Web Mining in the EVA Intelligent Agent Architecture

Philippe Millet, Jean-Claude Heudin
International Institute of Multimedia
Leonardo Da Vinci University – Paris France
Jean-Claude.Heudin@devinci.fr

Abstract

This paper describes the architecture of the fourth version of the Evolutionary Virtual Agent (EVA). This new light-weight java-based implementation is based on a dynamical rule-based subsumption architecture, an XML knowledge base and a scheme kernel for scripting behavior rules. Using this architecture, the agent is able to answer questions in natural language while learning a user’s profile. It also extracts relevant information from the web through search engines queries and use them in the flow of conversation. This paper describes the architecture and analyses its web mining process.

1. Introduction

Evolutionary Virtual Agent (EVA) is a software architecture for designing self-animated conversational characters for applications requiring a human-like interface. Autonomous characters are supposed to respond to human interaction in real-time with appropriate behaviors: no repetitive predetermined answers, broad in content, highly contextual and behaviorally subtle [1]. The character must also appear to think, make decision and act of its own volition [2]. This can be achieved by giving the artificial character a set of features such as an identity and a backstory [3]. In a previous paper, we have described an earlier EVA model designed using these principles [4]. In this paper, we present a new light-weight java-based implementation which is able to extract relevant information from the web and use them in the flow of conversation. The architecture includes a real-time 3D interface for displaying an animated face with emotional expressions, optional speech-to-text and text-to-speech agents, and a behavioral engine. We begin in section 2 by describing the multimedia interface. In section 3, we outline the behavioral architecture’s principles. Then, in section 4, we describe with more details the programming model of this architecture. In section 5, we present the integrated web mining process. We analyze experimental results in section 6. Finally, in the last section, we summarize our approach, we provide conclusions and identify future works.

2. Multimedia interface

An EVA agent consists of two software components and a set of multimedia interfaces. The behavioral engine is the “brain” part of the architecture. It is responsible for all perception, decision and action behaviors. It has access to perception and action interfaces through a networking server. This component enables to connect through sockets a set of interfaces to the behavioral engine. It includes speech-to-text and text-to-speech agents, one or more text-based “chat” clients, and the animated 3D face (cf. fig. 1). More interfaces could be added such as webcams or other sensors depending on the application requirements.

Fig. 2a gives an example of a 3D face in the current prototype. The character face has been designed with the 3DS software and rendered using a java-based real-time engine. Eva’s facial expressions and emotional engine have been described in a previous paper [4]. We are working on other character rendering alternative approaches using 3D high-definition modeling or 2D Macromedia Flash designs. Figure 2b gives an example of a virtual actor prototype.
3. Behavioral architecture overview

The behavioral engine architecture consists of three main software levels. The first level is composed of a set of Java-based core primitives that implements all the essential EVA’s “brain” features such as natural language processing, rule-based system, web mining, emotional engine, etc. The second level is a tiny Lisp interpreter based on the Scheme language [5] for integrating the core primitives in a user-friendly scripting language. The third level is composed of a dynamical layered architecture inspired by the subsumption architecture [6]. Each layer is encoded as a set of behavior rules that take advantages of the underlying level features.

![Figure 2. (a) A simple 3D face expression example on the left. (b) A 3D virtual actor design example (courtesy Hans Meyer).](image)

4. Behavioral scripting

4.1. XML knowledge files

The natural language processing features of EVA requires knowledge files that provide several information such as discussion topics or template expressions. These information are stored in XML files. The syntax of these knowledge files is close to the Artificial Intelligence Meta Language (AIML) [7]. Figure 4 gives an example of the syntax. The topic tag defines the topic of the discussion. Each category includes a set of keywords used for analyzing user’s sentences. A template defines a set of expressions for generating EVA’s answers.

```
<topic name="generic">
  <category name="HELLO">
    <key>hello</key>
    <key>hi</key>
  </category>
  <template name="BYE">
    <expr>goodbye *</expr>
    <expr>see you soon</expr>
  </template>
</topic>
```

Figure 4. Syntax example of XML knowledge files.

Topics may also include re-writing rules which are applied on the user’s entries. Figure 5 gives an example of a rule that systematically replace “you’re” by “you are”.

```
<rule name="R1">
  <get>
    <key>you're</key>
  </get>
  <set>
    <exp>you are</exp>
  </set>
</rule>
```

Figure 5. A re-writing rule example in EVA XML.

4.2. Scheme-based scripting

In addition to XML knowledge files, the EVA architecture enables to script behaviors using the Scheme language. In the following code example (see fig. 6), the program loads the topic “generic”, defines some global variables such as the user’s input sentence, and extracts categories using the topic. Functions format-sentence, parse-sentence and get-categories are typical examples of java-based core primitives.

```
(load-topic "xml/generic.xml")
(define *user-sentence* ""
(define *user-categories* "")
(set! *user-sentence* (format-sentence in))
(set! *user-categories* (get-categories "GENERIC" *user-parsed*)))
```

Figure 6. Syntax example for scripting behaviors.

4.3. Subsumption levels and behavior rules

The EVA architecture is basically a dynamical subsumption model. There are 10 layers numbered from 0 to 9. Layer 0 is the lower level. As in a classical subsumption architecture, each layer of a given level can use the lower levels. Fig. 7 gives the ordering and role of these 10 layers. The EVA’s core primitives includes a set of functions for managing this architecture: mask-level and unmask-level allow to temporarily disable a given layer, swap-levels...
allows to exchange two layers, save-subsumption and restore-subsumption allow to save and restore a given state.

| Level 9: AVOID | avoid discussion or answer |
| Level 8: MINING | web search information |
| Level 7: MEMORY | information retrieval |
| Level 6: PROFILING | learn about users |
| Level 5: ROLE | goals and “job” |
| Level 4: BACKSTORY | what shape the creature |
| Level 3: IDENTITY | who is the creature |
| Level 2: GENERIC | generic interactions |
| Level 1: EMOTION | emotional response |
| Level 0: TRICK | syntax, tricks, bad lang. |

Figure 7. EVA Subsumption layers.

Each layer is coded as a set of behavior rules. A behavior rule is basically a production rule composed of a condition function and an action function [8]. If the condition is satisfied, that is return the boolean value true, then the action is executed. Fig. 8 gives the example of rule “R2” which belongs to the second layer of the subsumption architecture.

```
(define-rule "R2" 2
 '(or (find-sentence *user-categories* "BYE")
    (find-sentence *user-sentence* "see you"))
 'begin
 (show HAPPY 0.5 10)
 (random-template "GENERIC" "BYE")))
```

Figure 9. Syntax example for defining a behavior rule.

The use of the Scheme language along with the EVA core primitives allow to design rich and efficient behavior rules. In addition, it will enable in a future study to easily implement genetic programming for learning new behavior rules [9].

5. Web mining

When the user asks a question that cannot be handled by the basic reactive layers (levels 0 to 5), and the “profiling” and “memory” layers (levels 6 and 7), the subsumption architecture activates the “mining” layer, that is level 8. This layer tries to find a pertinent answer by searching information on the web. The mining process is the following:

1. **Seek:** (a) extract keywords from the user input sentence and (b) launch a query on one or more web search engines; (c) for each result page, URLs are extracted, (d) badly formatted URLs or those that cannot be accessed are removed.

2. **Analyze:** (a) each resulting URL is accessed and downloaded; (b) each file is then cleaned and tags are removed; (c) the resulting text is formatted in regular sentences; (d) we remove any strange sentences, all those that do not contain any of the keywords, those two small or two large, and question sentences.

3. **Select:** (a) each URL file is scored, taking into account the number of co-occurrences of the keywords [10]; (b) URLs are sorted according to their score; (c) and the highest one is selected.

4. **Format:** (a) all sentences in the selected URL file are scored according to the keywords and the presence of a verb; (b) the highest scored sentence is formatted and used as the EVA output.

6. **Experimental results**

We report here three examples of web mining processes initiated by a question from the user. The web search engine were Google and Wikipedia through Google by using and additional keyword “wikipedia”.

6.1. “What are clinical trials ?”

This question leads to a search with “clinical” and “trial” keywords. It results in the following scored URL files:

1. [90] en.wikipedia.org/wiki/clinical_trial
2. [74] www.answers.com/topic/clinical-trial-1
3. [52] 123-physicians-rankingsswicki.eurekster.com/…
7. [0] www.loria.fr/~coulet/ontology/sopharm/…

The final EVA answer is: “Researchers use interventional clinical trials to test hypotheses about the effect of a particular intervention upon a pathological disease condition”.

6.2. “Who is Pascal ?”

This question leads to a search with the “pascal” keyword. It results in the following scored URL files:

1. [110] en.wikipedia.org/wiki/blaise_pascal
5. [2] runeberg.org/authors/pascabla.html

The final EVA answer is: “Pascal’s earliest work was in the natural and applied sciences where he made important contributions to the construction of mechanical calculators, the study of fluids, and clarified the concepts of pressure and vacuum by generalizing the work of Evangelista Torricelli”.

6.3. “What is turbo pascal?”

This question leads to a search with “turbo” and “pascal” keywords. It results in the following scored URL files:

1. [126] en.wikipedia.org/wiki/pascal_...
2. [64] www.answers.com/topic/turbo-pascal

The final EVA answer is: “A derivative known as Object Pascal was designed for object oriented programming. Pascal Paradigm imperative, structured.”

7. Discussion

These examples show clearly that the mining process is able to provide a pertinent answer to a user’s question. The first example “clinical trials” is certainly the best result because of the clarity and pertinence of the answer. However, some problems remain to be solved when we analyze the two other examples. The “Pascal” case answer is not enough pertinent and certainly too long. When we look at the other sentences in the selected URL file, we found the following expression: “Blaise Pascal (pronounced [blez paskal]), (June 19, 1623 - August 19, 1662) was a French mathematician, physicist, and religious philosopher”. This sentence is much better but was given a lower score due to the presence of information in parenthesis. The “turbo pascal” case also leads to a sentence that is not a direct answer to the question.

While the parenthesis problem can be easily fixed by removing this kind of expression, the main problem in this two examples is that there is a lack of information for making a pertinent search. The “pascal” example could be much more efficient if the search is made with an additional keyword such as “blaise” or “philosopher”. This is the same kind of problem for the third example. The solution we propose is based on adding keywords to the web search. This could be done by the profiling and memory layers: by analyzing the context of the question and extracting additional keywords from the user’s profile or the history of interactions. If this process does not provide relevant keywords, then the mining layer can ask an additional question to the user, such as “Do you mean Blaise Pascal?” in our example. Then, a second mining process with the additional keyword “blaise” will lead to the good answer. The problem of too long sentence could be solved by using a size parameter in the scoring function and a better process for formatting the selected answer.

8. Conclusion

In this paper, we have presented a new java-based implementation of the EVA architecture with web mining capabilities. We have described the main components of this architecture. Evidences have been shown that its web mining process results in pertinent answers to user’s questions. Future works include improvement of this web mining process through a cooperation between the mining, profiling and memory layers of the dynamical subsumption architecture.

9. References


