational Linguistics (ACL), Uppsala, Sweden (Jul 2010)
9. Tulving, E.: Episodic and semantic memory. In: Tulving, E., Donaldson, W. (eds.) Organization of Memory, pp. 381–403. Academic Press, New York (1972)
10. Xu, F., Adolphs, P., Uszkoreit, H., Cheng, X., Li, H.: Gossip galore: A conversational web agent for collecting and sharing pop trivia. In: Filipe, J., Fred, A.L.N., Sharp, B. (eds.) ICAART. pp. 115–122. INSTICC Press (2009)