Technical University of Crete

Department of Electronic and Computer Engineering

A Mixed Initiative Q&A and Chatterbot System

Maraggouli Afrodity

Diploma Thesis

Thesis Committee:

A. Potamianos (Supervisor)

V. Digalakis

E. Petrakis

Chania, 2009

... to my uncle

Acknowledgements

Initially I would like to sincerely thank my supervisor professor A. Potamianos for trusting me with this diploma thesis and for the significant guidance and help that he offered me until the last moment of this work. I would also like to thank the rest of the professors in my thesis committee, E. Petrakis and V. Digalakis. It would be a great omission not to thank the Ph.D. candidate Elias Iosif for his constant support and the valuable help that he offered me. In addition I would like to thank all the post graduate students being under professor's A. Potamianos supervisory who provided me their advises and help whenever I asked for them. The first person that I would like to thank for the enormous encouragement that she gave me and for being by my side during all this time is my friend Stavroula Founta. Also I would like to thank the rest of my friends that truly stood by my side, even without their physical presence, this period of my life. I would like to thank the guys that accompanied me during the many hours of my work in the lab and colored this period with pleasant incidents. At last I would like to thank my family for their patience and support which I will never be able to reciprocate.

Abstract

In this thesis is presented the implementation of a mixed initiative question answering (Q&A) and chatterbot system. The initial goals of this work was to increase the chatterbot's initiative during the dialogue along with giving the system some attributes that enables it to demonstrate performance similar to Q&A systems. The specific implementation is an extension of an already existing chatterbot system that consists of an AIML knowledge base and an AIML interpreter. The first application that was accomplished in the previously mentioned terms was a time counter that detects the user's input inactivity in order to encourage the user to continue the existing dialogue. Another attempt to increase the chatterbot's initiative was the accomplishment of an application that enables the system to detect in the course of dialogue a previously discussed topic in which the user reenters according to his utterances. Eventually an application that enables the system to detect the user's loss of interest was added in the initial system. The specific application resolves the detected state of the user by suggesting a subject which is produced from further procedures. These procedures involve a data base with biographical documents concerning significant personalities of various domains from where the stored information is submitted into certain information retrieval techniques along with the a part of the dialogue history that consists of the user's utterances that preceded the time of this detection.

Περίληψη

Κατά τη διάρκεια της εκπόνησης της συγκεκριμένης διπλωματικής εργασίας υλοποιήθηκε ένα διαλογικό σύστημα ικανό να επιδείξει χαρακτηριστικά παρόμοια με αυτά που επιδεικνύει ένας άνθρωπος ο οποίος κατέχει την θέση του συνομιλητή. Η είσοδος του εκάστοτε χρήστη λαμβάνεται από το πληκτρολόγιο διατυπωμένη σε φυσική γλώσσα και η έξοδος γίνεται αποδεκτή από το χρήστη στην οθόνη του διατυπωμένη επίσης σε φυσική γλώσσα. Σκοπός λοιπόν αυτής της διπλωματικής εργασίας ήταν το διαλογικό αυτό σύστημα να επιδεικνύει ορισμένη πρωτοβουλία κατά την διάρκεια του διαλόγου του με τον εκάστοτε χρήστη. Επίσης στη συγκεκριμένη υλοποίηση προστέθηκαν στοιχεία που καθιστούν το σύστημα ικανό να προσφέρει πληροφορίες οι οποίες απορρέουν από βιογραφικά σημειώματα σημαντικών προσωπικοτήτων τοποθετημένα στην γνωστική του βάση. Η υλοποίηση αυτή βασίστηκε σε ένα ήδη υπάρχων σύστημα και αποτελεί επέκταση αυτού. Το τελικό διαλογικό σύστημα έχει πλέον την ικανότητα να ανιχνεύει την επιστροφή του χρήστη σε παλαιότερο θέμα συζήτησης με σκοπό να το επισημάνει στον τελευταίο. Επίσης σε περίπτωση που η είσοδος του χρήστη παραμένει ανενεργή για ορισμένο χρονικό διάστημα τότε το σύστημα ζητά από το χρήστη να συνεχίσουν την ροή της συζητησής του. Η τελευταία προσθήκη που έγινε αφορά την ικανότητα του συστήματος να αντιλαμβάνεται την περίπτωση οπού ο χρήστης κατά την διάρκεια της συζήτησης χάνει το ενδιαφέρον του και μέσα από πολλαπλές διαδικασίες να του προτείνει ένα άλλο θέμα συζήτησης.

Contents

Introduction

Scanning through the scientific and technological achievements of the previous years it would be impossible not noticing that the main focus of the scientists is on inventing new methods or improving the already existing ones in order to facilitate human life. Working under this aspect eventually these attempts were focused on inventing machines which can perform similarly to the human nature and in many cases replace human's existence. Of course the resulting products of these attempts had various effects in human life. The use of them in some cases accomplished the facilitation of the daily life, though in other cases triggered work problems in people as the human existence was set unnecessary by the use of machines. Deviating though from the social effects of these achievements, the scientific labor and the remarkable technological progress being succeeded through this labor should be underlined. Scientists in their attempt to copy human behavior in order to construct machines that would perform processes having the same results as those that derive from the human's effect took a step forward and achieved the construction of machines that can imitate the human way of participating in a conversation which is conducted in natural language. These conversational agents are able of demonstrating intelligence when engaging in a dialogue with a human user. These specific computer programs are widely known as chatterbots or chatbots and their main goal is to conduct an open domain conversation with the other side user using textual or vocal techniques.

There have been many interesting implementations of chatterbots that can perform in similar ways with the human performance in a conversation and the place of the pioneer is possessed by the first ever constructed chatterbot Eliza. While Eliza could be involved in a simple conversation relying on simple pattern matching rules and a simple knowledge base, the technological progress brought to the light many other similar machines with additional properties much more interesting than Eliza's. As a result scientists introduced to the world machines with learning abilities through the interaction with the human user and machines that serve the needs of information retrieval through conversational process which are conducted in natural language. These latter involve the combination of chatterbots and question answering systems.

This thesis presents the design of a chatterbot system combining also some properties that are found in a question answering system. The implementation was based in an already

existing chatterbot, thus the resulting implementation is an extension of the previous. The initial goals were to extend the chatterbots implementation in order to increase the dialogue initiative of the chatterbot. Working under this aspect this thesis includes the accomplishment of an application that is able to detect a previously discussed subject of conversation and mention it to the user, an application of a time counter which with the arrival of the timeout encourages the user to continue the existing conversation, and an application that tries to detect the user's loss of interest during the conversation and suggest another topic for discussion. The latter application involves also techniques applied for information retrieval. When the user's loss of interest is detected then according to the previous user's utterances there are two possible stages. The chatterbot can either inform the user on the biography of a significant person relative to the suggested subject or just respond to the user by informing him on a significant event which has happened the same date that the specific conversation is conducted, or just return to the user a witticism that an important man said once.

In the first chapter of this thesis the first ever implemented chatterbot Eliza is presented in detail and similar implementations accomplished after Eliza are presented in order to show the progress on the field. The second chapter is dedicated to the question answering systems in order to comprehend the applied techniques on them and introduce the variety of information retrieval methods. The third chapter constitutes a brief tutorial of the Artificial Intelligence Markup Language which has been used in the specific work in order to import the knowledge in the chatterbot. The forth chapter presents in detail the work that have been made in order to result in the implementation that this thesis concerns. The fifth chapter consists of the evaluation process that was applied in the implemented system and also the results extracted from it. Finally the last chapter presents the conclusions extracted from the systems performance and the future work that can be applied in order to improve it.

1 Chatter bots

1.1 Introduction

A chatterbot or chatbot, as it is commonly known in the web, is a type of conversational agent whose design is based in a form of artificial intelligence. This type of computer program aims to imitate a conversation conducted in terms of human nature, with the significant and interesting extension of replacing the one participant. The communicative conditions under which this dialogue is accomplished rely on either text typing or acoustic methods. Throughout the interaction with such a program, the user gets the illusion that the program is intelligently interpreting the human input prior to providing an appropriate response. This user's misconception is desirable, as the intention of such a program is to deceive the user by giving him the fake impression of discussing with a real person. This misleading conversational performance of a chatterbot is trying to externally approach human conversational manners.

Although a good understanding of a conversation is apparently required for a human to follow up a meaningful dialogue, this does not apply to most chatterbots. The prevalent designing technique of a chatterbot is grounded on the principle of recognizing cue words or phrases from the user's input and reaching the resulting response through a matching process. The matching process is conducted with the use of pre-prepared or pre-calculated responses which are stored in a local database. Most chatterbots simply scan for keywords within the user's input and pull a reply with the most similar wording pattern from the local database. For example a simple approach is a chatbot to be programmed to answer the user's input "I am feeling sad today", by recognizing the phrase "I am" and replacing it with the phrase "Why are you", followed by a question mark at the end. This would result in answering with the question "Why are you feeling sad today?". Of course the more sophisticated a chatterbot is, the larger the database of words and phrases needs to be, and likewise the larger the chatbot logic will be.

The above programming technique of personal pronoun transformations was introduced by Eliza the first chatterbot ever constructed. There are other programs classified as chatbots that use other principles than simple pattern matching. Some chatbots attempt to model the human way of learning facts and language, and as a result they are increasing their knowledge bases

through the interaction with users. Some others use natural language processing in their attempt to provide the user with more meaningful answers.

1.2 Eliza the first chatterbot

Eliza is the first computer program classified as a chatbot which introduced the idea of human's textual communication with a machine in the form of natural language. Eliza was written by Joseph Weizenbaum between 1964 and 1966 (1). The program is simulating a Rogerian psychotherapist and operates by processing users' responses to scripts which consist of some decomposition and reassembly rules. Eliza's general operating technique is that rephrases the user's statements and poses those as questions to the other side user.

Eliza's first implementation is written in MAD-Slip for the IMDB 7094. The user can communicate with Eliza by simply typing a statement or a set of statements in natural language using the ordinary punctuation and grammatical rules. In brief the procedure which the program conducts in order to result in an appropriate response is the below:

- The user's text input is read and inspected for the existence of a keyword.
- During the keyword scanning execution certain unconditional transformations are being made if needed.
- If a keyword is retrieved through the user's input then a sentence transformation follows according to a rule associated with the keyword. Eventually the resulting sentence is printed out as a response to the user.

Although the above procedure doesn't appear to be burdensome, as its details are being examined more closely through this chapter, the complexity of such an attempt will be revealed.

The transformation rules (decomposition and reassembly), which are mentioned above, are a number of Slip functions. These functions are being used to decompose the input string according to a certain criteria and figure out which criteria satisfy and afterwards to reassemble the decomposed sentence according to appropriate assembly specifications. Eliza has the ability to recognize a comma or a period as a delimiter. Whenever either of them exists in a user's input and a keyword has already been identified, then the rest of the text is being discarded. If no keyword had yet been found, the whole input including the delimiter is

discarded. Hence only single phrases are ever transformed. To avoid unnecessary details an example of sentence transformation is apposed below:

User' input: It seems that you hate me

Decomposition rule:

(* YOU * ME)

User's input after the application of the rule:

(a)It seems that (b) you (c) hate (d) me

Reassembly rule:

(What makes you think I (c) you)

The "*" in decomposition rule represents an indefinite number of words while (c) in the reassembly rule represents the 3rd component of the decomposed sentence. Similar logic is applied for all the transformations.

Of course there is a predefined structure that represents the relation between keywords and the transformation rules, in order to detect the rule that matches the input sentence. The basic format of this structure is a list consisted of a keyword and its associated rules.

$$\begin{array}{c}(K((D_1)(R_{1,1})(R_{1,2})...(R_{1,m_1}))) \\ ((D_2)(R_{2,1})(R_{2,2})...(R_{2,m_2})) \\ \vdots \\ ((D_n)(R_{n,1})(R_{n,2})...(R_{n,m_n})\end{array} \qquad (1.1)$$

Where:

K is the keyword

D_i is i^{th} decomposition rule associated with the keyword K

$R_{i,j}$ is the j^{th} reassembly rule associated with the i^{th} decomposition rule.

This structure consists of 2 levels. The first and top level is consisted of the keywords followed by the names of lists which are the second level. This level is again a list structure that contains a decomposition rule followed by reassembly rules. This keyword dictionary is

constructed with the use of a particular key vector. Each word that is going to constitute a keyword is being hushed through a specific procedure and placed in its unique position in this key vector. Then the keywords' rules are added in this structure in adjacent positions to the keyword. Hence the identification of a keyword is executed with the same way the keyword was inserted in the structure. The programming time is eliminated as a result of this process.

We have mentioned before that certain word substitutions are made during the input's scanning time. In order to be consistent to this, any word in the key dictionary may be followed by the word to be its substitute interjecting the equal sign (=). At this point it would be necessary to mention that keywords have ranks. This derives from the fact that each word in a sentence holds different semantic weight. The procedure of the identification of a keyword has as a result a pointer to the list of transformation rules associated with the keyword to be found. Taking though into consideration the keywords' ranking, this procedure gets more complex. A list of keywords is used to store the previously mentioned pointers. When a word is identified as a keyword and its ranking is higher than that of any previously encountered word, then the pointer is placed on the top of that list of keywords which is called keystack, otherwise it is placed on the bottom of the keystack. Eventually on the top of the keystack will be placed the pointer associated with the transformations rules of the higher ranked keyword to be identified.

In order to avoid repetitions while attempting to match the input text according to the selected decomposition rule, there is a mechanism that insures that every reassembly rule will be tested only once whether it matches or not. Of course there will be the case that every decomposition rule of the highest ranked keyword will fail. The reassembly rule (NEWKEY) attempts to overcome this serious problem. Whenever this rule is invoked the keyword in the top of the keystack is removed and discarded and the next keyword with the highest rank is recovered so as the whole transformation process to reinitiate.

It couldn't be unremarked the case where no keywords remain to be tested for transformation when every previous attempts have failed. This problem arises when the (NEWKEY) reassembly rule is invoked and the keystack is empty or in the case where no keywords were identified in the user's input. In order to cope with this failure every script of ELIZA includes in the keystack the reserved keyword "NONE" which is associated with the 'catch all' decomposition rule. This decomposition rule is followed by content-free remarks in the form of transformation rules.

Concluding the presentation of the framework to which ELIZA is constructed, needs to be mentioned that there are more other mechanisms that are included in this work, but omitted from this chapter in order to be more comprehensive.

1.3 From early to recent chatterbots

Having presented in detail the implementation of Eliza, in this section the route of chatterbots through the following years will be lightened. Some of the most famous chatterbots will be presented in order to understand the progress made on this field. Of course in order to be comprehensive the chatbots which are examined closer below are only a few representatives of the work that it has been done.

1.3.1 Parry

After Eliza introduced itself as a pioneer in the world of chatterbots, many others made their appearances being based upon its initial model. An early entrance in the circles of chatterbots was made by Parry written in 1972 by psychiatrist Kenneth Colby (2) (3). In contrast to Eliza, instead of simulating a psychotherapist, Parry was a computer program which reflected the mind of a paranoid schizophrenic who was being under the examination of his doctor. His creator indented on using the program as a teaching system for students of psychiatry, before they were let loose on real patients. Parry was constructed using a custom design knowledge data base, like Eliza's, but positive or negative ranking was applied to words or phrases in the flow of the conversation. Parry's responses were based upon the emotional state of it, caused from the user's input.

1.3.2 Jabberwacky

After Eliza and Parry many other chatterbots were constructed. An interesting one is Jabberwacky (4). Its purpose is to accomplish a conversation in an entertaining and humorous way without demonstrating a specific behavior such as Eliza and Parry which simulate the behavior of a psychotherapist and a paranoid schizophrenic respectively. Jabberwacky is based upon a particular form of artificial intelligence to which it owes its severalty. Instead of having a preformed knowledge base as most of chatterbots have, Jabberwacky has no knowledge to language and is designed with the remarkable quality of being able to learn

everything from the ground up. This noticeable characteristic gives Jabberwacky the ability of manipulating several different languages without the requirement of any extra coding.

While a conversation is carried out Jabberwacky stores the dialogue history in log files. This occurs in every conversation and as a result the conversation base grows. Jabberwacky In order to respond in each user's input uses contextual pattern matching techniques. It uses the stored information and through the interaction with users it learns more. Its design relies on the principle of feedback. This freedom of learning any language, any linguistic idiom and borrowing any dialogue habit from the participant in the conversations makes it capable of developing a vary of personalities. Of course it's reasonable that such systems are depending entirely on their teacher for the developing of their personalities. One of these Jabberwacky personalities won the bronze prize in 2006 at the Loebner awards. It was described as 'sassy' because of the sarcastic way that it responded to user's questions whose answer was very obvious. The conclusion is that when a chatterbot relies on human responses to build its knowledge, this fact helps in appearing more realistic.

1.3.3 ALICE

ALICE (Artificial Intelligence Computer Entity) is another recent chatbot introduced by Richard S. Wallace in 2003 (5). This is a conversational agent that engages a human in dialogue. The difference between ALICE and the other existing chatterbots lies on the fact that this specific chatbots implementation is based upon AIML (Artificial Intelligent Mark-up Language) , a language similar to XML. The advantage of this chatterbot is the simplicity of AIML. This is a language whose the basic units are categories. Each category has a pattern which is used in the matching process of the user's input and from the template that is associated to each category the appropriate response is retrieved. So ALICE's implementation is based on prefixed input/output rules written in AIML. For programmers who already know HTML, working on ALICE in order to build an implementation such as a chatterbot is rather simple. Of course ALICE lacks the ability of self learning as the artificial intelligence which is used for its implementation can be classified as week. Further down AIML is examined in a more detailed way, as it is a part of our own application.

1.4 Artificial Intelligence from the perspective of chatterbots

Artificial intelligence (AI) is a complex field where the first attempts focused on the development of machines being able to behave exactly like humans even in the emotional section and to duplicate human intelligence. This idea was abandoned easily as researchers had to cope with the fact that intelligent behavior is more complicated than initially considered. This realization had as a result the separation of artificial intelligence in two sub categories, the strong artificial intelligence and the weak artificial intelligence.

The strong artificial intelligence as it can be assumed by the characterization involves the complexity of this field relied on the researchers' first belief that it can be possible to create a thinking machine. This form of AI includes the combination of clever programming and complex algorithms. Jabberwacky which was analyzed above can be classified in the machines that use some form of this AI.

On the contrary ALICE is a chatterbot based upon the principles of weak artificial intelligence. This form of AI focuses on modeling intelligent behavior in a modest way. Computer programs that use this kind of AI lack of sapience and reasoning ability and their operation depends on pure pattern matching.

1.5 Turing Test

A significant test that a chatterbot or generally a conversational agent can undergo is the Turing test (6). It is essential to be able to measure the ability of a chatbot to demonstrate intelligence and this serious issue was stated from Allan Turing in 1950. Turing Test is supposed to be inarguably one of the most disputed topics in artificial intelligence and philosophy of mind. Some believe that his work was the foundation of artificial intelligence and some other finds it useless. Turing was not in life to witness the realization of his own idea but followers of his beliefs and work turned this idea into reality.

The question that torments for decades the scientific communities of artificial intelligence, philosophy and psychology, is "what is intelligence?" or in other words "how can it be measured?" . Some philosophers during these years expressed the idea that the intelligence lies behind the ability of someone to deal with various symbol systems as those of mathematics. In the contrary others placed the definition of intelligence in a more emotional area. The latter seem to define intelligence as a reflection from feelings, personality and

morality and so on. Turing believed that the only practical mean to observe intelligence is behavior.

Turing in his work begins posing the question ‘Can machines think?’, which is until nowadays regarded a very ambiguous one. In his attempt of being more tangible and provide a method to answer the previous question, he proposed the so called “Imitation game”.

The ‘imitation game’ as introduced initially by Turing consisted of three participants a man, a woman and an interrogator. The interrogator is placed in separate room from the other two participants. The whole procedure that is about to take place is conducted through a teletype connection. The interrogator communicates with the rest of the participants in natural language. The underlying dialogue consists of any imaginable topic and there are no limitations. The interrogators goal is to distinguish without any doubts which one of the participants is the woman. The other two participants join the game having in mind that they should convince the interrogator of their female sex regardless of their actual sex.

Turing though, moving one step further, retracted his initial question and in its place posed a new one, having the form of ‘What will happen if a machine replaced one of the two participants?’. The present issue that is under consideration now has nothing to do with gender identification. The real issue now is transferred to whether or not the interrogator will be able to distinguish the machine from the human. As a result if a machine confused with its responses the interrogator and made him uncertain on his decision, the machine would be considered intelligent as its intelligent contestant.

Turing restrains the participation of all machines in the test and he permits only ‘digital computers’. As ‘digital computers’ he considers those who consist of the following three parts:

- Store
- Executive unit
- Control

The store corresponds to the data base of the machine which may consist of information and rules that the machine should follow. The executive unit is the part that consists of all the operations which employ every calculation that might need to be done. The control is the part which checks if the rules of the store are being followed.

Of course there are many versions of the Turing tests today and it cannot be determined which of all was intended by Alan Turing. Turing in his work did not clarify many things that left the space for disputes. It is regarded as standard interpretation though, that the computer to take part in the Turing test is examined for its ability to imitate human rather than to deceive the interrogator. The previously mentioned Eliza and Parry have succeeded on passing the Turing test and many others after them.

Loebner prize, which was previously mentioned, is a chatbot competition that lies upon the ideas introduced by Turing. In order to be accepted as a reliable competition many modifications were made in the original Turing Test. In this competition the central idea is the same as this on the Test. Hence, judges engage in dialogues with both humans and machines and rank these conversations according to how convincing they are. The contest though deviates from the original test in many ways. The topic to be under discussion in each conversation can be only one and the alternations are not permitted. There also restrains on the way the interrogator will handle the dialogue flow. This includes the use of tricky techniques like the use of gibberish, or repetition of questions during the conversation.

1.6 Real-life applications that use chatterbots

A reasonable consequence of the design of such computer programs as chatterbots is their utilization in several aspects of real life. A chatterbot's qualifications indicate its usage in several domains in order to replace human occupation or partially facilitate him. Chatterbots are being used for educational reasons as tutors of languages. Relying on the fact that chatbots employ well the language which they use to communicate, they offer services such as practicing of a language or even teaching. In addition to these services, chatterbots provide also customer services in places frequently visited by the public, such as libraries. The type of the service in this case can be to either help the visitors be informed of the library's provided services in the web or introduce them to the available electronic information. Chatterbots can be used in commercial fields playing the role of the shopping assistant and helping clients to reach appropriate e-commerce sites according to the seeking product. The use of chatterbots expands everyday due to their promising qualities that are improving day after day.

1.7 Summary

This chapter was an introduction to the interesting field of conversational agent's development. The very first chatbot introduced in the field of technology, Eliza, was explicitly analyzed and the successors of it were presented revealing the progress in this domain. The principles on which such operational engines rely were examined closer in order to understand their construction techniques. Various attempts of getting closer to human behavior through machines lacking reasoning and sense were unfold and confirmed Turing's prediction that machines could perform intelligently some day. The place that these achievements of artificial intelligence have in real life was presented in order to indicate that the progress accomplished in this field is significant even in everyday life.

2 Question Answering Systems (Q&A)

2.1 Introduction

Passing through the years, someone would observe as much the remarkable scientific evolution as the technology process, which led humanity in covering the century of information. It would be impossible not noticing the dominance of information in every aspect of the present daily life. In order to be able to deal with this irrational pace of information quest, researchers casted upon the flowering field of information retrieval. The fruits of their labor are the many and widely known information search engines. Nowadays most of people at least once in their life, had to address to such an engine to receive a satisfying answer of any kind of question which they might had.

Web based search engines, such as Google, are the endued carriers of the vast amounts of information that are available to be reached when it is needed. So given this enormous amount of information, these engines seem quite appealing as resources for answering a variety of questions. It is rather disappointing though, the fact that these kinds of engines have a very restrained way of accessing their stored information. It would make no difference if the submitted statement in such engines would have the form of a statement expressed in natural language or the form of a query consisted of the most representative words concerning the subject of the research. The engine in both cases will manipulate the input in the same way and produce the same form of information. Not only casual users but also the professional ones are frequently overwhelmed by the amount of information being return to them whenever they use these search engines. The reason why this is happening can be found in the hundred of thousand documents that users are flood with and they must wade through them to conclude in the subject of their research.

The above reasons led researchers to move a step forward in the field of information retrieval and introduce the emerging technology of question answering. This form of information retrieval appears vary promising as its aim is to provide more intuitive methods of information access. In contrast to the regular widely known search engines that follow the principles of formulating queries and browse results in the form of documents, the model adopted by question answering systems is the simple one of accepting the user's request expressed in natural language and return a response including the precise answer. Although its

apparent simplicity during the close examination of this kind of systems their complexity will start to unfold.

2.2 General framework of a Q&A system

After having introduced the actual orientation of a Q&A system, it is time to describe the common techniques followed by anyone who attempts to build such a system. Researcher who engage in building Q&A systems focus their concentration on the bellow targets

- Question type identification
- Answer type identification
- Information retrieval and answer extraction

2.2.1 Question type identification

Being engaged in a natural language conversation, someone in order to respond correctly to a question posed by the other side participant, needs to understand in depth the requested subject. As a consequence of this, a system lacking reasoning it would be impossible to expect it to perform in such a manner. Apparently the same question phrased in natural language has several ways in which it can be expressed, still though conserving the same meaning. Despite the occurrence of the above, a factoid question expressed in any possible way consists of some distinct elements that enclose the meaning of the question. An obvious and helpful approach is to create a kind of question type taxonomy and try to index a question among the several predefined types of this. These distinct elements of a question that were mentioned above contribute to the construction of this taxonomy. As a result of this are the types of questions that can be easily recognized such as the question beginning with ‘when’, ‘where’ or ‘what’. On the other side though there are also questions that are unique and cannot be easily classified into types. This kind of questions needs more complex logic.

2.2.2 Answer type identification

This next step of answer type identification is strictly related to the previous process. It would be reasonable to believe that an answer correspond to a specific question most of the times will be a rephrase of this question. The redundancy of information though that lies in the web comes to contradict this initial belief. Since the answer will be retrieved from a text written in natural language the same problems as in the process of the question type

identification are about to be revealed. As a result of the above the answer should be classified in more than one type, in order to avoid rejecting context that might be related to the question and include the correct respond.

2.2.3 Information retrieval and further contextual processing

This part of a Q&A system's framework involves the previously mentioned information search engines. After defining the question type and extracting the question keywords to formulate a query, this query is applied to a search engine in order to retrieve documents that possibly contain the requested response. Of course these documents are submitted into further processing responsible for retrieving textual snippets that correspond to the type of the question and the type of the answer. Moreover techniques are applied in order to rank these textual snippets or even prune them into smaller textual pieces such as paragraphs or sentences. In order to accomplish the answer extraction several various techniques might be applied. Such techniques are the application of the N-gram model to manage the sentence or phrase ranking or the part-of-speech tagger to enable recognition of the answer candidates. Some of these techniques are presented below in more detail.

2.3 Implementations of Q&A systems and various techniques applied on them

In this section an exploration of the different existing implementations of Q&A systems is going to be attempted. The reader will also be surveyed through the various techniques applied from researchers in order to develop Q&A systems. The complexity of these systems is going to be revealed through their close examination.

2.3.1 Answer type generation through learning surface text patterns

The work to be presented here concerns the production of surface text patterns from predefined question types that can be used for spotting the correct answer (7). In other words this implementation suggests an approach of constructing automatically an answer types table corresponding to several standard question types. These produced text patterns consist of regular expressions formulated during the process to be presented below.

The initial step of the underlying procedure is the application of a pattern – learning algorithm on many examples of each distinct given question type. The description of this specific algorithm can be summarized as follows:

1. According to the specific type of the question used as example, the question term and the answer term are determined. In this step the several ways that an answer term or a question term can be encountered have to be taken under consideration. This would result in recognizing all the several ways as the determined term.
2. The latter terms are submitted after being formulated as a query to the search engine in order to download the 1000 top ranked documents related to this query.
3. Retrieved documents are separated into sentences from the responsible application.
4. Sentences that contain either the answer or question term, or even none of them determined in the first step are discarded. The remaining sentences are applied to further procedure such as html tags removal and white space handling in order to require the appropriate form for being able to be submitted to regular expression matching tools.
5. Then the use of suffix trees on the remaining sentences follows. This operation is responsible of indentifying the useful answer substrings depending on the repeated word ordering in the sentences being under examination. Another useful element produced during this procedure is the number of the sentences responsible for the appearance of these substrings.
6. The resultant phrases from the previous step are filtered in order to keep only those which contain both the question and answer term.
7. In order to serve generality the replacement of the question term and the answer term by general tags such as <NAME> and <ANSWER> follows.

After the completion of this learning-pattern algorithm various patterns are produced for every question type. The procedure that follows is one that concerns the estimation of the accuracy of each pattern. The first 4 steps until the sentence segmentation of a document are repeated as described above in the following algorithm. The only difference is that the documentation retrieval is being conducted using only the question term of the example representing a specific question type and the sentences retained are the ones containing the question term.

After the execution of these steps the algorithm can be described in the following way. Using the results of the pattern-learning algorithm and the resulted sentences of this one, a matching procedure is being conducted. This procedure concerns the observation of two

cases. The first one is when a pattern appears in the sentence with the <ANSWER> tag replaced by any word and the other is when the pattern matches a sentence but the latter tag is replaced by the correct answer. Then the accuracy of each pattern is estimated using the equation (2.1).

$$P = \frac{C_a}{C_o} \tag{2.1}$$

Where:

C_a Corresponds to the number of patterns with the answer term present

C_o Corresponds to the number of patterns with the answer term replaced by any word

The next step is to keep only the patterns that match an effectual predefined number of examples. The 2 algorithms that have been described are applied in different examples of the same question type.

Having completed the work of generating the answer patterns follows the answer extraction for new questions. Initially the question type and term are determined. Documents related to the formulated query are retrieved and submitted to the sentence breaker and further procedure as mentioned above. The replacement of the question term by the appropriate question tag in the resulted sentences follows and the matching procedure of the sentences with the patterns provided for the specific type almost completes the answer extraction. One thing left to be done is to rank the resulted answers according to the accuracy of their matching patterns.

Of course the evaluation of this system revealed some deficiencies like the difficulty of managing long distance dependencies. Another weakness that has been detected is that although reaching to an answer through the correct pattern, the answer may be incorrect and this originates from the fact that the system lacks the use of a speech tagger or ontology.

2.3.2 Q&A system based in knowledge annotation and knowledge mining techniques

The implementation that is about to be presented in this section constitutes a Q&A system that reaches the desirable results through the combination of two different applications (8).

The first one represented by the technique of knowledge annotation handles the mass of questions phrased in natural language and are encountered frequently. These questions can be easily classified to certain question categories. The second application based upon the techniques of knowledge mining is used for the remaining questions that cannot be placed among the common categories.

This alternative way of a Q&A implementation that employs the knowledge annotation techniques takes advantage of the structured or semi structured knowledge existing in the web. In order to be able to access this kind of information repository this method is based upon a number of access schemata. These access schemata consist of pairs of a group of regular expressions that represent a question type and a properly formulated query. This query includes the information resource where the answer to be searched might lies, a variable representing the object in the posed question and the general characterization of the question's answer (e.g. birth date). The latter's value is the answer in the user's question. So this component in responsible for matching the user's question to one of the regular expressions and then execute the produced query. The query execution in order to find the correct answer involves hand crafted applications such as the wrapping of the data sources. These applications vary according to the information resource from which the data was extracted.

The second approach to the specific Q&A system relies on the data redundancy provided from the enormous amount of information in the web. It is time to present the knowledge mining as a method that deals with the attempt to relate a standard question to the several ways that its answer can be expressed. This the central focus of this technique and introduces a way of coping with the extend expressiveness of the natural language. Of course natural language processing such as semantic or syntactic is available to be applied, but instead of these the knowledge mining technique uses the simple pattern matching based on the fact that among the mass of information on the web the answer to be sought will appear as a rephrase of the question. The description that follows enumerates the different operations that need to be executed from the component of a Q&A system that involves knowledge mining.

1. The first concern is to produce the query to be executed in the section that involves information retrieval. The novelty here is that instead of formulating only a query based on pattern matching rules, this operation moves forwards on producing another query that is not as specific as the first one. The first query holds the information of the exact location of the answer and it is characterized as exact query, while the second one called inexact indicates

that the answered to be searched possibly occurs in the vicinity of a set of keywords. In this step two inexact and one exact query are produced and ranked.

2. Secondary follows the execution of the above generated queries using a search engine. In both of the cases the retrieved documents are submitted to further processing to ensure that the correct textual fragments are retrieved.

3. The next step consists of the generation of N-grams from the passages produced from the previous step. These N-grams are weighted according to the query that triggered their retrieval.

4. A new rank is assigned now to the resulting N-grams, which is estimated from the sum of the ranks of the appearances of each N-gram. This affects the documents of low quality in a negative way in order not to be considered credible while the frequently occurred textual segments are boosted.

5. A filtering process follows and some of the above answer candidates are discarded. Such candidates begin or end with stop words or include words appearing in the user's question except the question focus words. Also some heuristics and a set of fixed – list filters are applied in order to reduce the candidate answers when the expected answer is for example a language or a nationality or includes the word 'meters'.

6. The ranking process continues by adding the rank of a short answer to the rank of a longer one if the latter includes the former.

7. Completing the ranking process each answer candidate is multiplied by the following factor, equation (2.2), which balances the effect of individual keywords having different priors:

$$\frac{1}{|A|}\sum_{\omega \in A} \log\left(\frac{N}{\omega_c}\right) \tag{2.2}$$

Where:

A is the set of keywords in the candidate answer

N is the total number of words in our corpus

ω_c is the number of occurrences of word ω in the corpus

8. The last test that remains to be done is whether the resulting candidate answers appear in the initially retrieved documents from the web.

Given the results of both previous applications a number of heuristics are further applied in the answer candidates. These heuristics concern answers that deal with locations or dates and are based on large lists of them.

The main issue that remains the central problem of the system's part that employs the knowledge annotation technique is the manual labor. This labor concerns the wrapping of data sources and the production of the query database. The deficiency detected in the use of knowledge mining technique is that although the hard efforts there still several questions that cannot be answered correctly lacking the deeper understanding of the natural language.

2.3.3 Probabilistic methods applied in a Q&A system

This approach (9) of implementing a Q&A system that is about to be presented here includes stages that were introduced from the already mentioned implementations. The procedures though differ from the previous in most of these stages and this is the reason of examining them more closely. The stage of query modulation that would be expected to be the initial process is omitted in this approach as it is unnecessary in the stage of document retrieval that is going to follow later.

The first stage involves the recognition of the question type. To manage the desirable result questions being under examination were submitted through two different applications, the Riper a machine learning tool and a heuristic rule based algorithm. In order for the questions to be submitted to the Riper a further natural language processing is done. This extra processing resulted in the representation of each question of 13 features most of them consisting the semantic ones that are helpful for the question type identification executed by the Ripper.

The heuristic algorithm that is applied here uses a Part-Of-Speech tagger in order to tag questions. In addition this algorithm is trying to locate base nouns and informative nouns which can indicate the type of certain questions. A hand crafted lexicon is used to index the nouns in the appropriate question types.

After concluding to the question type the next step is document retrieval. In this stage the questions are submitted in three major search engines without any modulation. The 40 top ranked documents that are retrieved are segmented into sentences.

Sentence and phrase ranking are the subsequent stages. The sentence ranking helps improving the computational complexity of the phrase ranking operation. The sentence ranking is achieved with the use of an N-gram model and the ordinary Vector Space model.

After the completion of the sentence ranking follows the phrase ranking. This operation is conducted with the use of a chunker in order to achieve the segmentation of the retrieved documents into phrases. The ranking algorithm functions in such a way that results in attributing high scores in phrases which include many of the query terms or in phrases which appear in the vicinity of a large number of query words.

Another technique applied for phrase ranking is the one that relates the question to the expected answer and it is introduced by the name 'phrase signatures'. The score that is estimated from the latter technique is then multiplied to the previously estimated score and the resulting score of the phrases that constitute the candidate answers is produced.

2.3.4 A Chatbot intermediate between user and a Q&A system

Through the course of studding the various implementations of a Q&A system, it becomes even more apparent to the reader that a Q&A system serves only the needs of the pure question answering dialogue deviating from the natural conversation that can be conducted among humans. These constraining conditions of a dialogue in which someone can be engaged with a Q&A system can be altered by the intervention of a chatterbot. A chatterbot as previously mentioned can manage the looseness of a conversation that comes close to the human nature.

The induction of a chatterbot in the implementation of a Q&A system derives from certain deficiencies which were observed in the dialogic performance of the system (10). The incompetency of the system to recognize so much questions of multiple subject, as questions that can be the succession of previous ones are the substantial reasons of this specific approach.

The entire implementation is modeled upon a dialogue scenario consisted of specific potential moves that can be made by both participants in the conversation. The moves can be omitted but the scenario can be summarized. This scenario involves stages that correspond to the conduction of a realistic conversation. Anytime in the course of this scenario the user can ask for the system to become more conceivable by making some clarifications. Initially the participants greet each other, or the human user starts the conversation by addressing a direct

question to the system. The system now attempts to identify whether the posed question from the other side user is relevant to previously answered questions. If it is confirmed that the question is irrelevant then it is submitted to the Q&A system. On the contrary, if the relevance to the previous question is detected, then the system proceeds in the identification of this relation. Two forms of question relation can be recognized and according to the properties of this relation then the question is characterized either elliptic or anaphoric. Depending on the form of the relation, the user's question is treated differently. When the question is found to be elliptic which means that might not contain a verb, the system reformulates the user's question with keywords extracted from the antecedent questions and then submits the produced question to the Q&A system. In the case where the question is anaphoric, in other words when the question includes references to previous questions, the system proceeds again in the reformulation of the question. This time though instead of submitting the reformulated question to the Q&A component, the system verifies if the produced question corresponds to the initial information that the user is seeking. If this is verified then the system is permitted to process the question through the Q&A component otherwise the user is requested to rephrase his initial question until the new one is expressed in the appropriate way. The next step is for the system to provide the answer to the user and show eagerness to answer another question according to the user's will. If the user wants to carry on then he poses a new question, else the conversation ends up with the system's acknowledgement. The completion of the conversation is accomplished with the participant's greetings.

The chatterbot's responsibilities are focused on the part of identifying the elliptic or anaphoric questions and invoke the appropriate resolution components. For that reason some new AIML tags are added in order to support operations like those mentioned above or to invoke the Q&A component.

After having examined the dialogue model on which the architecture of the system is based, it is time to explore the structural techniques that have been used. It hasn't been mentioned yet that the system's incompetency of answering questions of multiple matters is partially resolved. In order to detect this kind of questions a chunker is used and with the condescension of the user the system answers on the part of the question where the most tokens were marked.

The system supports the resolution of elliptic questions, those which contain third person pronoun/possessive adjective anaphora, or questions which contain noun phrase anaphora.

The latter two constitute the anaphoric questions. So in order to be able to manage the occurrence of such questions initially a detection algorithm is applied.

The detection algorithm follows a number of actions that are invoked depending on the detected type of the question. If the question isn't verified to belong to any of the above types then the processing is transferred to the field of Q& A component's responsibilities. In the case that the question is elliptic, then the keywords of previous questions are used to fill the textual gaps of this one. There is also the occasion where the question is anaphoric and the systems performance varies according to the specific type of the anaphoric question. When the pronoun adjective anaphora is traced the first noun phrase compatible antecedent question is used. In the case of noun phrase anaphora the first noun phrase that corresponds to all the references in the question is used to replace them. Of course it could not be neglected the occasion where no previous question serving the demanded constraints can be found. I this particular case the system itself request the user to explicitly rephrase the question.

2.4 Summary

The readers of this chapter were introduced to another type of conversational agents, the Q&A systems. The common architecture of such systems was presented in order to put the reader in the way of comprehending the demands of building them. Nevertheless specific implementations of certain Q&A systems were examined in such a depth revealing the variety of the architectural techniques that can be applied on them. More specifically the readers was introduced to the method of extracting answer types through learning surface patterns, which is one of the most significant stages in a Q&A implementation. Moreover knowledge annotation and knowledge mining entered the contsractual fields of Q&A systems and suggested another interesting approach in the issue. Eventually probabilistic methods appeared in this section demonstrating the power that they offer when implemented in Q&A systems. Through the presentation of the above the differences between these systems and chatterbots are being noticeable. These operational engines serve different purposes and in order to signalize the fact the one complete the services that the other offers, an application based on the collaboration of both of them was presented. The conclusion is that the limitations of Q&A systems can be compensated by the interference of chatterbots in their implementations.

3 Artificial Intelligence Markup Language (AIML)

3.1 Introduction

The contents of the section that is about to be presented, aim to describe the most significant elements that constitute the artificial intelligence markup language, as well as their operational behavior. This language facilitates those who attempt to implement a chatterbot by providing them the means to import knowledge to the application being under construction. Although AIML was introduced by ALICE's implementation, its potentialities are still growing (11). Nevertheless in the present section the focus is on fundamental elements of AIML with some worth notice exceptions. It must be underlined that AIML is an XML variant and this is a justifiable observation to be made until the completion of this section.

AIML consists of several tags that are responsible of converting the response in order to accomplish many operations. These operations enable the application to store required information, to stimulate other programs if it is necessary and to return to the user responses under certain conditions or through the recursive use of pattern matching. The convenience which AIML pattern language offers, derives from the fact that permits only the use of words, numbers, spaces and the wild card symbols _ and *. The following are the most important data units and some of them are being presented in further detail below:

- <**aiml**> : this is the tag that marks the beginning and the end of an AIML document
- <**category**> : this is the tag that labels the basic units of knowledge in the application's knowledge base
- **<pattern**> : this tag encloses a simple pattern which attempts to match the user's input
- <**template**> : this is the tag that contains the appropriate response to the user's input

3.2 Category and recursion

Category is the most fundamental element of AIML and is composed from additional elements. The general structure of category includes a properly formulated query, the response corresponding to the specific query and further context that its existence is optional according to the requirements that are handled in each occasion. The above mentioned query

is represented by the pattern tag and is the responsible unit for matching the user's input. The corresponding reply is represented by the formerly mentioned tag of template. The additional context that may be encounter in the category's composition consists by the tags <that> and <topic>. The <that> tag consists of a pattern, that intends to represent the last utterance of the chatterbot and of course match it. While the latter tag is possible to be encountered only inside the category, the <topic> tag appears only outside of it. This tag is responsible of congregating several categories which deal with user's input concerning the same subject. The value assignment to the <topic> is performed inside the template. There are three category types represented below.

- Atomic
- Default
- recursive

Atomic categories are the simpler approach in category's formulation. The pattern description does not involve the use of wild card symbols. A representative sample of the specific occasion follows.

<category>

<pattern> WHAT IS A CIRCLE </pattern>

<template> <set_it> A circle </ set_it> is the set of points equidistant from a common point called the center.

</template>

</category>

This category matches the user's input "What is a circle?" with the pattern that includes. The value "A circle" is assigned to the "IT" variable. The returned answer to the user is the one enclosed in the template tags.

Default categories are another form frequently encountered and their substantial difference with the atomic categories is the presence of at least a wild card character in their pattern. Wild card characters are capable of substituting any other word in the pattern. The frequently encountered default categories include a pattern that involves a few words and a wild card symbol. An enlightening example of such categories follows.

<category>

<pattern> I NEED HELP * </pattern>

<template> Can you ask for help in the form of a question? </template>

</category>

The above category is capable of treating a group of user's inputs that consist of the phrase "I need help" and continue with any other sequence of words represented in this occasion by the star symbol. The use of that specific category type is a deceptive method that has as a result to make the user believe that the chatterbot does not lack reasoning. This result derives from the user's impression of conducting a coherent conversation with the chatterbot.

The last category type is known as the recursive type. Recursion is a significant quality of AIML because of the many offering facilities. This kind of categories define the route to be followed from the moment the user types the input, through the categories that congregate the knowledge map. The produced output of a category is used as the input of another and this procedure has as a result the formation of a category chain which ends up producing the final response. Recursion can be used in the above applications or the combination of them and the responsible tag is <**srai**>:

1. Symbolic reduction is the application where the AIML user attempts to convert complex grammatical forms to simpler ones. The common strategy followed is to create categories as simple as the atomic ones. When a query is represented in a more complex form from the user and the atomic category that represents this query in the simplest form exists in the chatterbot's knowledge base and then this category can be reached through recursion and produce the correct output.

<category>

<pattern> DO YOU KNOW WHO * IS </pattern>

<template> <srai> WHO IS <star/> </srai> </template>

</category>

In the above example the more complicated input "Do you know who * is?" is reduced to the simpler one "Who is *?" and the initial word being replaced by the wild card symbol is represented in the reply by the tag <star/>. The final response will be produced from the existing atomic category whose pattern will match the user's initial input.

2. Divide and conquer is another application that recursion is used in. This method is responsible for the segmentation of the user's input into many parts which afterwards are

applied separately as inputs in the appropriate categories and eventually the desirable output is produced by the responses returned from each part.

3. The resolution of synonyms is the most common application of the recursion. This derives from the fact there are many words in natural language that can describe one and only thing and unfortunately only one pattern per category is permitted. Thus the recursion provides the ability of accessing the appropriate category consisting a simple pattern through other categories whose templates include this specific pattern while their patterns are synonyms of this.

4. Spelling and grammar correction is an additional application of recursion. This handles the common mistakes that are possible to appear in the typed input. Besides of correcting them this application also produces the corresponding response. A representative example follows.

<category>

<pattern> YOUR A * </pattern>

<template> <srai> YOU ARE A <star/> </srai> </template>

</category>

5. The use of <srai> contributes also in the detection of keywords in the user's input that can be used as leverages to activate specific categories. The following example illustrates the format of such an application.

<category>

<pattern> MOTHER_ </pattern>

<template> <srai>MOTHER </srai> </template>

</category >

In the above example the keyword is detected when it appears as a prefix in the user's input. Similar format have the categories that indentify keywords in the place of prefix or infix in a sentence and afterwards attempt to reach the category with the desirable template which will produce the final answer.

3.3 Significant tags and elements

It has been mentioned previously that in a category format there is a possibility of using the tags <that> and <topic>. These tags are very important for the reasons mentioned above and

in an attempt to comprehend their operational behavior an example of their usage is presented here.

<category>

<pattern> THAT IS GREAT </pattern>

<that> Today I am happy. </that>

<template> But will I still be happy tomorrow? </template>

</category >

In the above category the pattern matches the user's input "That is great" only if the precedent utterance of the chatterbot is the one enclosed in the <that> tag. It is obvious that in the knowledge base it is possible to encounter categories that have the same pattern but differ in the part of the tag <that>. On the other side it is explained that <topic> is responsible for grouping certain categories together. The reason of referring to these tags together is that the later depends on the existence of the former. A category that belongs to a specific topic will be retrieved only if exists in the knowledge base another category with the same pattern and the same <that> tag.

Other significant elements of AIML are the conditional ones. The conditional elements are used in order to retrieve responses given certain criteria. At this point the predicates of AIML are going to be introduced. These predicates represent variables that are used to store useful information from the user's input. The tag <set> is the one responsible for this storage procedure and its appearance is permitted in the tag <template> or in other special tags. So in conditional elements the output depends on the matching process of a predicate against a given pattern. There are three types of conditional elements the block condition, single-predicate condition and the multi predicate condition. These types are described in further detail below.

The block condition is the type of conditional element that involves an AIML predicate declared by the use of an attribute name e.g. "occupation" and a corresponding attribute value that consists of a simple pattern expression such as "engineer". During the predicates matching process if the attribute value agrees with the value that has been assigned to the predicate with the specified attribute name, then the contents of the condition are returned otherwise the empty string is returned. The single-predicate condition is the type of conditional element that involves an AIML predicate but instead of the attribute value used in the above condition type involves some other AIML element the li elements. These li

elements are a type of variables with predefined values. The procedure followed here begins with comparing the value of the predicated specified the name attribute to the value of the first li element. If these values match then the contents of the li item are returned and the process stops. Otherwise the matching process continues until the match process succeeds. There are three types of li elements though the reference to them would be redundant. The multi-predicate condition is a variation of the latter and their basic difference is that each of them uses different types of li elements.

3.4 Summary

This section was an introduction to the fundamentals elements of AIML. The important tags that consist AIML were examined closer in order to comprehend their operational behavior and the distinguish occasions that each one of them handles. During the examination of the category element and the recursion operation which can be conducted through the <srai> tag, the convenience that AIML offers in the procedure of knowledge loading in a chatterbot was made obvious in addition to the capability of handling many complex cases. The conditional elements came to contribute to the previous conclusion by verifying that despite the simple technique of pattern matching AIML provides the means to apply even more demanding techniques. Of course the potentialities of this language are not limited in the ones having mentioned. There are still more AIML elements that were wittingly omitted in order to provide to the reader a more global illustration of AIML avoiding entering in heavy details.

4 Our Architecture

4.1 Introduction

In the previous sections the focus was on two different types of conversational agents, chatterbots and Q&A systems. Thus far it has become obvious that chatterbots provide the means for the user's engaging in a natural language conversation conducted in ostensibly realistic terms. On the other side, Q&A systems serve the needs of producing responses to the user's questions relying upon predefined question and answer types and the techniques of information retrieval and extraction. A few representative implementations of the former mentioned systems were studied in the previous chapters in order to predispose the reader for the main issue of the present thesis and provide him the adequate knowledge to comprehend the work that have been conducted here.

The initial goal of the present thesis was to accomplish the construction of a chatterbot based upon the techniques applied in the implementation of ALICE bot. The extended reference in AIML elements and their operational details intended to accommodate the reader in the following presentation of the specific thesis, since AIML is the partially responsible tool for the knowledge that have been loaded in this chatterbot. The intention of this work was to achieve an implementation of a system which can engage the user in a conversation conducted in natural language and based on the communicative method of text typing. Besides the common characteristics of an open domain dialogue the attempts in this thesis focused mainly on increasing the chatterbot's initiative towards the human user's initiative. Unlike the common existing chatterbots the one implemented in the present thesis attempts through the course of each dialogue to take responsibility for changing under certain circumstances the subject in issue and provoke the user to participate in the new subject.

Additionally the focus of the present thesis was on partially inserting in the chatterbots construction qualities that describe Q&A systems and involve specifically information retrieval. Besides the knowledge base of the system that is constructed using AIML, under certain circumstances responses returned to the user are extracted from a base of documents concerning biographies of significant personalities after a special procedure of the documents and the user's input.

4.2 PyAIML

In the present work the implementation of the chatterbot in issue was not initiated from "scratch". On the contrary an already existing interpreter for AIML written in Python was used to handle AIML categories that already existed and moreover which were added in the course of this thesis. Of course in the contents of this interpreter, the PyAIML interpreter, were made many alterations and additions in order to accomplish the desirable result and reach the goals of this thesis. The initial components of this interpreter were six classes written in python and their responsibilities are going to be described below.

The first class to be presented and its written in python is the AimlParser. This class as it can be assumed by its name is responsible for parsing the AIML documents that exist in the chatterbots knowledge base. This class is capable of parsing AIML documents and checking for their validity according to the structure that is followed by them. For example some substantial inspection to be made by this component is whether the tags of <pattern>, <template> and <that> are inside the category. In general this component is responsible for the preservation of the regulations that AIML imposes. Beside the previously mentioned operation, the specific component executes additional fundamental operation. This operation is responsible for the production of a substantial dictionary which is a basic and commonly used structure in python. Such a structure written in python consists of two essential elements, the key and the value. In this occasion the key each time is the contents of the category that was parsed and as it was demonstrated earlier these contents consist of the information enclosed in the tags <pattern>,<that> and <topic>. The value in each case is assigned to the contents inside the tag <template>. So the conclusion to be extracted from here is that the product of the specific component is the contents of the AIML knowledge base in a useful formation which facilitates the following procedures.

The next class written in python and constitutes another substantial component of the specific AIML interpreter is the PatternMng. The word PatternMng is abbreviation of the words pattern manager. This component is responsible for many tasks but two of them are the most significant ones. The first task is the transformation of the knowledge base in a structure that can be approached easily in order to retrieve the correct enclosed information. This task aims to give the knowledge base a tree structure similar to the one that exists in the common linguistic dictionaries or encyclopedias. The resulted structure is a graph that consists of a collection of nodes and branches that connect them to each other. From the root of this graph

starts the mapping of the branches that each one of them represents the first word of every parsed pattern. The branches are either single words or wild cards. The number of the leaf nodes in the collection is the same as the number of the categories in the AIML knowledge base. The leaf nodes also contain the contents of the <template> tag of each category. So initially the pattern is separated into its consisted words and its one of them enters the tree node represented by a branch. The next insertion to be made in this tree structure is the contents of the <that> tag if they are non empty. Afterwards the contents of the <topic> tag are included in the tree node if they are non empty and the last one is the insertion of the <template>.

The above procedure of the construction of the knowledge tree is strictly defined by the way the matching of the input is executed. The resulted tree is constrained to be in this formation because of the stages followed in the matching process. When the user's input is given then a standard matching process is followed. If there is a given word "X" and the root of the graph is also know then this process consists of the following stages:

1. The navigation through the tree nodes initiates from the root. The first key that is expected to be found is the wild card "_". When the specific key is found then the process continues by searching the subgraph that has in the place of the root the wild card "_". The remaining suffixes of the input that follows "X" are checked for matching and if there is no match found then the process proceeds to the next stage.
2. The next thing to be sought here is the word "X". If the search succeeds then the subgraph initiating with the child node "X" is searched. The next expected matching is the one of the tail of the input, in other words the user's input with the word "X" removed. If this matching process fails then the next stage is invoked.
3. This stage is responsible for the finding of the wild card "*". If this key is found then the search proceeds in the subgraph rooted at the child node linked by "*". In the specific subgraph any of the remaining suffixes of the input following "X" is expected to be found. If no match is found then the procedure starts from the top.

The terminal case here is the one where the input is left with no more words and the leaf node contains the matched template. Of course if the procedure fails there is an easy way to obviate such a failure. This failure is confronted by the existence of the wild card "*" pointing to a leaf node.

Of course it must be mentioned that the entire matching process involves also the contents of <that> and <topic> in order to result in the correct response. In the previous section was made a reference in the way that the tags <that> and <topic> contribute in the matching process. If the formerly mentioned matching stages succeed then the contents of the two latter tags are being searched for matching in the tree structure to result to the corresponding template.

Through the examination of the matching process someone would notice that in every node the wild card "_" has the major priority and then follows the word matching with the wild card "*" owning the lowest priority. An additional observation is that the matching process is conducted word by word and no category by category. Nevertheless there is no need for the patterns to be alphabetically ordered but only partially ordered so that the wild card "_" comes before any word and the wild card "*" comes after any word in the tree structure. An example of the way that the knowledge tree is build piece by piece is presented below. A demonstration of the insertion of three categories in the tree structure follows.

<category>

<pattern> I HAVE A PROJECT </pattern>

<template>

<think> <set name="topic"> PROJECT </set>

On which course?

</template>

</category >

<category>

<pattern> ON NLP </pattern>

<template> Good luck! </template>

</category >

<topic name="PROJECT">

<category>

<pattern> ON NLP </pattern>

<template> Is it difficult? </template>

</category >

</topic>

When a category does not contain the tag <that> or does not belong to a specific topic suggested from the tag <topic> then the keys of the dictionary that refer to this parts of the category to be parsed are assigned to the symbol "*". Through the parsing of the above categories the corresponding dictionary is produced with the appropriate keys and values. The next stage that is responsible for constructing the knowledge tree according to the keys and values of the dictionary. The resulting structure is presented below using the next codification:

- "_" = 0
- "*" = 1
- <template> = 2
- <that> = 3
- <topic> = 4

After the insertion of the first category the tree has the following image:

Knowledge Tree

{

{u'I':{u'HAVE':{u'A':{u'PROJECT':{3:{1:{4:{1:{2:{categories[value]}}}}}}}}}}

}

After the second category's insertion the image of the knowledge tree is the below:

Knowledge Tree

{

{u'I':{u'HAVE':{u'A':{u'PROJECT':{3:{1:{4:{1:{2:{categories[value]}}}}}}}}}},{u'ON'

:{u'NLP':{3:{1:{4:{1:{2:{categories[value]}}}}}}}}

}

And eventually after the last category's insertion to the tree, the later has the following image:

Knowledge Tree:

{

{u’I’:{u’HAVE’:{u’A’:{u’PROJECT’:{3:{1:{4:{1:{2:{categories[value]}}}}}}}}}},{u’ON’
:{u’NLP’:{3:{1:{4:{1:{2:{categories[value]}}}},{u’PROJECT’:{2:
{categories[value]}}}}}}}}
}

After having explicitly described the operation of the two most important components of the AIML interpreter it is time proceed in the presentation of the less significant ones. The WordSub class is the component that handles the word substitution when is needed. The most common word substitution is for example if the user’s input is “I’m not …” in order to be able the interpreter to process the sentence must transform it to a new one having replaced the truncated phrase “I’m” with the integral one “I am”. In addition to this kind of substitution this component also handles the substitutions of personal pronouns. The class DefaultSubs is the repository of these substitutions and it is used from the WordSub in order for this task to be achieved. In other words the DefaultSubs consists of dictionaries that refer to the several kinds of word substitution and have as the key the word to be replaced and the corresponding value is the substitute word.

The class Utils written in python is the component that is responsible for segmenting the user’s input in sentences. The segmentation of the input into distinguished sentences is a simple procedure that relies on the search of the input for punctuation. When a full stop, a question or an exclamation point is detected in the user’s input this component keeps the already parsed elements of the input and stores them in a list considering them elements of an individual sentence. The implementation of this process derives from the fact that the specific chatterbot has the ability of processing one sentence at a time.

The last but not least component of the interpreter is the one that handles most of the operations and additionally is responsible for invoking the formerly mentioned components. This component is the class Kernel of this implementation and it is the one in which several interventions were made in order to achieve the goals of this thesis. The operations that this component handles initiate from processing the user’s input and with the contribution of the remaining components produces the appropriate response to the user expressed in correct formed natural language.

4.3 Detection of previously discussed topic

In the general goals of this thesis was included the increment of chatterbot's initiative. So working under the aspect of the initiative increment an additional function was added in the operations of the chatterbot. This function enables the chatterbot to detect whenever the user returns in a previously discussed subject. Nevertheless this function succeeds its purpose under certain conditions. These conditions will be revealed in the following paragraphs as the corresponding work will be explained in detail.

Initially it must be underlined that the chatterbot is programmed to store certain information of every dialogue session in which each individual user participates. The sessions are individualized by a specific identification number. The way that this identification number is reproduced will be analyzed in a following section. The bottom line here is that from every session some information is retained in order to achieve some required checks. The component that is responsible for the stored information is the class of Kernel.

Before intervening in the interpreter's components the information that could retain the chatterbot during the course of a conversation with a user was the input history and the output history. The input history represents the user's utterances. This history is stored in a form of list and every insertion concerns the user's input segmented in individual sentences. So when the user's input consists of three sentences then three will be the insertions on the input history list. The output history respectively is a form of list where the responses that the chatterbot produces and returns to the other side user are stored.

The intervention that was made in the kernel's component concerns the retaining of further information that can contribute in the detection of a previously discussed subject. Another list called TopicHistory, where the discussed subjects are stored, is added. It has been mentioned earlier that given the user's input a matching process follows across the knowledge tree in order the template containing the correct response to be extracted. The returned template is in the form of a dictionary containing besides the correct answer some other AIML tags which serve various operations. One of these tags that is possible to be encountered in the returned template is the tag <set name = "X">. This specific tag assigns a value to the predicate X. The value is represented from the words that follow the predicate X in the structure of the category from which the template derives. When the name of the predicate is the "topic" then the value refers to the specific subject that the user's recently posed utterance introduces. After retrieving the correct template additional functions in the Kernel component are responsible

for handling the various tags that are included in it. The function which is responsible for processing the tag <set name = ”X”> exists in the Kernel’s component and is called processSet. In this process when the name of the predicate is “topic” then the value of this predicate represents the current subject of the conversation. The last element of the TopicHistory list, which represents the last subject of the present conversation, is retrieved and is compared to the current discussion’s subject. If the two elements set into comparison are proved to be the same then the insertion of the topic’s value in the TopicHistory is omitted, otherwise is executed. If the current subject of the conversation is different from the last one then follows the search of the TopicHistory list for the existence of this specific topic’s value. If the search succeeds this means that the user brought up again a previously discussed subject and the value of another predicate called “oldTopic” is assigned to TRUE.

In the next stage where the process of the template has been completed and the correct response is ready to be returned to the user, the value of the predicate “oldTopic” is checked whether is TRUE or FALSE. If the value is proved to be TRUE then the fetched response is not returned to the user until through the same matching process another response is fetched. This latter response constitutes of a sentence which reminds the user that the new introduced subject of the discussion it is actually a previous discussed one. The formerly fetched response that corresponds to the user’s input is concatenated to this one and the flow of the conversation keeps on.

Besides the interventions that were made in the Kernel’s component in order to achieve the detection of the previously discussed subject the AIML knowledge also has been extended. Certain categories were added in the knowledge base which included the tag <set name = “topic”>. In addition to these categories on more was created to serve the needs of the needs of verifying this detection and it is presented below. In order to be retrieved the template of this category the input is provided from inside the Kernel’s component and is the same with the category’s pattern.

```
<category>
<pattern>OLDTOPIC</pattern>
<template>
<random>
<li>I see we ended up talking about <get name="topic"/> again.</li>
<li>Here we go with <get name="topic"/> again. </li>
```

So we are back in <get name="topic"/>.

</random>

</template>

</category>

In the beginning of this subsection a reference was made to certain circumstances under which this application succeeds its performance. These circumstances are described above by the extensions of the AIML knowledge base. In order to achieve the best performance of this application there should be a careful selection of the categories to be added in the knowledge base which is a rather demanding manual labor. In spite though of these obstacles the specific application combined to a properly formed AIML knowledge base can accomplish its initial aims.

4.4 Detection of max inactive input

Continuing working under the aspect of increasing the chatterbot's initiative the application to be presented in this section was accomplished. Most of the popular existing implementations of a chatterbot are constructed in such a way that only the user can hold the initiative of engaging in a conversation. In an attempt to augment the chatterbot's initiative in this application the focus was on the detection of the max inactive input from the side of the user. The method applied in this occasion did not interfere with any of the existing components of the initial implementation of the AIML interpreter or with the AIML knowledge base.

For the needs of this purpose a specific server-client application written in python was created. Whenever the user wants to connect with the server the program that concerns the client should be activated. The connection with the server is succeeded through an individual socket which is different for every user. The knowledge base of the chatterbot is not constructed from the start for every individual user. Instead of this an object of the class Kernel is created for each one user. In this way a significant space saving is achieved. In a previous subsection an identification number that individualizes every session being conducted between the user and the chatterbot was mentioned. In the server's application this number is reproduced randomly among 65.000 thousands possible numbers and as a result every time a user attempts to connect with the server obtains his identification number that is different from any other used number. When the client's application receives any data then

the data are sent to the server's application. The time that the server's application realizes the existence of the data in the socket's input then the corresponding method of the Kernel class is called in order to handle the data. This method is called respond. In the client's application a timer is activated when the user connects to the server. If the user's input remains inactive for a period of time equal to the predefined timeout of the timer then the client's application sends a signal in the server's application. The signal is stored in the data that are sent to the server. When the server recognizes the signal then sends to the client's output a response that encourages the user to continue the conversation.

The returned response in the particular occasion is not retrieved from the AIML knowledge base but derives from the server's application. As result the specific response is not stored in the output history. A slight defect of this implementation is that the input is activated when the user have completed the typing of the data. In this way when the predefined time is over and the user is in the middle of the typing task the encouraging response will be send to him. That is the most frustrating reason for fetching the encouraging response from the server's application instead of retrieving it from the AIML base. Nevertheless the chatterbot's initiative is incremented due to this small contribution.

4.5 Detection of the user's loss of interest

In the present section another application will be presented which is conducted in the terms of this thesis. This section concerns two different parts, the process followed in order to detect the user's loss of interest and the process followed in order to resolve the problem. In this part of the specific thesis the other side of the present implemented chatterbot will be revealed. This side is the one that can place this implementation among those which attempt to combine the application of a chatterbot with a Q&A system. Of course in this implementation only some borrowed methods that are applied in the general framework of a Q&A system constitutes the elements which can lead to the conclusion that this is an application based on the combination of a Q&A system and a chatterbot. Through the course of this section the presentation of further details will be enlightening.

4.5.1 Process of detection

In order to detect the user's loss of interest a counter of the letters which the user's input consists of is set in the Kernel's component. When the user's input applied then the counter is

activated and the resulted number is stored in a list called ChaCountList. The counting of the characters is applied in the entire user's input without any consideration of the number of sentences that the input consists of. If this number is lower than fifteen then the last five elements of the list are retrieved and it is checked whether each of them is equal to fifteen. If the check succeeds then the ChaCountList is evacuated from the existing elements and the user's current input segmented in sentences is inserted in another list called beforeChangeTopic. It must be mentioned that the matching process has not been applied yet to the user's input. In this point instead of fetching the corresponding reply to the user's input the given input to be searched and matched is the one that will result in the chatterbot's asking the user if he would like to discuss something else besides the present subject. This response is fetched through the matching process that is applied to the input "CANGETOPIC". The input is given from inside the Kernel and the returned response is retrieved due to the existence of the below category in the chatterbot's knowledge base.

<category>

<pattern> CHANGETOPIC </pattern>

<template>

<think> <set name = "errChangeTopic"> TRUE </set> </think>

Would you like to discuss something else with me? </template>

</category>

The chatterbot in this point accepts only a "yes" or "no" answer. If the user's answer is negative then the initial input of the user is extracted from the beforeChangeTopic and is applied to the matching process in order to continue the previous conversation. On the other side if the user agrees to discuss something else then another process is followed in order for the chatterbot to find the appropriate subject and suggest it to the user. This part of the application is explicitly analyzed in the following subsection.

4.5.2 Resolution of the user's loss of interest

In order to resolve the user's loss of interest and in the case that the user willingly agreed to change the conversation's subject another process is followed. This process aims to find a subject for discussion and suggest it to the user. Instead of picking randomly a subject this application relies on the information that can be retrieved from the history of the dialogue.

Initially it must be mentioned that besides the AIML knowledge base which exists in the specific implementation, there is also another knowledge base that consists of a collection of biographies that concern famous people such as actors and actresses, athletes, authors, scientists, philosophers, painters and singers or even music bands. This collection of biographies was downloaded and the corresponding HTML documents were submitted in a cleaning process through which the HTML tags were discarded and the resulted documents contain only plain informative text ready to be processed.

In order to find the subject to be suggested to the user another component written in python was used, the Trigger.py. This component contains a dictionary that consists of keys and their corresponding values. The values consist of a list of index terms extracted manually from each biographical document while the keys that the values belong to represent the category that the documents from which the index terms where extracted belong to. There are seven categories in the existing data base and are presented below.

- Cinema
- Sports
- Literature
- Music
- Painting
- Philosophy
- Science

In this component is also stored a dictionary that consists of dates and the corresponding significant historical events that have happened from time to time. The dates are the keys and the events are represented by the keys' values. When the user agrees to change the subject of the conversation then a part of the input history is submitted to the Trigger component. This part consists of these elements of the input history that have been inserted in the list after the last detection of the user's loss of interest. The process to be followed searches these elements in order to find existing terms that correspond to the stored index terms. If no term is found in the input history then the dictionary in the form of a journal is searched and its keys are checked whether they match to the present date. If this matching process succeeds the corresponding value to the key is returned. In this case the returned response to the user is in the form of a question asking if the user was aware of the retrieved fact. If the matching process through the journal fails then the returned response will concern a bon mot, in other

word a witticism, fetched randomly through the corresponding AIML category that is presented below. The input, in order to be retrieved one of the several witticisms, is submitted to the matching process inside the Kernel's component.

<category>

<pattern> BON MOT </pattern>

<template>

<random>

 Having been at someone's else's mercy suggests that mercy may matter .

 The pain was an extension of experience, so new and astonishing as to have intellectual interest.

 One half of the world cannot understand the pleasures of the other.

 It is personalities, not principles that move the age.

 Progress is the realization of utopia.

 In this world, nothing is certain but death and taxes.

 The art of prophecy is very difficult, especially with respect to the future.

</random>

</template>

</category>

In the case that some index terms are identified in the input history their corresponding category that represents also the topic that they belong to is stored in a list. These index terms are also stored in a dictionary and constitute the values of the keys that are represented by the found subjects. In another dictionary are stored the sets of the default terms that belong to the subjects that have been found. The Trigger process returns those three lists. In order to suggest a subject for discussion to the user the component of the Kernel follows another procedure.

When the Trigger component has completed its operation and the resulted lists and dictionaries are returned then the first element of the list, containing the candidate subjects, is checked in order to verify that this subject was not the one suggested when the previous loss of interest was detected. This procedure is conducted by searching another list of subjects existing in the Kernel's component. The later list stores the subject that is suggested each time to the user. If a repetition of the subject is detected then recursively the next subject to be

checked is the next in the returned list from the Trigger component. If all the elements of the list are checked without any success then the returned response to the user is again a bon mot. In the contrary if the specific process succeeds to find the appropriate subject the user is asked whether he is interested in it or not. If the reply of the user is negative then the response of the chatterbots is retrieved either from the journal list or from the category that contains the various witticisms. In the case that the user expresses his eagerness to discuss the suggested subject then the list of the found index terms is checked to see if there are any specific index terms besides the general ones that describe this subject. If this list is empty then the list containing the default index terms is retrieved and it is asked from the user to select one of them.

The next stage is to submit either the selected default index term or the elements of the input history in another component responsible for finding the most similar document to the user's selected utterances. This component estimates the similarity between the user's utterances and the documents that belong in the suggested subject. The weights of the words in every document were estimated using the equation for the Tf-Idf weighting (12). The weights of the words of every document are stored in a dictionary in this specific component. With the help of this dictionary the similarity is estimated in this occasion using the equation that represents the cosine similarity (13).

$$w_{i,j} = \frac{freq_{i,j}}{n_{w_j}} \cdot \log \frac{N}{n_i} \tag{4.1}$$

Where:

$w_{i,j}$ is the weight of the i word in the j document

$freq_{i,j}$ is the frequency of the word in the j document

n_w is the number of all the words in the specific document

N is the total number of the documents to be compared

n_i is the number of the documents in which the i word appears

$$Sim(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{\left|\vec{q}\right|\left|\vec{d}\right|} = \frac{\sum_{i=1}^{M} w_{iq} w_{id}}{\sqrt{\sum_{i=1}^{M} w_{iq}^{\ 2} \cdot \sum_{i=1}^{M} w_{id}^{\ 2}}} \quad (4.2)$$

Where:

$w_{i,q}$ is the weight of the i word in the user's query which in the specific occasion is the input history

$w_{i,d}$ is the weight of the i word in the d document

M is the total number of the words in the d documents

After estimating the cosine similarity between the documents of a subject and the user's input history, the top ranked document according to this similarity is now available. Of course it is not retrieved the entire document. An additional work was preceded for the present case where the most similar document to the user's input history is available.

For this case in the AIML knowledge base were added two more special categories. These categories are formed in such a way that they are capable of giving the user the basic biographical information about the person that the retrieved document from the estimation of the cosine similarity concerns. This basic biographical information is the full name, the date of birth and the place of birth. If the document concerns a music band instead of a person then the corresponding category informs the user about the name, date of the formation of the band and the place of action of the band. These specific categories retrieve the biographical information from some AIML predicates that are assigned their values in the Kernel's component. For their values assignment the process of the extraction of this information has been preceded and the biographical information that was extracted from each document was stored in text documents. Thus when the most similar document is available the document which contains the basic biographical information is also available and inside the Kernel the corresponding predicates obtain their correct values and the appropriate input is submitted to the matching process in order to be fetched the template from the corresponding category.

The time that the biographical information is retrieved and returned as response to the user the chatterbot also asks the user whether he would be interested in further information on the same subject. The same subject corresponds to the person whose the biographical elements were extracted. If the user responds expressing his eagerness the second phase of this process

initiates. In other text documents the summary of each document is stored after being extracted from the initial documents. So when the user accepts to be further informed on the specific person or band the returned response consists of the summary of the corresponding biographical document.

4.5.3 Index term's extraction

As it was previously mentioned the extraction of the index terms from each document was conducted manually. In order to prove that this process was more efficient than applying a process that would produce automatically the index terms, a second process of index term extraction was applied in the documents. The C/NC value method was used in the latter process (14). This method relies on the combination of linguistic and statistical information. The process begins by estimating initially the C-value of each word in the corpus in order to use these values for the estimation of the NC-value.

The C-value method consists in its linguistic part of three components. The first component is the part-of-speech tagger that identifies grammatically each word of the corpus attributing a relative tag. The second component is a linguistic filter which takes under consideration the grammatical tags of each word and identifies possible patterns as terms. Last component that is used in the C-value method is a stop-list that consists of words which have minimum possibility to represent index terms. The statistical part of the C-value method estimates the possibility of a candidate string to represent an index term and this part takes under consideration statistical properties of each candidate string. The statistical properties consist of the frequency that the candidate string is encountered in the corpus and the corresponding frequency that this string is a part of other longer candidate strings. The number of these longer candidate strings and the number of the words that constitute the string are included also in the statistical properties.

The NC-value method can be described in three stages. The initial stage is the one where the corpus of the document is submitted to the process of the C-value estimation and results in the production of a list filled with candidate terms and their corresponding C-value scores. The second stage of this method concerns the extraction of a list of words relative to the document that undergoes the entire process weighted according to the number of candidate terms near which they appear in the corpus. This stage uses an initial list of terms related to each document in order to find the most important words, which are called context words, in

the vicinity of them. The weights of the extracted context words are estimated according to the following equation:

$$weight(w) = \frac{T(w)}{n} \tag{4.3}$$

Where:

w	is the word that is found in the vicinity of the candidate term
$weight(w)$	is the weight assigned to the word w
$T(w)$	is number of the terms that the word appears with
n	is the total number of terms

In an attempt not to deviate from the automatic extraction of the index terms this list consists of the top ranked candidate terms extracted from the C-value method. The third and last stage of the CN-value estimation involves firstly the estimation of another factor. Every candidate term's factor is estimated by multiplying each weight of the context word lying in the vicinity of the specific candidate term with the frequency of the occurrence of both of the terms in the corpus. The last estimation is the one of the CN-value of each candidate term, which derives from the next equation:

$$NC - value(a) = 0.8 * C - value(a) + 0.2 * CF(a) \tag{4.4}$$

Where:

a	is the candidate term
$C - value(a)$	is the C-value of the candidate term a
$CF(a)$	is the factor estimated for the candidate term a

The application of the C/NC-value method in the documents that are used in this work produced a list of candidate index terms consisting of one word and ranked with their NC value. Because of the fact, that the number of the manually extracted index terms were maximum 7 per document, the precision and the recall between this two lists of terms was estimated among the manually extracted index terms of each document and the group of the

20 first terms extracted automatically segmented in bunches of 5 terms. The following figure shows the results of this estimation. The precision and the recall estimated by the equations (4.5) and (4.6).

$$Precision = \frac{\left|\{relevant\ index\ terms\} \cap \{retrieved\ index\ terms\}\right|}{\left|\{retrieved\ index\ terms\}\right|} \tag{4.5}$$

$$Recall = \frac{\left|\{relevant\ index\ terms\} \cap \{retrieved\ index\ terms\}\right|}{\left|\{relevant\ index\ terms\}\right|} \tag{4.6}$$

The relevant index terms are considered to be the terms which are extracted manually while the retrieved index terms are those extracted automatically with the use of C/NC-value method. The precision for the 5 top ranked retrieved index terms and the relevant index terms was estimated per document and the average precision for all the documents was estimated afterwards. This procedure continued in order to estimate the average precisions for each group of 5 terms among the 20 top ranked. Hence, the following results to be presented in the next table are estimated:

Table 4.1: Precision - Recall

Retrieved index terms	**0-5**	**6-10**	**11-15**	**16-20**
Precision	0.225	0.047	0.017	0.020
Recall	0.226	0.051	0.018	0.027

The conclusions that can be extracted are that the C/NC-value method fails to extract the right index terms and this is attributed to the fact that this method relies more on statistical features of the corpus. In general the frequency of a term is taken under significant consideration while in the process of the manual extraction of the index terms semantics features consider to be more important than the frequency of a term. In other words the manually extracted terms are semantically related in high degree to the person or the music band that the biographical document is presenting.

5 Evaluation

5.1 Introduction

An inevitable and significant concluding stage of this thesis was the objective evaluation in which the implemented system was submitted. In cases where the knowledge base of a chatterbot system is full of information, the chatterbot in issue is open to be engaged in conversations which concern every topic that the user suggests. When such a system is submitted in an objective evaluation a major part of the concentration is placed to whether the dialogue between the user and the system qualifies or not coherence. On the contrary in the evaluation process being presented here the focus deviated a lot from this target. In the specific occasion the AIML knowledge base is not sufficient to support a big variety of subjects that the user might suggested, so some boundaries on the subject's domain had to be set. The insufficiency of the knowledge base is attributed to the time consuming process of information storage. The AIML knowledge base of the specific implementation is limited and the user's who have participated in the conduction of the evaluation were informed about the topics that the chatterbot would be able to support. Despite this fact the produced conversations in a few segments lack coherence and as a result of this, coherence was not the most significant criteria of this evaluation process. Instead of coherence though the significant evaluation criteria was whether the system under evaluation meets with success the detection of user's loss of interest.

5.2 Description of the evaluation process

The specific evaluation process included 6 users. Each user after being informed on the topics which the system can support was asked to engage in a conversation with two distinguished chatterbot systems. The first system called system 1 represented the initial system that the specific implementation is based on. The second system with which the user's engaged in a conversation was the system being under issue in this thesis. The users during the entire evaluation process were totally unaware of the identity of both of the systems submitted in this process. Of course both of the systems contain the same stored information in their AIML knowledge base. On the other side their operational attributes obtained by each AIML interpreter's implementation are the ones that lead each system in expressing an

individual performance. The participants after being engaged in a conversation with both of the systems were asked to fill an evaluation form. In this form the users, relying in their personal experience with each one of the systems, evaluated them by applying scores concerning different aspects of each system's performance. The scores ranged from 1 to 10, where 1 represented a very poor performance of the system, 5 represented an average performance while 10 represent an excellent one. Initially the participants were involved in an overall system's evaluation, secondary in evaluating the system's success of detecting the time in which the user loses his interest and finally they were asked to evaluate the coherence during the course of the dialogue. Attributing scores of the same range, the users showed their beliefs on whether the system worked the way they expected and additionally they showed in which degree they would be eager to re-interact with the same system. Eventually under the previously defined terms this evaluation process was completed and produced the following results that are presented in the next subsection.

5.3 Results

In the overall system's 1 evaluation the estimated average score is 7 while in the corresponding evaluation of the system 2 the resulting average score is approximately 8.3. Evaluating the performance of the system in the detection of user's loss of interest the score corresponding to the system 1 is estimated approximately 4.2 and the corresponding score for the system 2 is approximately 8.8. During the evaluation of the coherence that the dialogue appeared, the average score for the system 1 is 7.2 while the score for the system 1 is 7.3. In an attempt of measuring the satisfaction of the participants deriving from the interaction with each one of the systems, the estimated score concerning the accomplishment of the user's expectations from the system 1 is approximately 6.7 while for the system 2 the score is approximately 8.2. The participants' eagerness to use the system again is expressed for the system 1 through the score of 7.5 and for the system 2 through the score of 8.3.

This evaluation process proves that despite the restricted domain of topics that the specific chatterbot's knowledge base is able to support, the initial goals of this work have been achieved in a considerable degree. The detection of a previously discussed topic was achieved as long as the stored categories in the AIML base, from which the answer were extracted, conserved the terms that were discussed in the relative subsection. The detection of loss of interest of the user had also satisfying results. Whenever the process of the detection

succeeded the suggested subject for discussion was indeed related to the preceded user's utterances.

5.4 ANOVA Analysis

The average value by itself is not sufficient to describe a data set, as it doesn't provide information about the variation in the data. ANalysis Of VAriance (ANOVA) is an alternative technique used to determine if differences between two or more data sets are statistically significant. ANOVA technique involves six steps which are described below:

- Calculation of sum of squares (SSs) due to each source
- Calculation of degrees of freedom (dof), associated with each source
- Calculation of mean squares $(MS = SS/dof)$
- Calculation of the F ratio (F)
- Determination of the critical F ratio $(F_{critical})$
- Determination of significance, $p-value$

It must be mentioned that the only requirement for ANOVA is that there are sets of data for at least two different "treatments".

In this thesis in order to estmate how significant are the results of the evaluation process of the two systems, the Matlab's function $p = anova1(X)$ is used, where X is a matrix and p is the return value of the function. Anova1 function performs balanced one-way analysis for variance, for comparing the mean of two or more columns of data in matrix X. Each column represents an independent sample with independent observations. Returned value, p, shows the level of significance that the samples represent. If the $p-value$ is near zero, it means that at least on sample mean is significant different than the other sample means. Commonly encountered significance levels are 0.01 or 0.05. The very small $p-value$ indicates that differences between columns means are highly significant. The matrix which is used in order to evaluate how statistically significant the systems are, is presented below.

$$X = \begin{bmatrix} 6 & 9 \\ 6 & 7 \\ 8 & 9 \\ 8 & 9 \\ 7 & 9 \\ 7 & 8 \end{bmatrix}$$

The first column represent the users' scores for the system1 and the second column the scores for the system2. Anova fuction displays two figures, the standard ANOVA table and box plots of the columns of X , which suggests the size of F and the $p-value$. The figures are presented below.

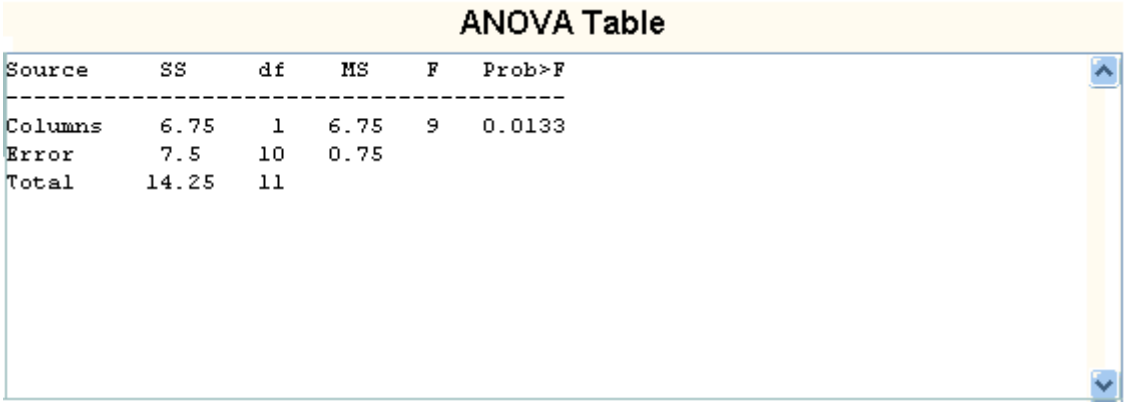

Figure 5.1: ANOVA table

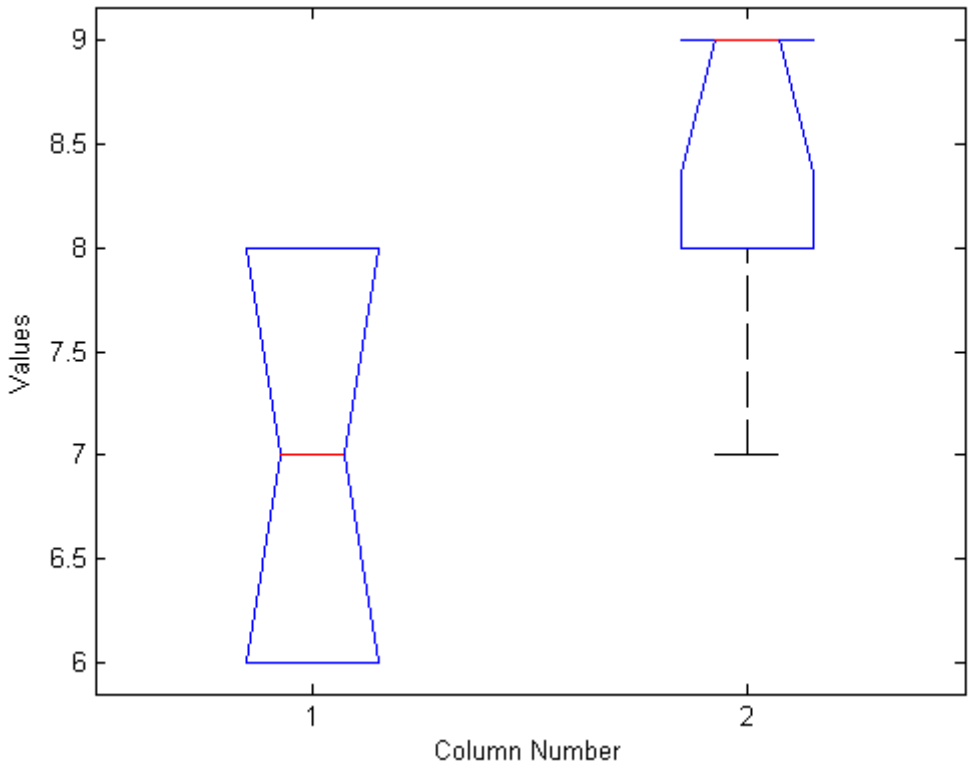

Figure 5.2: ANOVA box plots

Observing the resulted score that is assigned to the $p-value$ through the ANOVA analysis it is reasonable to consider that the results of the preceded evaluation process are statistically significant.

6 Conclusions – Future work

During the course of this thesis and having examined closely the framework and the various implementations of chatterbot's system and involving partially with the philosophy that encloses the implementation of Q&A systems some interesting extensions on an already existing chatterbot system have been achieved. These extensions derived from an initial central goal which was to increase the dialogue initiative of the chatterbot against the initiative of the human. Working under this major aspect the resulted system acquires now some interesting attributes.

At first a time counter was set upon an implementation of a server client application. This counter is activated each time that the user inserts the input utterance into the system and counts down to a certain period. When the counting, called timeout, finishes and the user has not posed yet the next utterance then a response is sent to the user trying to encourage him to continue the dialogue. This application though lacks the connectivity with the preceding or the following dialogue because of the fact that the chatterbot's utterance is retrieved from the level of the server's application. The obstacles of retrieving this encouraging message from the AIML base were that this message would be send to the user over and over again even if the user missed to reply and the retained information of the dialogue history would be confused. Another significant obstacle was that the time counter was not possible to be implemented in the application of the Kernel's class. Nevertheless a modest attempt was made.

The next intention of this thesis was to make the system capable of detecting the time that the user reenters a previously discussed topic. The initial implementation was able only to detect the current discussed topic and identify the change of it. For this work the responsible was only the AIML base and the categories stored in it. In the present implementation the responsibility of detecting a previously discussed topic share both the AIML base and the AIML interpreter. Having filled the AIML knowledge base with the correct categories the interpreter acquired this interesting attribute.

The important part of this work is the last one that concerns the ability of the chatterbot to detect a certain state of the user and try to cope with this state. This certain state represents the user's loss of interest during the course of the conducted dialogue. When this certain state is

detected the user is asked to verify this detection. If this detection doesn't stands the dialogue flows according to the previous input of the user. On the contrary I if the user is eager to change the subject of the discussion, the chatterbot is capable of suggesting one. The process that is responsible of finding this subject relies on an information retrieval technique known as the cosine similarity. The details have been presented in a previous chapter. The performance of this application is more that satisfying as the results of the evaluation process can confirm this fact.

The future work that can be suggested in order to improve the system's performance derives mostly of the encountered obstacles while working in this thesis. The great dependency of the user's output from the AIML knowledge base was rather a restricted factor for this implementation. The pattern matching technique that was followed set the need of a fully updated base inevitable in order to succeed coherence on the dialogue. So a well updated knowledge base having an easy administrating structure is required. This would take off the demanding work of filling the base. Despite the efficiency of the knowledge base, some semantic and pragmatic features should be added in the implementation, in order to improve the coherence of the dialogue. The retrieved answers are impossible to correspond always to the user's input correctly and as a result of that a semantically process of the data would be considerable. Another future extension it would be to extract from the documents index terms that would consist of more than one word and extend the corresponding application in order to be able to handle the user's input by separating it in groups of 2 or more subsequent words. This suggestion implies the use of N-grams generation techniques.

7 Bibliography

1. **Weizenbaum, Joseph.** *Eliza -A Computer Program for the Study of Matural Language Communication Between Man and Machine.* 1966.

2. **Kenneth Mark Colby, Franklin Dehnis Hilf,Sylvia Weber, Helena C. Kraemer.** *A Resemblance Test for the Validation of a Computer Simulation of Paranoid Processes.* 1971.

3. **Kenneth Mark Colby, Roger C. Parkinson, Bill Faught.** *Pattern-Matching Rules for the Recognition of Natural Language Dialogue Expressions.* 1974.

4. **Patrick, Crews.** *Protochat: An Exploration of Chatbot Construction.*

5. **R., Wallace.** *The Elements of AIML Style.* 2003.

6. **A.M., Turing.** *Computing machinery and intelligence.* 1950.

7. **Hovy, Deepak Ravichandran and Eduard.** *Learning Surface Text Patterns for a Question Answering System.* 2002.

8. **Katz, Jimmy Lin and Boris.** *Question Answering fron the Web using Knowledge Annotation and Knowledge Mining Techniques.* 2003.

9. **Dragomir Radev, Weiguo, Hong Qi, Haris Wu, Amardeep Grewal.** *Probabilistic Question Answering on the Web.*

10. **Manandhar, Silvia Quarteroni and Suresh.** *A Chatbot-based Interactive Question Answering System.* 2007.

11. **R., Wallace.** *http://www.alicebot.org/documentation/matching.html.* 2001.

12. **Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze.** *Introduction to Information Retrieval.* 2008.

13. **Schutze, Christopher D. Manning and Hinrich.** *Foundations of Statistical Natural Language Processing.* 1999.

14. **Frantzi, K. & Ananiadou, S.** *The C-value / NC-value domain independent method for multi-word term extraction.* 1999.