An Approach to Enhance Chatbot Semantic Power and Maintainability: Experiences within the FRASI Project

Agnese Augello, Giovanni Pilato, Alberto Machi'
ICAR - Istituto di Calcolo e Reti ad Alte Prestazioni
CNR - Consiglio Nazionale delle Ricerche
Viale delle Scienze - Edificio 11 - 90128 Palermo, Italy
e-mail: {augello, pilato, machi}@pa.icar.cnr.it

Salvatore Gaglio
Dipartimento Ing. Chimica, Gestionale, Informatica e Meccanica
Università di Palermo
Viale delle Scienze, Edificio 6 - 90128, Palermo - ITALY
e-mail: gaglio@unipa.it

***Abstract*—The paper illustrates the implementation and semantic enhancement of a domain-oriented Question-Answering system based on a pattern-matching chatbot technology, developed within an industrial project, named FRASI. The main difficulty in building a KB for a chatbot is to handwrite all possible question-answer pairs that constitute the KB. The proposed approach simplifies the chatbot realization thanks to two solutions. The first one uses an ontology, which is exploited in a twofold manner: to construct dynamic answers as a result of an inference process about the domain, and to automatically populate, off-line, the chatbot KB with sentences that can be derived from the ontology, describing properties and relations between concepts involved in the dialogue. The second one is to preprocess user sentences, and to reduce them to a simpler structure that can be referred to existing elements of the chatbot KB. The enhanced symbolic reduction of user questions and the automatic population of question templates in the chatbot KB from domain ontology have been implemented as two computational services (external modules).**

I. Introduction

Question Answering (QA) systems can be thought as information retrieval systems which try to answer to natural language queries by returning answers rather than a simple list of document links. Question Answering (QA) systems make use of natural language techniques to select the most appropriate answers by using linguistic features [1]. They differ mainly for the knowledge sources, the broadness of domain expertise, the kind of information to get, and the kind of results to give [2].

On the other hand Natural Language Dialog Systems (NLDs) are an appropriate and easy way to access information. Users can type natural language questions and expect to receive short answers in natural language. Usually, in its simplest form, this approach does not require sophisticated natural language processing or logical inference.

In last years Chatbots have played a prominent role as human-computer interfaces [3] [4] [5]. Chatbots are generally composed of three modules: the user interface, an interpreter, and a knowledge base. One of the most versatile chatbots is the Artificial Linguistic Internet Chat Entity (Alice) system [6]. Alice has a knowledge base constituted by a very large number of simple stimulus-response rules named categories. In [7] authors have pointed out that Alice is used more like a search engine rather than a conversational tool. Alice can therefore be used as a fact-driven chatbot suitable for giving domain-specific information in a pleasant manner. In [8] a chatbot has been built by firstly converting an ontology into a relational database as a basis for the construction of the NLD Knowledge Base. The transformation of ontologies to relational databases is made by using a set of rules that describe the manner in which constructs on the ontology to the relational model.

In [9] an open domain QA system, named YourQA, which uses a chatbot interface has been illustrated. It takes the top 20 Google results for a user question, retrieves the corresponding Web pages and analyzes them to extract answers and rank them by relevance to the question. Building a chatbot knowledge base requires a remarkable effort; in order to overcome this limitation many approaches have been proposed. For example, in [10] a way to access information using a chatbot, without the need for sophisticated natural language processing or logical inference, is illustrated.

Authors have proposed machine learning techniques to develop Alice language models starting from a training corpus of dialogue transcripts. The result is the transformation of the corpus into a set of categories or pattern-template pairs. The illustrated approach should be sufficient if the conversation between the chatbot and the user are limited to a very narrow and specific topic. Besides, in [10] Abu Shawar shows that sophisticated NLP is not better than word-based pattern matching when answering questions from restricted-domain FAQ.

In [11] Semantic Web knowledge has been used in order to improve the capabilities of a language independent conversational agent oriented to solve question answering tasks. In [12] authors introduced a chain of conversational agents and subdivide questions into domains: definition, measure, list, comparison, factual and reasoning questions. In particular, the first agent has the task of converting Semantic Web knowledge to AIML format, according to the approach proposed by Freese [13], while a second agent deals with the task to detect the domain of the input and consequently build an answer by querying an OWL ontology with Protege API. The agent uses also a lexical database, used to search for synonyms, in order to analyze a wider range of inputs. Besides, before any of the conversational agents use the ontology, a preprocessing phase is used to expand the ontology using data from a lexical database and Wikipedia, in order to enrich the answers of the chatbot.

The work presented in this paper was proposed within the industrial research project FRASI (Framework for Agent-based Semantic-aware Interoperability), to develop a question answering system with the aim of supporting consumers to get useful information regarding products of interest. The choice was to implement a Question-Answering (QA) system using a pattern-matching chatbot technology. This choice represents a simple solution to build a friendly dialogue interface. Among the technologies available today we have decided to use the Alice (Artificial Linguistic Internet Computer Entity) [6] implementation, which is simple and based on an open source technology. The strongest limitation of Alice derives mainly from the pattern matching algorithm used by its engine for the dialogue management, and the rigidity of its knowledge base, based on the definition of specific, unmodifiable rules, organized as question-answer pairs. This architectural setting implies the definition of all possible interactions with the user during the design phase, leading to a cumbersome Knowledge Base (KB) and a time-consuming development process.

For these reasons, as a solution, we propose a streamlining of the chatbot knowledge base, leading to an improvement of the sentence analysis process. In particular we propose the use of three modules. The first one, called Enhanced Symbolic Reduction (E-SRAI) module, preprocesses user sentences, and reduces them to a simpler structure. At present this is a preliminary work, and the E-SRAI module recognizes only a simple set of sentences, however the approach is scalable. A second module, named bootstrapping module, makes it possible to exploit ontologies in order to automatically extend the chabot KB: this makes a further simplification of the KB development process. On the other hand, domain ontologies, or other knowledge repositories, can be used to automatically build in background a set of question answers pairs for the chatbot in order to quickly create a basic and scalable dialogue system KB, capable to answer simple and common questions specialized for a specific topic. Besides, a third module makes it possible to use ontologies in order to have a dynamic generation of answers, obtained by means of an inference process about the user questions.

II. Architecture

The conversational agent architecture, (fig. 1), shows how the standard Alice technology can be enhanced by the introduction of new modules with the proposed solution. The dialogue engine communicates with two main modules, called E-SRAI and CYD through the matching of special AIML categories which contain *ah-hoc* defined tags. The former analyzes each user sentence: the aim is to recognize a specific kind of sentence and consequently reduce it to a simpler structure. The latter extends the chatbot KB with the knowledge stored into a common sense ontology.

These modules are executed at run time: every time the user introduces a new question, it is first analyzed by the E-SRAI module, and the answer is obtained by executing the rules defined in the chatbot KB, which can trigger the invocation of the CYD module to query the ontology.

Finally, the architecture is composed of another module, which is run offline and called AIML Bootsrap, which is aimed to an automatic population of the chatbot KB, through a process which queries an ontology and fill in a set of predefined question/answer schemas. This feature allows to easily and quickly adapt the chatbot to different application domains.

A. Domain Ontology for the case of study

The architecture has been introduced in order to realize a user friendly QA system on a very specific domain: the milk weaving factory. We have collected documents about the domain from the world wide web and other documents provided by the industrial partner. We have grouped entities described in the retrieved documents in production processes and objects with properties. It is possible to have a description of the process by giving the parameters that influence its execution. A product is transformed by some agents, has parts or components, it may have a brand, etc. We have formalized knowledge about the domain in CYC [14] by introducing the main concepts (such as the main kinds of milk, the manufacturers, ingredients, processes, brands, ...) definitions associated to concepts, the main predicates in addiction to the well known *genls* and *isa*. In the following we show a sample of useful predicates:

```
hasPart/partOf:
 {Product} hasPart {Product | Component}

produce/producedBy:
  {Agent} produces {Product}
  hasParameter/parameterOf:
  {Process} hasParameter {Parameter}
```

and some examples of assertions:

```
  (#$genls #$PartiallySkimmedMilk #$Milk)#$MilkMt
  (#$comment #$PartiallySkimmedMilk
   "Partially Skimmed Milk contains less fat ...")#$MilkMt
  (#$hasPart #$PartiallySkimmedMilk #$Lactose) #$MilkMt
  (#$producedOf #$Zymil #$Parmalat) #$MilkMt
```

B. Dialogue Engine

The chatbot KB is composed of question/answers pairs, called categories, structured according to the AIML (Artificial Intelligence Markup Language) language [15] within pattern/template tags, which define the rules for its lexical management understanding. In particular each user question is compared with the KB patterns and the template associated to the matched pattern is therefore examined. If the template contains other tags, the execution of other modules is accomplished in order to generate a more coherent, answer, otherwise

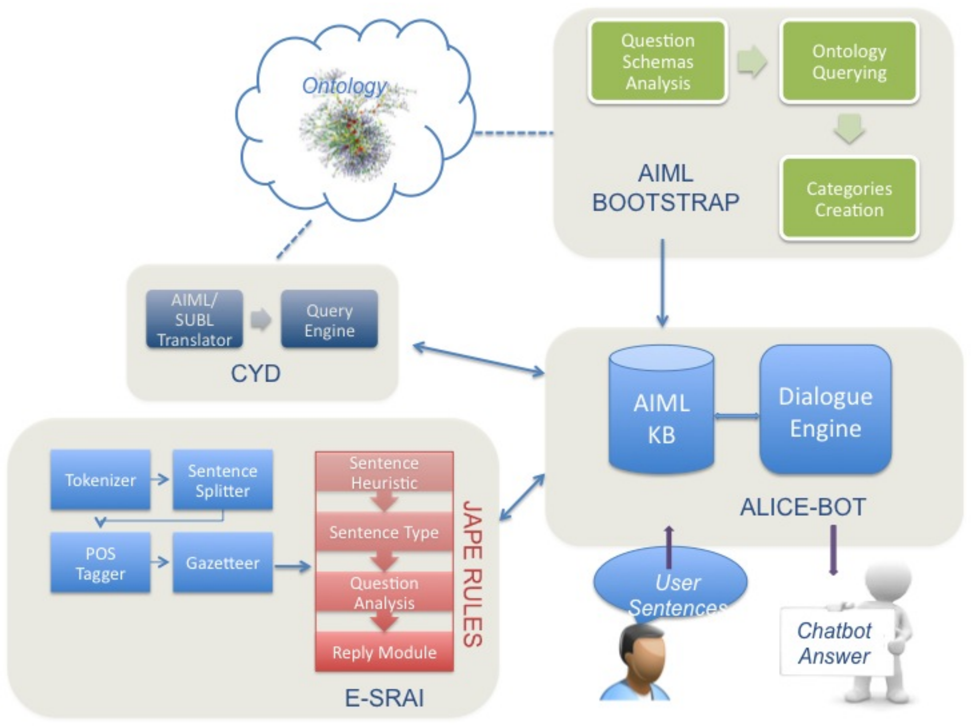

Fig. 1. Alice architecture enchanted by the introduction of new modules

the answer will be the plain text contained in it.
The dialogue engine is based on a pattern matching algorithm that at runtime searches for a match between the user question and the chatbot KB.
The main drawbacks of this approach are: the time-expensive designing process of the AIML Knowledge Base (KB), because it is necessary to consider the possible user requests, and the dialogue management mechanism, which is too rigid. The problems due to the limitative matching rules are overcome by the use of the E-SRAI and CYD modules. The template can contain specific tags, to trigger the execution of these two modules obtaining useful advantages.

In the standard Alice architecture the botmaster has to create a great number of predefined question/answer couples (atomic categories) to create a convincing chatbot, with a great effort both to write and to anticipate possible user questions. He can also prefigure only a partial structure of the sentences that will be typed by the user. In fact, inserting special symbols, called wildcard (* and _), in the pattern he can write more generic categories (default categories) to make a partial or total match with the user question. The botmaster has to manually write more specific categories, or write more generic categories but generating vague answers. The connection with these modules change the envision of the AIML KB and consequently the work of chatbot KB building which results greatly simplified. In figure 2 we can see the AIML KB as a pyramid, where it is possible to identify four main types of categories:

- *atomic*: specific categories which are defined to anticipate a predetermined set of questions and respective answers; these categories can be manually created by the botmaster, or automatically built by querying the ontology (by means of the AIML Bootstrap module);
- *default*: categories that have wildcards inside their pattern, and standard AIML tags in the template;
- *ontology based*: categories that can have specific tags introduced to communicate with the ontology. The pattern can contain wildcards which can match with ontology concepts.
- *ultimate default*: a category containing only a wildcard in its pattern, it matches with any user input; the template executes the module E-SRAI to perform the processing of the query, and uses its result to call again the pattern matching.

In the proposed solution, to create more specific categories the botmaster can take advantages from the AIML Bootstrap module and moreover to implement a smart dialogue system, he can design of a small number of default and ontology based categories, containing simple questions schemas. The user questions will be analyzed by the E-SRAI module, which tries to reduce them to one of these schemas. It can return the schema and other useful informations; as an example it can understand that the user searched for a definition for a specific item. The results of this module will be analyzed by the chatbot, which will search for a match between the schema and its patterns, and, according to the template rules, will query the ontology in order to generate the more specific answer for the user question. As a result, the botmaster can create only few schemas of categories, which are matched and analyzed in order to obtain a correct answer. This implies an automatic intrinsic generation of more atomic categories. Moreover in this case, the answers obtained by more generic categories, such as the ontology based category, are coherent and specific and not vagues.

C. Enhanced Symbolic Reduction Module

The effectiveness of the Alice chatbot has been enhanced by performing a natural language process, which we named called E-SRAI (Enhanced Symbolic Reduction), of user queries. The process aims to the reduction of lexical variability of AIML patterns, through the recognition of meaningful semantic schemas in user questions and to their mapping into standard patterns that are easily manageable by the AIML engine. This module allows to perform a more efficient analysis of the user sentences, instead of simply search for a matching within the chatbot categories, and, moreover, it allows to reduce the writing process of the chatbot's knowledge base. The task of the module is to understand the kind of question written by the user and to extract particular variables by using syntactic, semantic or pragmatics rules. At present we have a simple italian parser that recognizes and analyzes italian questions. We have used the tools provided by OpenGate [16] in order to define analysis rules. Because of the complexity of the identification and recognition of the semantic language schemas, the definition of the linguistic analysis module requires the restriction of the case study to a limited knowledge domain. In this project, knowledge domain is restricted to the description of products and processes of milk product weaving factory. Similarly, the language patterns to be managed are restricted to very specific structures of sentences, such as direct and indirect simply structured questions. The modular architecture of the question answering system and a straight-chain implementation of the NLP module (lexical, logical and syntactical analysis of the sentences) helps in making the solution scalable with respect to the recognition of more complex semantic schemas as the ones involving cause/effect relationships between entities recognized as being significant for the domain and described in the knowledge base.

1) Classes of sentences in the analyzed domain: We make a preliminary analysis of a corpus of documents containing common questions about the knowledge domain (FAQs or Frequently Asked Questions) to get a classification of sentences to be recognized and properly answered or to be refused by the automatic QA. In the analyzed domain we suppose that the user may ask questions related to specific issues such as production, food quality and security, nutrition or curiosities. In the production area, we expect the user will submit questions regarding the ingredients in products, their origin, where the product is transformed, information about any treatments it undergoes. The categories of meaning formal, constitutive and agentive are involved in such questions. In the food quality / safety area the user will formulate questions related to the quality of a product, expressed in terms of information on safety and food hygiene, information on complying to the regulations on traceability of foods, the meaning of the norms, or their importance in order to protect health of the consumer. We recognize in such questions the categories of meaning formal, constitutive and the expression of situations. In the class of FAQs with a formal sense, we can recognize two structural patterns corresponding to requests for identification of an entity (a product, a process, a manufacturer, etc.) or the difference between the descriptions of two entities). In the class of FAQs with a constitutive sense, we can recognize three patterns corresponding to requests for: listing of parts/ or components of an entity, testing whether an entity is contained or is a component of another one, quantify the strength of the membership, or testing if an entity is participating in a context (situation). In particular our prototype deals, at present, with FAQs having a formal structure, which may be described as:

```
FORMAL/Definition: WHAT+ IS +ENTITY
WHAT = {which|what|who}
IS={about|be|be meaning of|be definition
        of|be sense of|mean|appear}
ENTITY=
{SUBSTANCE}|{PRODUCT}|
{PROCESS}|{TASK}|{AGENT}
```

2) Question analysis: The process of question analysis consists of breaking down a document in sentences , then recognizing the parts of speech (grammatical analysis) and finally the roles terms play into the sentence (logical analysis).

In particular it consists of the following steps: text segmentation in tokens eventually reduced to their root term (lemma), the identification of their role in the construction of the phrase and the recognition of dependencies between the roles attributed in standards patterns (standard pattern of connections between roles in the structures of typical sentences). This corresponds for the Italian language to morphological analysis (from the token to the lemma), to grammatical analysis (part of speech of terms), to logical analysis (recognition of the roles attributed to the terms or to a combination of them), and the period analysis (recognition of sense via the structure of the period).
FAQs are questions aimed at obtaining a description of individual entities, of their composition, of the effects of external agents on them or to the recognition of their practical functions. Significant terms are therefore ones that express requests (interrogative pronouns / adverbs, terms expressing a query or request), verbs that express the essence of an action, nouns (plain or proper names) that express the subject or the object of an action about which the user requires information. The minimization and translation of a question to a skeletal form can be obtained by defining an ad-hoc pipe of natural language processing modules, exploiting the standard processing chain of GATE (General Architecture for Text Engineering) [16], called ANNIE (A Nearly New Information Extraction system) [17]. The pipe of modules (computational resources) grows documents in an input corpus (language resources) with semantic annotations. Each module operates iteratively on each text of the corpus being analyzed. It receives as input the text already annotated by the previous modules,

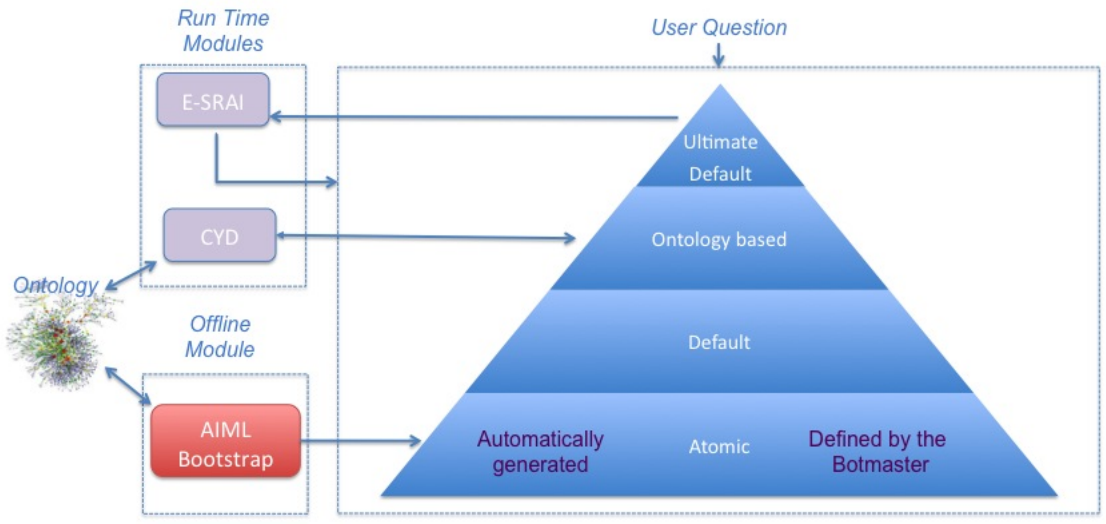

Fig. 2. AIML Knowledge Base in the proposed solution

and copies it back to a new enhanced or modified version. The framework activates each computational modules in a sequence, or under condition on the basis of text annotation contents, according to the value of parameters described in an XML configuration file. Annotations may be grouped in sets and each computational module operates on a subset of current annotations and outputs other annotations.

The framework includes a Java Annotation Patterns Engine Transducer (JAPE Transducer) [18] which translates scripting and Java code segments into a finite state engine able to apply regular expressions on text and annotations. It is a translation system, enabled by the verification of Horn clauses on annotations, expressed in a rule scripting language called JAPE. In JAPE, the antecedent of each clause is a lexical pattern or a logical test on a set or type of annotations; the consequent (result) is a new annotation or the action of a fragment of Java code.

In our case of study the modules of the chain have been included in a stand-alone GATE application embedded into a WebService, and called at request by the AIML engine. The ANNIE Computational modules have been configured and modified by introducing specific rules for the italian language in order to reconfigure the english Stemmer, to increment the generic lexicon available for the italian language with nouns, adjectives and verbs commonly used in the dialogues about milk, to configure the Gazetteer with Gazettes including an item for each listing of the individual described in the knowledge base and labeled with its class of membership. Moreover the JAPE Transducer has been augmented with rule modules in JAPE defining Horn clauses for recognition and annotation of interrogative pronouns and verbs, for the identification and counting of common and proper names (probable subjects/complements or properties), of verbs phrases or actions and for the identification of question sense category.

The Jape Transducer interprets rules contained in the following four modules:

- Sentence Heuristic module: it seeks for the presence of one or more interrogative pronoun, of a question mark, of one or more verbs, and of one or more nouns.
- Sentente Type module: it tags a sentence as “Simple” if it contains just one verb and as “Question” if it contains a question mark and an interrogative pronoun or an equivalent interrogative multiword expression.
- Question Analysis module: it contains three rules which in sequence implement a small decision tree with the purpose of identifying the sentence structure. Simple well defined structures are searched for at first, complex ones later. The first rule tests if the sentence contains exactly one verb tagged as equivalent to verb “to be” (a “copula” according to italian classical logical analysis) and just one noun naming one instance, property or ontology class. Copula presence implies a FORM question, single object complement implies a FORM-DEFINITION sense. Other rules manage incomplete check. More verbs suggest a complex structure (one main and one coordinated or subordinated phrase). More objects require further analysis. Unrecognized name or class indicates an unknown object.
- Reply module: it composes and outputs an answer according to sentence classification. Answers to FORM-DEFINITION questions with recognized object contain a prologue such as “FORM-DEFINITION” followed by a customization the recognized object class (eg. “Process”) and the name of the object. (eg. “pasteurization”). Answer to sentences with unchecked syntax create standard AIML patterns chosen according to the level of structure identification: Definition of an unknown object. “Unmanaged complex question” or assertive sentences create an AIML pattern containing “Not a question” followed by the sentence text. Such standard patterns are usable during dialog to force the user to express the question with a simpler structure or less ambiguous terms.

D. CYD module

The methodology of the dialogue, on which the chatbot Alice is based, is enhanced by the CYD module, which allows to generate responses to user requests through the composition of semantic queries addressed to the common sense ReasearchCYC ontology [14]. The CYD module is a Java wrapper of the CYN technology [19] a project developed at the Daxtron Laboratories with the aim to extend the chatbot knowledge base with the information stored in a wide common sense repository such as Cyc.

The AIML language is extended with specific tags in order to make the chatbot capable to query the Cyc ontology directly from its KB rules. The chatbot AIML template can contain these new "ad hoc" AIML tags which transform it into a "meta-answer" that must be processed by the ResearchCyc inference engine to produce the chatbot most appropriate answer to the user question. The main tags for communication with Cyc are the following [19]:

- *cycterm*: translates a natural language term into a Cyc constant;
- *cycsystem*: executes a CycL query into the Cyc KB. If the Cyc query contains a variable, it returns its corresponding value, otherwise it returns the value *TRUE* or *FALSE*.
- *cycassert* and *cycretract*: asserts or deletes a Cyc formula into or from a Cyc microtheory respectively;
- *cyccondition*: verifies the value of a CycL statement and in consequence give an answer;
- *cycrandom*: executes a CycL statement and return a random answer.

The Cyc answers are embedded in a natural language sentence according to the rules of the template. The integration with Cyc has several advantages among which being we can include the benefit of making categories writing easier: it is possible think of different ways to express general topics creating generic, default categories. The string matching the wildcard symbol belonging to the category pattern will be searched into the Cyc KB; if a corresponding Cyc constant is found, the chatbot can analyze all the information associated to it exploiting the different Cyc predicates.

For example, the chatbot can extract a definition of the constant by means of the *#$comment* predicate, it can verify if the constant is a collection or a collection instance and analyze the other related concepts by means of the *#$isa* or *#$genls* predicates. Or in specific contexts it can analyze "ad hoc" predicates.

It is possible to arrange all these information together, creating dynamically exhaustive answers, not present in a traditional AIML knowledge base. The possibility of exploiting a wide, common-sense ontology, like Cyc, makes the dialogue more fluent and reduces the number of default answers needed in traditional chatbots to fill up their " cultural" gaps.

As an example the following category

```
<category>
 <pattern>
   FORM-DEFINITION PROCESS *
 </pattern>
 <template>
      <cycsystem>(fi-ask
      '(#$comment <cycterm><star/></cycterm> ?X) )
       </cycsystem>.
  </template>
</category>
```

leads to the following conversation:

```
User: Puoi parlarmi della pastorizzazione?
        (Can you talk about pasteurization?)
Chatbot: Processo di risanamento termico applicato
 ad alcuni alimenti allo scopo di minimizzare...
         (Process of  heat treatment applied
         to some foods to minimize ...)
```

E. Aiml Bootstrap

A background process is performed offline in order to automatically build a set of categories for typical domain-oriented questions; this approach greatly simplifies the process of populating the AIML knowledge base. After writing a few questions, related to definitions of domain concepts, to description of processes, to the extrapolation of instances etc., the AIML bootstrap module issues specific queries to the ontology in order to properly fill the answer field in question/answer pair schemas. A set of question schemas are defined in advance, then, according to the question, the ontology is queried in order to create a set of AIML categories, filling the pattern and the template schemas with the concepts or relations derived from the ontology. Figure 3 shows an example process for the AIML categories bootstrapping creation. The process will be repeated each time the ontology is updated in order to align information present in the chatbot's knowledge base and in the ontology.

III. An example of interaction among the modules

An example of interaction between modules during the dialogue with an user is shown in fig. 4 . The figure illustrates an example regarding the fact that the sentence introduced by the user is wrongly recognized as an affirmative sentence. The E-SRAI module, after preprocessing the sentence gives the result to the AIML module, which identifies a matching with one of the AIML patterns, executes the statements written into the template section in order to build the answer and communicates with the ontology in order to save context information acquired during the dialogue.
Another example shows a sentence correctly recognized by the E-SRAI module as a formal question for a definition. The question is translated into a simpler form which is subsequently analyzed by the AIML engine. The pattern matching mechanism finds a match between the simplified question and one of the AIML categories and subsequently queries the ontology in order to generate a proper answer to the user.
Figure 4 shows for each sentence a translation in english language.

IV. Conclusion

An approach to enhance chatbot systems and simplify their implementation has been presented. The work illustrates our

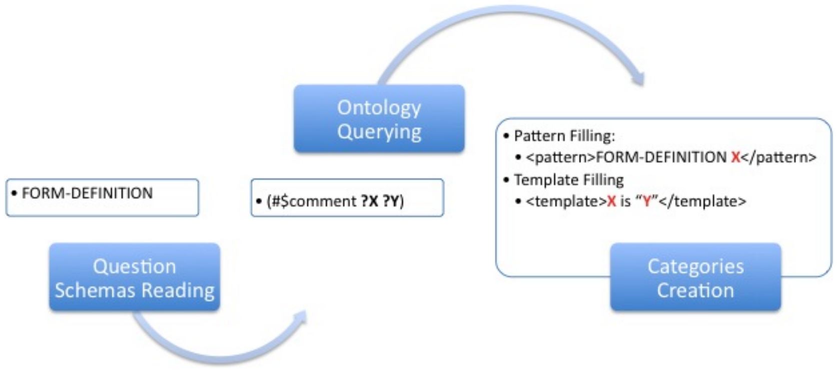

Fig. 3. AIML Bootstraping: an example

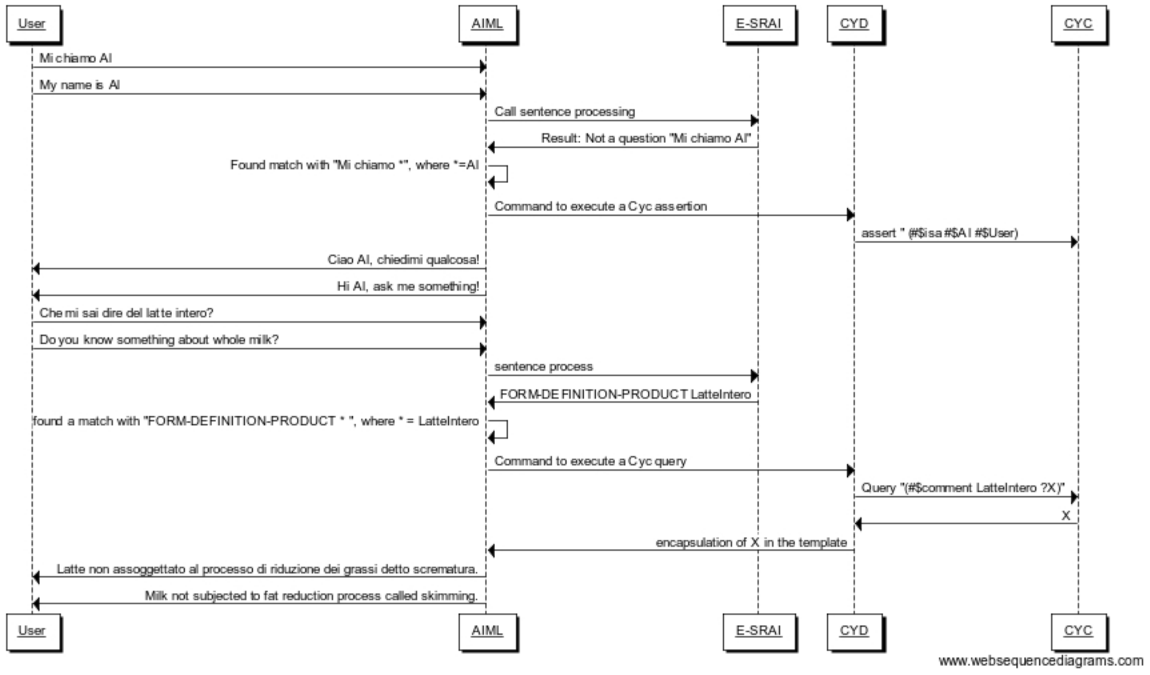
User
AIML
E-SRAI
CYD
CYC
Mi chiamo Al
My name is Al
Call sentence processing
Result: Not a question "Mi chiamo Al"
Found match with "Mi chiamo *", where *=Al
Command to execute a Cyc assertion
assert " (#$isa #$Al #$User)
Ciao Al, chiedimi qualcosa!
Hi Al, ask me something!
Che mi sai dire del latte intero?
Do you know something about whole milk?
sentence process
FORM-DEFINITION-PRODUCT LatteIntero
found a match with "FORM-DEFINITION-PRODUCT * ", where * = LatteIntero
Command to execute a Cyc query
Query "(#$comment LatteIntero ?X)"
X
encapsulation of X in the template
Latte non assoggettato al processo di riduzione dei grassi detto scrematura.
Milk not subjected to fat reduction process called skimming.
www.websequencediagrams.com

Fig. 4. An example of interaction among the modules

experience within the FRASI project, regarding the design and implementation of a Question Answering System specifically suited for italian milk weaving factory. The approach exploits an ontology to construct dynamic answers as a result of an inference process about the domain. This helps the botmaster to obtain smarter and up-to-date answers to user questions, without rewriting the AIML categories. Besides, the ontology is exploited also to automatically populate, off-line, the chatbot KB by translating properties and relations between concepts into AIML categories. Moreover a preprocessing module analyzes user sentences, and subsequently simplifies them in order to obtain a pattern that can be matched with existing elements of the chatbot KB. This allows the botmaster to write one rule for each set of AIML patterns to be matched.

Acknowledgment

This work has been partially supported by Italian MIUR (Ministero dell'Istruzione, dell'Universita' e della Ricerca) within the FRASI "FRamework for Agent-based Semantic-aware Interoperability" project.

References

[1] O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, and C. Jacquemin, "Terminological variants for document selection and question/answer matching," in *Proc. Assoc. Comput. Linguistics*, 2001, pp. 1–8.

[2] R. P. Schumaker and H. Chen, "Interaction analysis of the alice chatterbot:a two-study investigation of dialog and domain questioning," in *IEEE Transactions on System, Man and Cybernetics- parta A: Systems and Humans*, 2010.

[3] K. OShea, Z. Bandar, and K. Crockett, "A conversational agent framework using semantic analysis," in *International Journal of Intelligent Computing Research (IJICR)*, vol. 1, 2010.

[4] A. M. M. Neves, F. A. Barros, and C. Hodges, "Iaiml: A mechanism to treat intentionality in aiml chatterbots," in *ICTAI,18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006)*. IEEE Computer Society, Nov. 2006, pp. 225–231.

[5] F. Agostaro, A. Augello, G. Pilato, G. Vassallo, and S. Gaglio, "A conversational agent based on a conceptual interpretation of a data driven semantic space," *Lecture Notes in Artificial Intelligence*, vol. 3673/2005, pp. 381–392, 2005, proc. of AI*IA 2005: Advances in Artificial Intelligence: 9th Congress of the Italian Association for Artificial Intelligence, Milan, Italy, September 21-23, 2005. (eds. Stefania Bandini, Sara Manzoni).

[6] R. Wallace, "The anatomy of a.l.i.c.e." resources avalaible at: http://alice.sunlitsurf.com/alice/about.html.

[7] R.Moore and G.Gibbs, "Emile: Using a chatbot conversation to enhance the learning of social theory," in *Huddersfield, U.K.: Univ. Huddersfield*, 2002.

[8] Al-Zubaide, H. Ayman, and A. ISaa, "Ontbot: Ontology based chatbot," in *2011 Fourth International Symposium on Innovation in Information Communication Technology*, 2011, pp. 7–12.

[9] S. Quarteroni and S. Manandhar, "A chatbot-based interactive question answering system," in *Decalog 2007: Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, 2007, pp. 83–90.

[10] B. A. Shawar, "Chatbots are natural web interface to information portals," in *Proc. of INFOS 2008 The Sixth International Conference on Informatics and Systems*, 2008, pp. NLP101–NLP107.

[11] T.-A. Dobrila, "From semantic web knowledge to a functional conversational agent: A practical approach," in *http://airtudor.com*.

[12] R. Sumbaly, A. Kumar, G. Paruthi, and S. Malhotra, "Artificially intelligent grid assistant (a.i.g.a)," in *International Conference on High Performance Computing*, 2007.

[13] E. Freese, "Enhancing aiml bots using semantic web technologies," in *Proc. of Extreme Markup Languages*, 2007.

[14] "ResearchCyc," http://research.cyc.com.

[15] "Artificial intelligence markup language (aiml)," resources avalaible at: http://alice.sunlitsurf.com/alice/aiml.html.

[16] "Gate," http://gate.ac.uk.

[17] "ANNIE: A Nearly New Information Extraction System," http://gate.ac.uk/ie/annie.html.

[18] "JAPE: Regular Expressions over Annotations ," http://gate.ac.uk/userguide/chap:jape.

[19] K. Coursey, "Living in cyn: Mating aiml and cyc together with program n," CYC, Tech. Rep., 2004.