# ANALYSIS OF THE CHATBOT OPEN SOURCE LANGUAGES AIML AND CHATSCRIPT: A Review

**Preprint** · February 2017

DOI: 10.13140/RG.2.2.34160.15367



**3 authors**, including:

Sasa Arsovski
Raffles University Iskandar
**28** PUBLICATIONS **4** CITATIONS
SEE PROFILE

Idris Muniru
University of Ilorin
**8** PUBLICATIONS **2** CITATIONS
SEE PROFILE

# ANALYSIS OF THE CHATBOT OPEN SOURCE LANGUAGES AIML AND CHATSCRIPT: A Review

**Sasa Arsovski [1]*, Adrian David Cheok [2], Muniru Idris [3], Mohd Radzee Bin Abdul Raffur[4]**

[1]Imagineering Institute, Johor, Malaysia (sasa@imagineeringinstute.org)
[2]Imagineering Institute, Johor, Malaysia (adrian@imagineeringinstitute.org)
[3]Imagineering Institute, Johor, Malaysia (idris@imagineeringinstute.org)
[4]Imagineering Institute, Johor, Malaysia (radzee@imagineeringinstute.org )

**ABSTRACT**
Human-Computer Interaction that characterizes dialogue between man and computer is gaining momentum in computer interaction techniques. This type of program is called a Chat-bot. This paper presents a survey on the techniques used to design a Chat-bot. The authors discuss similarities and differences in chat-bot implementation techniques and analyze most used open source languages deployed in the designing of chat-bots (AIML and ChatScript). The purpose of the paper is to give a technical overview of those two languages and provide comparisons between them according following parameters: Ease of implementation and complexity of language, access to external resources, knowledge acquisition, linking to customized ontologies and the possibility to build chat-bot for a Mobile application.

**Index Terms**: AIML, ChatScript, Chat-bot, NLP, Ontology

## I. INTRODUCTION

The last few years have seen unique peak in interest in chat-bots. Microsoft bot "Tay" and the decision of Facebook to integrate chat-bot capabilities into Messenger sped up development of numerous new chat-bot platforms. Some of them are:

Microsoft Bot platform - Microsoft announced that it was bringing chat-bot capabilities to Skype and launched bot builder for Node.js a framework for constructing your own bots.

ChatScript - launched in 2011 as a language for the new generation of chat-bots. ChatSript engine won a Loebner Prize four times. It provides an open source C++ framework for developing, building and employing chat-bots.

Pandorabots - Is an online service that lets developers build, host and deploy chat-bots. Pandorabots uses AIML (Artificial Intelligence Mark-up Language) which is an open standard for writing chat-bots and open source Java framework.

Facebook Bots for Messenger - Facebook launched this tool which allows developers and businesses to build chat-bots for its Messenger platform.

Rebot.me - Is a simple service that enable users to build, teach and deploy chat-bots on their web sites.

Imperson - Disney Accelerator 2015, Imperson, a created chat-bot generator for building and deploying and managing chat-bots via Facebook, Skype, Twitter.

Chat-bot technology is not new. Chat-bots appeared almost a decade ago when their design and method of use were defined. Therefore, it is reasonable to ask the question "Why have a number of initiatives to create a chat-bot recently appeared?"

The answer is the following: The technology is changed and evolved. Computers are faster and smart phones are available to everyone. Numerous internet services provide ubiquitous connections among people, information and software. Large amounts of human knowledge are collected and available. Due to the increasing amount of information, techniques for automatic analyzing that information has become an important research topic.

Authors [1] analyze current chat-bots and state that any machine, system or program (chat-bot) exhibiting some, but not all fundamentals of intelligence are a Partially Intelligent System. Therefore, chat-bots are Partially Intelligent Systems. Moreover, today's machine intelligence is greatly confined to the boundaries of Partial Intelligence.

It raises the question "Which research topics and technologies do the current chat-bots not use in the most optimal way"?

Research in Natural Language Processing (NLP) reached their peak intensity after a widespread release of commercial chat-bot applications. These chat-bot-NLP applications are currently challenged in their ability to understand the intention of users within their diversity in syntax and semantics. In a chat-bots, the Natural Language Understanding (NLU) is referred to as the unit where the system extracts meaning from the input text. NLU comprises everything related to machine reading comprehension. In chat-bot applications, the NLU component represents a way to transform input into a semantic representation.

There are a lot of projects in ontologies, knowledge organization and knowledge representation. These initiatives are aimed to apply reasoning functionality into chat-bot engines. Semantic reasoners can infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms to work with. Numerous projects working

on organizing knowledge.

Natural Language Processing, Natural Language Understanding, knowledge organization and knowledge representation are things that current chat-bots do not use enough to their best advantage.

Today, chat-bot usage introduces different types of commercial applications such as: Personal assistants; Travel agents; Customer service representatives; Technical support; Automatic call routing. Chat-bots in this application are enhanced with speech processing.

There are several technologies available for use in creating a chat-bot. The choice between markup or scripting language depends on the skill, experience and functionality you want to achieve.

Our goal in this study was to identify technology for building a chat-bot that fulfill technical requirements of today's ubiquitous computing. The ideal technology should meet the following requirements: To be based on an open source platform; To be simple enough and easy to implement; To be able to connect with the customized ontologies and have support of the free source community. The selected technology should enable the construction and implementation of a chat-bot on Android and iOS platform to be usable for mobile app.

In this paper, we will analyze those two most used open source language deployed in the designing of chat-bots. Each of those two languages has their Pros and Cons and the preferred choice is determined by the application which the chat-bot is intended to support. We will give a technical overview of those two languages. Also, we will provide comparisons of those two languages according following parameters: Ease of implementation and complexity of language, access to external resources, knowledge acquisition, linking to customized ontologies and possibility to build chat-bot for a Mobile app.

The paper is organized as follows: the second section provides a background of chat-bot technology. The third section provides an overview of the AIML 1.0 and AIML 2.0. The fourth section provides an overview of the ChatScript language. The fifth section presents a comparison of AIML and Chatscript language. In the final, sixth section, concluding remarks are given.

## II. BACKGROUND

In 1950 Alan Turing published an article [2] proposing the Turing test as a criterion of intelligence. This criterion evaluates the ability of a computer program that communicates in real time with a man to effectively mimic human interaction, enough to be indistinguishable from a human conversational partner. Joseph Weizenbaum's program ELIZA, published in 1966, could fool users into believing that they were conversing with a real human. Authors [3] agreed that the best way to facilitate Human Computer Interaction (HCI) is by allowing users "to express their interest, wishes, or queries directly and naturally, by speaking, typing, and pointing".

Numerous chat-bot architectures have been developed since then. Some of them are: MegaHAL [4], CONVERSE [5], ELIZABETH [6], HEXBOT [7] and ALICE.

A.L.I.C.E. (Artificial Intelligence Foundation) [8] is the Artificial Linguistic Internet Computer Entity, which was first implemented by Wallace in 1995. Alice's knowledge about English conversation patterns is stored in AIML files. AIML, or Artificial Intelligence Mark-up Language, is a derivative of Extensible Mark-up Language (XML). It was developed by Wallace and the Alicebot free software community from 1995 onwards to enable people to input dialogue pattern knowledge into chat-bots based on the A.L.I.C.E. open-source software technology.

A.L.I.C.E. is the forerunner of third-generation chat-bots with more advanced Pattern Recognition techniques.

Authors [9] stated that the third generation of chat-bots is based on the concepts of Pattern Recognition, or Matching Pattern technique. It is applied to natural language modelling for the dialogue between humans and chat-bots that follow the stimulus-response approach. For this purpose, a set of possible user inputs is modelled and, for each one of these sentences (stimuli), pre-programmed answers were built to be shown to the user.

The ALICE (Artificial Linguistic Internet Computer Entity) chat-bot was the first to use the AIML language and interpreter. In ALICE, the AIML technology was responsible for pattern matching and to relate a user input with a response of the chat-bot Knowledge Base (KB).

## III. AIML

### *A. AIML 1.0*

According the author [8] the aim of AIML language is to simplify the modelling process of dialogue. AIML benefits are access to stimulus-response. In addition, AIML is an XML-based markup language. AIML defines a class object that is responsible for modelling the patterns of conversation. AIML is the most-used chat-bot language due to its simplicity, ease of learning, ease of implementation and the availability of pre-authored AIML sets. It is a simple word pattern-matcher.

The AIML robot responds according connection between the questions set by the user and knowledge located in AIML files. Reasonable interaction of the users and computers is determined by means of construction of knowledge.

AIML has its advantages and disadvantages. Some of the advantages are: Easy to learn and implement, simplicity and user-friendliness of the system of dialogue, the use of XML for the formal, computer readable representation of knowledge.

Some of the disadvantages are: Knowledge is presented as an instance AIML files. If knowledge is created based on data collected from the internet will not be automatically updated and must be updated periodically. Original AIML has no possibility of extension.

AIML has relatively poor matching patterns and it is difficult to maintain. Although, the content is very easy to enter, the main problem is the large amount of content which must be manually entered to create a functional chat-bot.

In this paper, focus will be on the AIML 2.0. AIML 2.0 is a completely new chat-bot language that fits all requirements of new generation of chat-bots.

*B. AIML 2.0*

AIML 2.0 is motivated by the technological and social changes of mobile era, the ALICE A.I. Foundation has released the AIML 2.0 specification [10].

1) Ease of implementation and complexity of language

AIML 2.0 is an attempt to address the shortcomings, while balancing the original goal of keeping the language as simple as possible. The AIML 2.0 specification is, for the most part, designed to be backwards-compatible with the AIML 1.0 and earlier standards, in that way preserving the simplicity of the original language.

One of the most important improvements in AIML 2.0 is pattern matching. In addition to the wildcards _ and * already displayed, # is now available. These matches on 0-N words and thus allow more compact spellings of frequently used matching rules. Previously, in the absence of 0-N wildcards, several patterns were necessary to intercept the cases where the word is at the beginning or end of the sentence [10]. The sets and maps also introduced in version 2.0 [11] are used to integrate knowledge from external files.

Sets are sets of objects, which are stored line-by-line as plain text in the embedded file. The set tag is used to access the set by filename in the pattern. In the case of a hit, this can then be included in the template with the already known star tag. Sets and maps have a higher priority than and * when pattern matching, but the remaining operators (including the exact hit) have priority.

Maps contain assignments from one term to another, for example, a list of countries and their main cities. Especially about the sets, there is a powerful possibility to define a lot of knowledge and a large amount of interactions with the chat-bot in a small space. In AIML 1.0, to achieve the same functionality, a separate category would have to be created for each entry.

As an extended possibility to intervene in the control flow, explicit loops were introduced in AIML 2.0 with the loop tag. The syntax is somewhat unconventional; In concept, it is a mixture of Do-While-Loop and Go-To-Statement.

2) Access to external resources

AIML 2.0 introduces <obb> tags - out of band commands for device Virtual Assistant responses. Those tags enable Dial, Send SMS, Send email, search, maps etc. Also, AIML 2.0 enables access to the Wolfram Alpha, Trueknowledge.com, Answers.com, Weather Service, Shopping sites, other chat-bots and Netbase. Netbase is a semantic web database. Netbase knowledge Graph contains over 600,000,000 nodes and statements, from all the Internet's sources, including Freebase, Wikidata, DBPedia, Yago, Wordnet, as well as many custom GraphDBs. Netbase is much more than a triple store: it has the Wordnet ontology and its relations built-in, so that you can quickly query with synonyms, transitive class hierarchies, part meronyms etc. [12]. In AIML 2.1 there is a facility to create ontologies that enable the chat-bot to make use and reason with knowledge. AIML can extract domain knowledge from semantic Web ontologies using a scripting language called OwlLang and to store new knowledge obtained from the conversations in the ontologies [13]. Listing 1. Shows code for accessing external resources.

Listing 1.

```
<category>
<pattern>WHAT IS THE WEATHER</pattern>
<template>
<sraix service="pannous">WHAT IS THE WEATHER
</sraix>
</template>
</category>
```

The category element defines a question-response pair; the question is defined within the pattern tag pair (<pattern></pattern>) and the response in the template tag pair (<template></template>). The chat-bot response could be from categories within its own knowledge base, those of other bots or other knowledge sources. The *sraix* element allows a chat-bot to call categories that exist within another chat-bot, and return response as if it was its own. In the code that is shown on Listing 1. the chat-bot reports weather information retrieved from external sources provided by the pannous web service. Panonus web service aggregate knowledge from DbPedia, WordNet, Weather Service, Shopping sites etc.

3) Knowledge acquisition

The *learn* element allows the user to generate new category blocks from within a conversation. This powerful AIML 2.0 functionality allows users to teach the bot new information. Categories generated by the learn element are stored in memory, and they are only accessible with the client_name that was used to create them [14].

There is also a tool called Pattern Suggester that is part of Program AB, the most recent reference implementation of AIML 2.0. Pattern Suggester helps to automate the process of creating new patterns through a type of unsupervised learning [15].

The *learnf* element is identical to the learn element in syntax, however, the generated category is written to an AIML file that you may specify with the property learn filename. Learning example is given in the Listing 2.

Listing 2. AIML 2.0 Learning Example

```
<category>
```

```
<pattern>* LIKE *</pattern>
<template>
   I will remember that you like coffee.
   <learn>
      <category>
      <pattern>WHAT DO * LIKE</pattern>
      <template>You like <star/></template>
      </category>
   </learn>
</template>
</category>
```

In the code that is shown on Listing 2. category tag defines a question-response pair. The keyword LEARN instructs chat-bot to create a category for the question-response specified within the <learn> element, to be able to provide response to such pattern subsequently. The <star/> tag is used to retrieve variables (denoted by *) supplied by the users to the chat-bot.

4) Linking to customized ontologies

Program AB is a Java application. Program AB is structured to enable developers to easily extend AIML with custom tags. Program AB implements AIML 2.0 features including the capability to connect to remote bots and web services through the new tag.

Program AB implements some memory optimizations that make it possible to run a chat-bot on a mobile device or embedded system. Program AB serves as reference implementation of the AIML 2.0.

Implementation of the personal written ontology files in the chat-bot database is simple task with JENA library. Another way is to use script shown on the Listing 3.

Listing 3. Querying the customized ontology

```
<category><pattern>SPEEDOF *</pattern>
<template>
<get var="? x">
<tuple><first><select>
<q><subj><star/></subj><pred>hasSpeed</pred><obj>
?x</obj></q>
</select></first></tuple>
</get>
</template>
</category>
```

In the code that is shown on Listing 3. the SPEEDOF keyword defined in the <pattern> tag, trigger the query defied in the <template> tag for the entered entity variable (denoted with *). Query variable is initialized in the <getv var="? x> tag. In the tag <q>, query code is presented. Query returns *hasSpeed* attribute value of the entity variable entered in the <pattern> tag. Entities and their attributes are defined in the external file named *tripels.txt*. that is shown on the Listing 4.

Listing 4. Triples.txt file

```
CEETAH:hasPurpose:to hunt wildlife
CEETAH:hasSize:7
CHEETAH:hasSpeed:10
CHEETAH:hasSyllables:2
CHEETAH:isa:Animal
CHEETAH:isa:Wild Animal
CHEETAH:lifeArea:Physical
```

5) Possibility to build chat-bot for a Mobile app

The CallMom app for Android is the first mobile assistant app to incorporate multiple chat-bot personality choices and a sophisticated learning feature. In 2013, the successor app CallMom became the first virtual assistant to embed the AI function directly on a mobile device [10]. The AIML 2.0 specification includes several features that support development of mobile apps with chat-bot functionality.

## IV. CHATSCRIPT

ChatScript is a scripting language designed to accept user text input and generate a text response. Chat proceeds in volleys, like tennis. The program inputs one or more sentences from the user and outputs one or more sentences back. ChatScript is a system for manipulating natural language, not just for building a chat-bot [16]. ChatScript starts by transforming input words using substitutions files. It has files for texting, spellings, common spelling mistakes, contractions, abbreviations, noise and interjections mapped as speech acts. These are all in live data, meaning they are not stored into the dictionary but loaded on start-up. Trailing punctuation is removed, with bits set to reflect punctuation status (question, statement, exclamation) [17].

1) Ease of implementation and complexity of language

Since ChatScript is both a scripting language and an engine, implementation can be somewhat more difficult than AIML.

Fundamentally a script is a series of rules. A full rule has a kind, a label, a pattern, and an output.

Rules can be restricted to statement inputs, question inputs, or only when the chat-bot takes control of the conversation and wants to volunteer something (gambits). Rules are bundled into collections called topics [16].

Topics can invoke other topics. Exactly how to process an input is controlled by a control script which itself is a topic.

A pattern is a set of more specific conditions which allow or disallow this rule, usually trying to match words of the current input sentence, but sometimes considering the prior history of the conversation, the time of day, or whatever [16].

The output is what this rule does if allowed to execute. Since the goal of the overall is to generate a response, the simplest output is merely the words to say.

More complex outputs can do conditional execution, loops, function calls, etc. The system normally executes rules in a specified order until one not only passes the kind and pattern restrictions, but also actually generates output destined to reach the user. Once one has output,

the system is done [16].

2) Access to external resources

Built-in WordNet dictionary for ontology and spell-checking. WordNet is a lexical database of the English language created and maintained at the Cognitive Science Laboratory of Princeton University. It groups words into sets of synonyms—synsets—each expressing a distinct concept. WordNet gives short, general definitions and records the semantic relations between synsets. ChatScript has ability to read structured JSON (JavaScript Object Notation) data from websites and Postgres support for big data or large-user-volume chat-bots [16]. ChatScript has several routines for calling external resources. Script code shown on the Listing 5. enable acquiring and processing data from the any URL (unified resource location) or Web.

Listing 5. Requesting data from the Web

***^tcpopen (kind url data 'function)***

3) Knowledge acquisition

ChatSript has an option for memorizing certain kinds of information and use that information in the future conversations. ChatScript memorize parts of chat and store it in the long-term data (variables and facts). When the variable once set, its written with user data unless are explicitly erased. Variable can be used all over the chat. The variable also remains for other rules to trigger on.

4) Linking to customized ontologies

ChatScript can create facts—subject, verb, object triples— and to organize them into tables. The data saved as facts and in tables may then be recalled using queries. According our findings, ChatSript has possibility to connect to the external ontologies and acquire knowledge from the ontology and write this knowledge into local concept files.

5) Possibility to build chat-bot for a mobile application

To build mobile application, ChatScript should be embed within another program, meaning that ChatScript is not the main program. In this context, the main program is controlling the mobile application, and invoking ChatScript for conversation or control guidance. Depend of mobile application platform, the ChatScript memory should be reduced. The full dictionary may take 25MB. For mobile applications, dictionary should be minimized.

## V. COMPARATIVE REVIEW OF AIML AND CHATSCRIPT

Since the goal of this study was to identify the best technology for building our personal chat-bot, first we chose chat-bot languages that are based on the open source platform. Next step was comparison of selected languages per requirements that are defined in the introduction section.

Table 1. shows the comparison of AIML and ChatScript according the following parameters: To be simple enough and easy to implement; To be able to connect with the customized ontologies and have support of the free source community. The selected technology should enable the construction and implementation of a chat-bot on Android and iOS platform to be usable for mobile applications.

Table 1. Comparative review AIML and ChatScript.

| | **AIML 2.0** | **ChatScript** |
|---|---|---|
| **Ease of implementation** | Very simple process of dialogue modelling | Implementation is more difficult than AIML |
| **Complexity of language** | Balancing between keeping the language as simple as possible and the technological and social perquisites of mobile era. | Complex, system for manipulating natural language and conversation agents |
| **Connection to external resources** | Freebase, Wikidata, DBPedia, Yago, Wordnet | WordNet, ConceptNet |
| **Learning** | Defined syntax for learning new patterns and templates. | Yes, variables and facts |
| **Personal ontology implementation** | JENA library | Yes, script code |
| **Mobile App** | YES | YES, but with restrictions |

ChatScript is created to provide natural language understanding capabilities for characters in games, but has also been used for the chat-bot. ChatScript is meaning-based and have supporting sets of words called concepts to represent synonyms.

If the user goal is to create the illusion that chat-bot understands the user, this means that he will try to minimize instants where chat-bot says something completely unrelated to what the user said and maximize the instants when the chat-bot responds completely appropriately, then the ChatScript is a right solution.

AIML is a widely-used markup language for specifying chat-bots. AIML 2.0 has a numerous new feature as [18]:

The <topic> tag can now be set on a category to make it easier to add categories to topics. Zero or more wildcards - new ^ and # pattern wildcards that match zero or more words. Pattern priority - new $ pattern marker to make a pattern word match highest priority. Attribute tags - Any template tag attribute can now be set using a sub-element. Sets

New pattern side <set> tag to evaluate a pattern based on words defined in a predefined set.

New template <map> tag to allow the lookup of the element value in a predefined mapping, returning the mapped value.

Condition patterns - wildcards can now be used in condition values to provide default conditions. Condition loops

New template <loop> tag to loop a conditional statement. Local variables

New <var> attribute for variables scoped to a category. Remote requests:

New template <sraix> tag to make a remote request of another bot instance, or service. Normalization and denormalization - New <normalize> and <denormalize> tags to convert special character into words, and back again. Date formats - New formatting options for dates. Request - New template <request> tag to return the user's previous input request. Response - New template <response> tag to return the bot's previous response. Learning - New template <learn>, <learnf>, and <eval> tags to dynamically train a chat-bot with new responses. Explode - New template <explode> tag to split a word into its characters. Mobile tags - New <oob> (out of band) tag to support client and mobile device commands.

AIML 2.0 introduces the program AB. Program AB implements AIML 2.0 features including Sets and Maps, Zero+ wildcards, and the capability to connect to remote bots and web services through the new tag.

Program AB implements some memory optimizations that make it possible to run a chat-bot on a mobile device or embedded system. Program AB has already run on Android phones and tablets.

If the user goal is to build customized chat-bot that fulfill technical requirements for ubiquitous connections among people, information, software and follow the social changes caused by mobile era, then is a AIML 2.0 is a logical choice.

## VI. CONLUSION

In this paper, we gave a technical overview of the two most used open source languages for building a chat-bot. We provided comparisons between those two languages according to the following parameters: Ease of implementation and complexity of language, access to external resources, knowledge acquisition, linking to customized ontologies and possibility to build chat-bot for a mobile application. AIML 2.0 specification fulfills the need for new features but keeps AIML as simple as possible, especially for non-programmers.

From the survey, above, it can be said that the technologies for development and implementation of chat-bot are still widespread and no common approach has yet been identified. The use of technology depends on developer decision.

Study on today's chat-bots shows that chat-bots need design improvements such as an advanced NLP and NLU, advanced pattern recognition approach, more comprehensive knowledge bases, better knowledge organization and knowledge representation.

Use of the personal customized ontologies which can provide reasoning functionality and define specific personality as trough recording behavior patterns from history of the conversation should be a subject of further research.

## REFERENCES


***Journal Articles***

[1] A. Khanna, B. Pandey, K. Vashishta, K. Kalia, B. Pradeepkumar and T. Das, "A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence", IJUNESST, vol. 8, no. 7, pp. 277-284, 2015.

[2] A. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE", Mind, vol., no. 236, pp. 433-460, 1950.

[9] M. Bruno Marietto, R. Aguiar, G. Barbosa, W. Botelho, E. Pimentel, R. Franca and V. da Silva, "Artificial Intelligence Markup Language: A Brief Tutorial", International Journal of Computer Science & Engineering Survey, vol. 4, no. 3, pp. 1-20, 2013

***Books & Book Chapters***

[5] Batacharia, B., D. Levy, R. Catizone, A. Krotov, and Y. Wilks. "CONVERSE: a conversational companion." In Machine conversations, pp. 205-215. Springer US, 1999

[15] M. McTear, C. Zoraida, and G. David. "Conversational Interfaces: Devices, Wearables, Virtual Agents, and Robots." In The Conversational Interface, pp. 283-308 Springer International Publishing, 2016

***Conference Proceedings***

[3] Shawar, Bayan Abu, and Eric Atwell. "Chatbots: are they really useful?." In LDV Forum, vol. 22, no. 1, pp. 29-49. 2007

[4] Hutchens, Jason L.; Alder, Michael D. (1998), "Introducing MegaHAL" (PDF), NeMLaP3 / CoNLL98 Workshop on Human-Computer Conversation, ACL (271): 274

[13] Lundqvist KO, Pursey G, Williams S, "Design and implementation of conversational agents for harvesting feedback in eLearning systems". In: Hernandez-Leo D, Ley T, Klamma R, Harrer A (eds) Scaling up learning for sustained impact. Lecture notes in computer science, vol 8095, pp 617–618., 2013

[17] Ramon López de Mántaras Badia, ARBOR Ciencia, Pensamiento y Cultura, Vol. 189-764, no 2013, a086 ISSN-L: 0210-1963, doi: http://dx.doi.org/10.3989/arbor.2013.764n6009)

***Technical Reports***

[6] Abu Shawar B. and Atwell E. 2002. A comparison between ALICE and Elizabeth chatbot systems. School of Computing research report 2002.19, University of Leeds

***Online Sources***

[7] HEXBOT chatbot website. http://www.hexbot.com/

[8] Richard S. Wallace, The Elements of AIML Style, ALICE A. I. Foundation, Inc. March 28, 2003

[10] Wallace, R. S. (2014a). AIML 2.0 Working Draft, https://docs.google.com/document/d/1wNT25hJRyupcG51aO89UcQEiG -HkXRXusukADpFnDs4/pub

[11] (Wallace, AIML - Sets and Maps in AIML 2.0, https://docs.google.com/document/d/1DWHiOOcda58CflDZ0Wsm1CgP3Es6dpicb4MBbbpwzEk/pub

[12] NetBase API documentation, (https://market.mashape.com/pannous/netbase)

[14] AIML 2.0 Reference, http://callmom.pandorabots.com/static/reference/#elements/-lt-learn-gt)

[16] Bruce Wilcox, ChatScript Basic User's Manual,Revision 11/18/2013 cs3.73, http://chatscript.sourceforge.net/Documentation/ChatScript%20Basic%20User%20Manual.pdf

[18] What's new in AIML 2.0, https://www.botlibre.com/forum-post?id=1156738