Intelligent Web Navigation Using Virtual Assistants

Eduardo M. Eisman, Víctor López, Juan Luis Castro
Department of Computer Science and Artificial Intelligence
University of Granada (Spain)
{eisman,victor,castro}@decsai.ugr.es

Abstract

Some time ago, companies and organizations did not store very much information about themselves on the Internet. However, the Web has evolved a lot over the last years and websites contain more and more information. In many cases, that information is not well organized and users have to waste their time looking for what they want to know using the traditional menu-driven navigation and keyword search that websites provide. This decreases users' interest in surfing websites and hence in finding out about companies. One way to resolve this problem is to change the structure of websites. However, if a company has just spent a lot of money on redesigning its website or it simply likes its current design, it will refuse to change the way contents are organized. For this reason, we propose a real time Virtual Assistant, which can work together with any existing website, to help users find the information that they look for. Preliminary experiments have shown that Virtual Assistants can outperform traditional navigation.

1 Introduction

With the arrival of the digital era and the development of the Internet over the last years, there are more and more companies that store all the information they own in a digital medium and publish it on the Internet, so that everybody can have access to it. However, due to the big amount of information, it is very difficult for them to structure their contents in a logical way so that they can be accessed using only one or two mouse clicks. Nowadays, this objective is usually a utopia because users have to waste their time looking for what they want to know using the traditional menu-driven navigation and keyword search that websites provide. If the information to handle is small, this problem can be resolved changing the structure of the contents. However, as the information grows, it is more and more difficult to provide users with an easy and fast access to all that data. The necessity of new ways of accessing that information becomes evident.

One of the reasons that make users spend a lot of time wandering through websites to get some useful information arises from the diverse nature of the Internet. Nowadays, there are millions and millions of websites about different topics, and each website is different from the others. For this reason, there is a learning curve when you visit a website for the first time, and the steepness of the slope depends on how well organized the contents are and how skillful the user is. People who are used to working with new technologies can find the information that they are looking for in a website very quickly, even though they have never visited it before. However, other users find it very difficult to navigate through a website using menus to discover new information, and they need somebody to support them.

All these problems make it evident the necessity of an efficient and effective mechanism for organizing and accessing the information of a company when it becomes very big. The new system must be real time. Users' time is highly valuable, so if they had to wait every time they wanted to get some information, the system would be useless. It must be effective, that is, it must achieve the goals and objectives that users want. In this sense, the quality of the results must be high. Moreover, the communication between the system and users must be natural, allowing complete natural language questions rather than simple keywords. Finally, the interface must keep as simple as possible so that everybody can use it, regardless of their computer skills. To sum up, the system must make the access to information easier for everybody.

The rest of this paper is organized as follows. We begin with presenting some related work. We then describe the main features and advantages of our Virtual Assistant. We explain the architecture and the different modules of our system in detail. Then, we describe the user interface and its functionality. Afterwards, we carry out a comparative analysis in order to evaluate the performance of our system in comparison to traditional navigation. Finally, we conclude the paper and provide some directions for future work.

2 Related Work

As far back as the mid-nineties, people became seriously concerned about the recent explosive growth of the Web. Lieberman [1995] pointed out the necessity for some sort of intelligent assistance to a user browsing for interesting information. He introduced a behavior-based interface agent, Letizia, which tracked the user's browsing behavior and attempted to anticipate items of interest. Thus, the system suggested links to other documents, determining an ordering of interest

among them and providing a reason for making those choices, which decayed over time. However, it did not have natural language understanding capabilities.

Wexelblat and Maes [1999] proposed a set of tools, called Footprints. They said that buying a book is not the same as borrowing it. It is the same book (same words, pictures, and organization), but the borrowed book has additional information (notes in the margin, highlights, underlines, dogeared pages, and so on) and reflects its history because it opens more easily to certain places. These traces should be accessible to future users who could take advantage of the work done in the past. With this aim, Footprints showed the traffic through a website using a graph in which nodes were documents and links were transitions between them.

Cassell *et al.* [2000] created a very complete example of Embodied Conversational Agent (ECA) called REA (Real Estate Agent) which played the role of a real estate salesperson. However, it needed a lot of sensors and computational resources so it was not portable.

Abbattista *et al.* [2004] presented SAMIR (Scenographic Agents Mimic Intelligent Reasoning), which consisted of a 3D face, a custom version of the ALICE (Artificial Linguistic Internet Computer Entity) chatterbot [http://www.alicebot.org/], and a classifier system to keep conversation and face expressions coherent with each other.

Kim *et al.* [2005] proposed an information retrieval assistant, CHATTIE, which used natural language and could be integrated with outer information provision systems such as conventional information retrieval and relational database management systems in order to fill some slots in the answer.

AIML (Artificial Intelligence Markup Language) is an XML dialect for describing conversational scenarios for ECAs [Wallace, 2004], which specifies pairs of patterns and templates, so the agent answers the template associated to the pattern that best matches the question. Traditionally, programmers use AIML to create conversational rules by hand, considering the contents of web pages. However, every time a page is updated, related conversational rules have to be modified, and this is a problem when many pages are frequently modified. In order to avoid this problem, Kimura and Kitamura [2006] used RDF (Resource Description Framework) to represent the semantic contents of web pages.

Pilato *et al.* [2008] proposed an intelligent tutoring system for the Java programming language. When a student asked a question, the system looked for the concept that best matched that question and then created a backward path from that concept to the student's current knowledge state. The concepts and their relations were stored in a hierarchical ontology. Relations could be strong (prerequisites) or weak (something related). Most of the subjects were mandatory, although some of them were optional. Each document had a list with the most frequent non-stopwords, used to cluster documents and evaluate if a student knew the concept referred by a document. The learning path was the list of all the unknown nodes.

In conclussion, many of the existing systems, especially those from the earlier years, lack a virtual character which can engage in conversation with users, making the interaction process friendlier. In addition, some systems do not allow users to ask natural language questions. They should also take into account the context during the dialog, so that users could omit the implicit words in the conversation. Moreover, some others systems focus on the navigation history, only trying to organize the pages that have been already visited, instead of recommending related web pages to continue the navigation. These and other problems make the navigation process much more difficult. For this reason, we propose a natural language Virtual Assistant which covers all those features.

3 The Virtual Assistant

The Virtual Assistant is an intelligent system for supporting users when they look for some information in a website. It is a real time system because we must not keep users waiting for getting the information they want. Users engage the system in conversation using natural language queries, as if it was a real assistant, so it is really easy for users without computer skills, and much better than the one offered by traditional websites, where users can only click on static menus and do searches specifying some keywords. In addition, the aim is not only to answer users' questions but also offer recommendations that lead users and keep the conversation going. Context is another important feature. It allows users to omit some words if they are implicit in the conversation.

Next, we explain the architecture of the system and each module in detail, and discuss the features of the user interface.

3.1 The Architecture

Making the access to information easier for any kind of user should be the main objective of our system. For this reason, the Virtual Assistant has been designed using a client-server architecture, as can be seen in Figure 1. In this way, all the computation is done on the server side, so users only need a web browser. This picture also gives us an overview of how the system generates an answer for a specific query. First of all, the user's query is taken from the client to the server using the AJAX (Asynchronous JavaScript And XML) technology. There, it is processed by the Natural Language Understander (NLU) in order to identify the Information Units (IUs) to which it refers (in the next section we explain the concept of Information Unit in detail). Then, the Dialog Manager (DM) receives a list with the identified IUs, including the matching degree between each unit and the query. The DM uses a filter to include some of the IUs from that list in the backpack, which is the structure employed to store all the IUs about which it can talk in the future. Afterwards, the DM chooses the most suitable IU for that moment taking into account the previous dialog, and updates the backpack with the recommendations associated to that IU. Once it has been decided about which IU the Virtual Assistant is going to talk, the DM decides what to say about it. So, it creates a new action for that IU. There are different types of actions: inform, inform and suggest, ask for clarification (if the query is ambiguous), and ignore. Next, the Communication Generator (CG) retrieves and adapts a specific answer which fulfills the selected action. Finally, the generated answer is sent back to the user at the client side using AJAX.

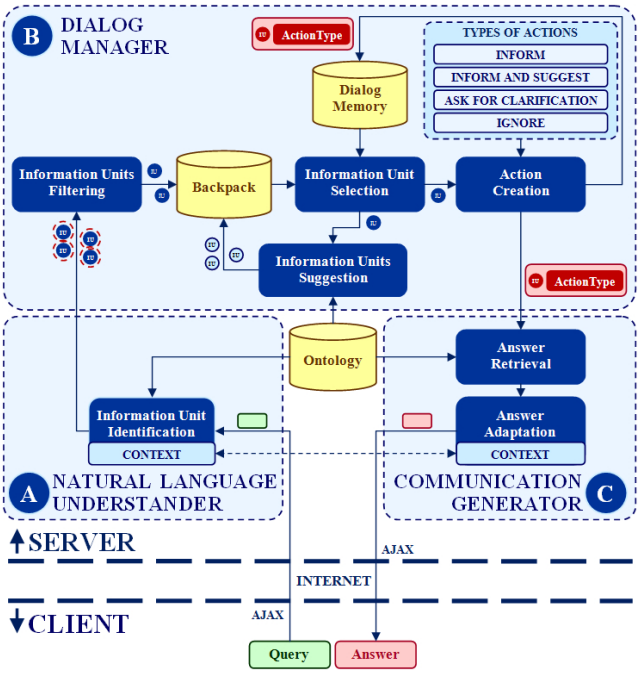

Figure 1: The Architecture

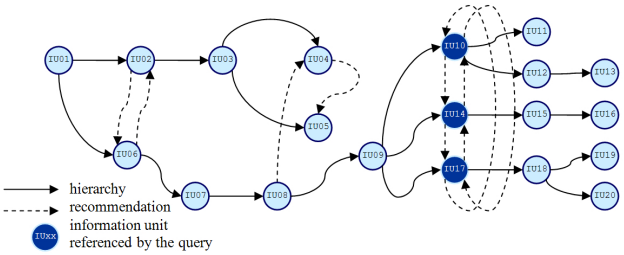

Figure 2: A simplified overview of the ontology

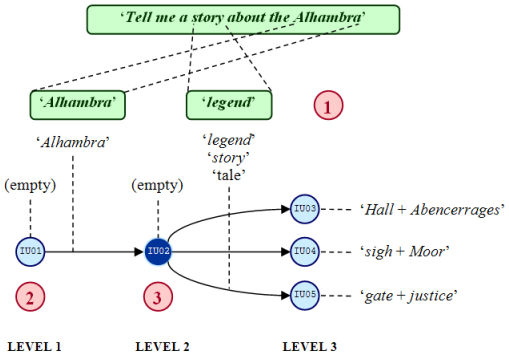

Figure 3: The IU identification process

3.2 The Ontology

All the knowledge about the domain is stored by the system using an ontology. Essentially, an ontology is a formal representation of a set of concepts within a particular domain and some relationships established between those concepts. Our ontology is based on entities called Information Units (IUs). An IU is a piece of information about a specific concept which has a meaning by itself. It includes the definition of the concept and also some meta information like its name, different ways that people can use to ask about it, a URL where more information about it can be found, and so forth. We can distinguish between two different types of IUs, objects and properties. The main different between them is that objects may have properties, but properties must not have either objects or properties. The reason for such a distinction is double. First, we can define templates for the objects of our domain, specifying the names of their properties, so that we do not have to define the properties every time we create a new IU of this kind of object. Second, as we distinguish between objects and properties, when a user asks a question we can identify both of them individually. Therefore, if the question is ambiguous and only the name of the property can be identified, the system can urge the user to concrete the object.

As we can see in Figure 2, IUs are connected with each other. Consequently, we can see our ontology as a kind of graph in which the IUs are the nodes and the connections between them are the edges of the graph. This special graph has some particular features. First, there is a hierarchy defined over it. Thus, IUs can be organized according to different criteria, such as objects included inside other objects, specialization, or any other kind of relationship between them (what we call recommendations).

3.3 The Natural Language Understander

When a user asks a question, the first thing that the system has to do is to identify the IUs that are referred by the question, in order to be able to generate an appropriate answer later on. This is the role of the Natural Language Understander (NLU). Its input is a natural language query, and its output is a list with the IUs that are referred or might be being referred by the query, and which are used by the Dialog Manager (DM) to determine about what to talk, when and how.

The first objective of the NLU is to recognize the keywords of the query that are going to be useful in the next phase. This process is not limited to the textual selection of keywords but it goes further, allowing the use of synonyms, derived words, and so on. The aim is to reduce the language to a specific vocabulary so that it can be easily managed by the system. Remember that our system must be real time. Figure 3 shows an example of IU identification process for a Virtual Assistant for the Alhambra monument. When the user asks the question '*Tell me a story about the Alhambra*', the NLU replaces the word '*story*' by '*legend*', and the word '*Alhambra*' remains constant. These two keywords are used to identify the properties and objects related to the query.

Each property has a set of keywords that represents different ways that people can use to talk about it. In this way, the system moves through all the property names, that are placed at the same level, checking whether they match the keywords or not (each matching word adds one vote to the property).

The process of generating the list of identified objects is different. Since there are much more objects than property names in the ontology, the system must avoid having to move

through all of them in order to be time and memory efficient. For example, if we consider the object '*Legends of the Alhambra*', a user could refer to it saying '*Tell me a tale about the Alhambra*', '*I would like to know a story about the Alhambra*', '*Do you know any legend about the Alhambra?*', and so on. Therefore, we could link the following keyword set to that object: '*tale + Alhambra*', '*story + Alhambra*', and '*legend + Alhambra*'. To avoid redundant information, the system makes good use of the hierarchical structure that objects have in the ontology. In this way, each object has a specific keyword set associated to it, and also an additional keyword set that all its descendants must match. In Figure 3, the specific keywords are linked to the IUs whereas the common keywords are linked to the connections between them. Hence, the system uses these two different sets to move through the hierarchy. If we go on with the example, the NLU starts from the first level, where both the specific keywords of the level (none in this case) and the keywords that all its descendants have in common ('*Alhambra*'), appear in the query. Therefore, the NLU marks this IU about the Alhambra as a candidate to be referred by the query, and descends to the second level through that branch. In this point, the two keywords sets of the IU also match the query, so the NLU adds a new candidate object and descends to the third level, where no keywords match. Consequently, at the end of the process, the NLU has identified two candidate objects which match the user's query ('*Alhambra*' and '*Legends of the Alhambra*').

Once the NLU has identified a list of property names and a list of candidate objects, it joins them, creating new IUs and putting them on the final list of identified IUs. If the NLU cannot identify any logical connection between them, it puts all that objects on the final list and passes the buck to the Dialog Manager (DM). The same applies when the NLU cannot identify any object but only some property names.

Context is another thing to take into account. If our system did not support it and the user wanted to know the price of the tickets for the museum, the question would be '*How much do the tickets for the museum of the Alhambra cost?*' instead of being '*How much do the tickets cost?*', which is much more natural. In order to resolve this problem, the NLU does not start the search from the top IU in the hierarchy but from the last IU that has been transmitted to the user. In this way, the NLU explores the graph below the last IU. After that, it moves to the parent IU and repeats the process but without exploring the last branch again. The searching process continues until the NLU reaches the top IU in the hierarchy.

3.4 The Dialog Manager

The Dialog Manager (DM) makes decisions about what the Virtual Assistant must do, when, and how. Its input is the list of IUs that match the user's query and its output is an abstract action which is later transformed into a specific answer by the Communication Generator (CG).

An action is made of an IU (the concept about which we want to talk) and an action type (the kind of information to provide about that concept). For example, we might want to talk about a particular concept, ask the user for clarification about a set of identified objects or properties, or deny answering about a concept because we have already talked about it. So, we can define different types of actions and specify the behavior of the Virtual Assistant for each one.

Two structures are used during this process. The memory, a chronological list with the actions that have been performed, lets us know if an action has been performed recently or not, how many times we have talked about an IU, and what we have said about it. The backpack stores the IUs about which we have planned to talk in the future, so we can lead the conversation instead of simply answering the user's queries. As we have already mentioned, the DM filters the IUs identified by the NLU so that only the most specific ones are included into the backpack. Next, in order to generate the new action, the DM chooses the next IU from the backpack. Afterwards, some IUs related to the selected one are included into the backpack in case we want to continue talking about related items in the future. Finally, the DM chooses what to say about that IU, that is, it selects the type of action to generate. This new action is added to the memory and sent to the Communication Generator, which generates a specific answer.

3.5 The Communication Generator

The Communication Generator (CG) transforms the abstract action provided by the Dialog Manager (DM) into a specific answer. In other words, it looks for the specific words that are going to be used to carry out that action. Whereas the action is language independent, the answer is language dependent, and it can be expressed in different languages. So, the Virtual Assistant is multilingual.

The answers of the system are templates that can contain up to three different information categories, depending on their updating frequency. Information can be static, immediate, or dynamic. Static information always appears in the answer. It can be fixed (e.g. the history of the Alhambra will always be the same, although each time the specific words of the answer might vary) or variable (e.g. the schedule of the Alhambra is different from November to March than from March to November, so, depending on when the user poses the query, the answer will be either the former or the latter). Immediate information may change quite often, so it should be stored in a database to make updating tasks easier (e.g. the price of the tickets for the Alhambra). Finally, dynamic information is automatically included in runtime (e.g. the time of the system at the moment of generating the answer).

3.6 The User Interface

There is a famous quotation from Albert Einstein which says '*make things as simple as possible, but no simpler*'. Following this line, Figure 4 displays a web page which can be divided into two different areas. Paying special attention to the top of the page, we can identify three areas. On the left side, we can see the Virtual Assistant. She can move the head, blink, and also speak. The text associated to the answer is transformed into speech using the Loquendo program [http://www.loquendo.com/]. This makes the conversation much more natural and similar to a real dialog. In the middle, there are two icons that allow the Virtual Assistant to lead the conversation instead of having to wait until the user asks any question. In addition, there is an input field for asking questions. Finally, on the right side, there are three main

Figure 4: The user interface

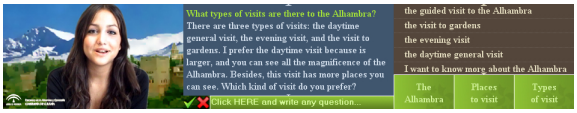

Figure 5: The user interface in action

topics about the application domain. Besides, once the user has asked a question, we can see in Figure 5 how the system modifies the interface in order to show the transcription of the generated answer as well as a list with some different topics related to it, on which the user might click. Finally, the bottom area is used to show web pages related to the query.

4 Comparative Analysis

In order to prove the reliability, effectiveness and efficiency of our system, we present an analysis that compares the Virtual Assistant to the traditional menu-driven navigation and keyword search that websites provide. In particular, we consider the website of the Alhambra (http://www.alhambra-patronato.es/) for this purpose. In testing a system which is designed to support users with an imprecise task such as browsing, it is difficult to find useful measures. We have considered three different tests, and we have employed fifteen IUs about the Alhambra that form a representative subset of all the IUs contained in the ontology of our system: the Alhambra (01), the Generalife (02), the history (03), the schedule (04), the telephone number (05), the types of visits (06), the places to visit (07), the Court of the Lions (08), the Palace of Charles V (09), the Hall of the Abencerrages (10), the prices of the tickets (11), the purchase of tickets (12), the accesses (13), the exhibitions (14), and the museum (15).

All these tests have been carried out on an Intel Pentium IV at 2.40 GHz, with 1.00 GB of RAM, Microsoft Windows XP, and Apache Tomcat.

Test One: The Menu-Driven Navigation

The first test compares the navigation using the recommendations proposed by the Virtual Assistant to the navigation using the menus of the website of the Alhambra.

Table 1 shows the minimum number of clicks that a user has to make in order to have access to the IU associated to the identifier of the corresponding column of the table. As we can see, the difference between both systems is very small. This is a menu-driven navigation and, in general, you can go wherever you want with two or three mouse clicks at the most. In the case of the telephone number of the Alhambra (IU05), the number of clicks is 0 because it appears at the bottom of all the web pages. Actually, through the *sitemap* of the website of the Alhambra, we can have access to any of those IUs with only two mouse clicks. However, users without an advanced knowledge about computers do not usually know what a sitemap is and therefore it is highly improbable that they will use this feature of the website. So, if we consider the averages that appear in Table 1, the Virtual Assistant performs slightly better (20%) than the website of the Alhambra. The main reason is that the website contains more information about the Alhambra than our system and for this cause it uses more levels to organize the data. However, this makes the usual information more difficult to be found and users might easily get lost along the way. In other words, two mouse clicks on the Virtual Assistant take less time than two clicks on the website of the Alhambra, because in the second case you have to find out to which category the information that you are looking for belongs.

The problem of identifying the category of the information that we are looking for deserves special attention. We are considering the best possible situation, that is, users know perfectly where all the information is placed in the website, even though it is the first time they visit it. However, this is an ideal situation because they often wander through websites to get some useful information, and this problem becomes more apparent as the user's computer skills decrease. For the users who have a high skill level (experts), the improvement achieved by the use of virtual assistants is really low because they are used to navigating with traditional menus. What is more, they could prefer traditional navigation because they can find the information faster. However, as the skill level decreases, users find it more difficult to move through websites, and here the Virtual Assistant can be really useful.

Using an analogy, the Virtual Assistant is like a Global Positioning System (GPS). When people are used to driving in a specific city, and they always go to the same places, they do not need a GPS. What is more, they could find it very stressful if they wanted to go to a place taking a short cut, and the GPS recommended another alternative path because it thought that it is the best. However, nobody can throw doubt on the fact that GPSs are really useful for people visiting a city which is completely unknown for them, because they can guide them through such an amount of streets.

Test Two: The Searching Process

The second test analyzes the searching power of both systems, considering the amount of time that searches take.

In order to measure the speed of the search engines, we have queried both systems about the fifteen IUs proposed. The results have shown that the Virtual Assistant outperforms in time the search engine of the website of the Alhambra in 340%. The main reason of such a strange huge difference is that the website probably performs a bad sequential search, reading all the documents once. On the contrary, our system

Table 1: Number of clicks needed to reach some IUs

System	# Information Unit (IU)															Average
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
Virtual Assistant	1	3	2	2	2	1	1	2	2	2	2	2	2	2	3	1.9
Alhambra's website	1	2	2	3	0	3	2	3	3	3	3	3	2	3	3	2.4

Table 2: Number of results per query found in the website

ID	Query[1]	#Docs[2]	#Docs[3]	%[4]
IU01	Alhambra	0	20853	-
IU02	Generalife	81620	8320	89.8
IU03	history	19768	1583	92.0
IU04	schedule	1118	395	64.7
IU05	telephone number	183	9	95.1
IU06	types visits	5	3	40.0
IU07	places	1036	9	99.1
IU08	Court Lions	20803	1063	94.9
IU09	Palace Charles V	58953	5169	91.2
IU10	Hall Abencerrages	2245	32	98.6
IU11	prices tickets	97	3	96.9
IU12	purchase tickets	1	1	0.0
IU13	accesses	24388	6464	73.5
IU14	exhibitions	13975	17	99.9
IU15	museum	105898	11252	89.4
	TOTAL	330090	55173	83.3

[1] Translation of the query from Spanish
[2] Results found in all the website
[3] Results found in the specific category to which the IU belongs
[4] Reduction in the number of results

takes advantage of the hierarchical structure of the ontology to index the search. However, these results must be handled carefully because the website stores more information than our system, so the comparison is not completely fair.

Another point that deserves special attention is that the Virtual Assistant allows contextual natural language questions rather than simple keywords, as it happens with the website.

Test Three: The Quality and Reachability of the Information

The aim of this test is to evaluate the quality of the information provided by both systems, that is, if it is relevant to the query and users do not have to waste their time looking for what they want to know among all that information.

With regard to the amount of information retrieved, Table 2 shows the number of results found by the website of the Alhambra for each query. Note that the results for the query '*Alhambra*' in the first search are really 0 (perhaps too many documents for that search string). In general, if we limit the search space, the reduction in the number of documents is over 80%. Even so, there are too many results. However, what is really important is not the amount of documents retrieved, but the fact that some times the results do not contain the requested information. In the case of the Virtual Assistant, the answer is always immediate.

5 Conclusion & Future Work

In this work we have studied the problem of the reachability of the information on websites. When the information grows, it is difficult to organize it in an appropriate way so that users can easily find it. For this reason, we have presented an intelligent natural language Virtual Assistant which makes the access to information easier specially for inexperienced users.

Regarding the future work, we must make the Virtual Assistant much more dynamic and with different personalities, so that it could behave in different ways and choose between several paths for generating the answer. Another point to take into account is the adaptation of the recommendations to the particular user of the system, according to his/her preferences. Finally, it would be a good idea to assist the navigation all over the website, not only when the user interacts with the system in a direct way.

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Technology under Research Project TIN2007-67984-C02-01, the Andalusian Government under Research Project TIC-P06-01424, and a FPU scholarship from the Spanish Ministry of Science and Innovation.

References

[Abbattista *et al.*, 2004] F. Abbattista, G. Catucci, G. Semeraro, and F. Zambetta. Samir: A smart 3d assistant on the web. *PsychNology Journal*, 2(1):43–60, 2004.

[Cassell *et al.*, 2000] J. Cassell, J. Sullivan, S. Prevost, and E. Churchill. *Embodied conversational agents*. MIT Press, Cambridge, MA, USA, 2000.

[Kim *et al.*, 2005] H. Kim, C. N. Seon, and J. Seo. A dialogue-based information retrieval assistant using shallow nlp techniques in online sales domains. *IEICE - Trans. Inf. Syst.*, E88-D(5):801–808, 2005.

[Kimura and Kitamura, 2006] M. Kimura and Y. Kitamura. Embodied conversational agent based on semantic web. In *PRIMA*, pages 734–741, 2006.

[Lieberman, 1995] Henry Lieberman. Letizia: An agent that assists web browsing. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[Pilato *et al.*, 2008] G. Pilato, R. Pirrone, and R. Rizzo. A kst-based system for student tutoring. *Applied Artificial Intelligence*, 22(4):283–308, 2008.

[Wallace, 2004] R. Wallace. The elements of aiml style. alice ai foundation, 2004.

[Wexelblat and Maes, 1999] A. Wexelblat and P. Maes. Footprints: history-rich tools for information foraging. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 270–277, New York, NY, USA, 1999. ACM.