PyGenbot for IoT: a demonstration of how to generate any restricted stateless AIML FAQ-chatter bot from text files

Giovanni De Gasperis, Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Italy, email: giovanni.degasperis@univaq.it

***Abstract*—Internet of things applications (IoT) are required to interact with the user in the best natural possible way; the voice based conversation is the ultimate human-machine interaction in terms of easy to use and requirements from the user part, which also has the advantage for the user to interact hands free, non necessary watching a computer screen. Chatter bots are conversational agents that simulate, and capable to sustain, a conversation with a human. Technology do exists that allows to create a lexical knowledge base to be used by a restricted chatter bot, i.e. expert on a specific dominion. This work shows a methodology of restricted chatbot generation using Python program, called PyGenbot, that is capable to derive an AIML (Artificial Intelligence Markup Language) knowledge base starting from a simple textual data set, including: a FAQ, a keywords, a stopwords, a multiwords and a glossary file set. Any WOA attendee is welcome to supply arbitrary and simple formatted text files; then using PyGenbot, I will first edit the text input files needed to generate automatically the corresponding AIML knowledge base set that can be used with any standard AIML interpreter to implement the desired chatter bot, which can then be integrated into an IoT application.**

I. INTRODUCTION

Internet of things applications (IoT) are required to interact with the user in the best natural possible way; the voice based conversation is the ultimate human-machine interaction in terms of easy to use and requirements from the user part, which also has the advantage for the user to interact hands free, not necessary watching a computer screen such as the scenario of a car driver. Many commercial solutions have come recently from major smartphone corporations, mostly specialized on the smartphone usage scenario: sending messages, handling the calendar, fix an appointment, searching for a restaurant close by. Also, home appliance with similar capabilities have shown up in the market. Most of the time, these are proprietary solution, not readily available to developers, but strictly integrated into commercial products, or proposing a licensed cloud API. The voice recognition phase is not in the focus of this work. So I give it for granted that do exists a hardware device, or cloud API that provides the voice-to–text recognition task. The focus in this work is to generate a proper textual reply to a text generated by the user by any means, typing or talking.

I offer a general purpose tool that can be applied to any IoT application, with limited computational capability by using AIML automatically generated chatter bots. They are conversational agents that simulate and sustain a conversation with a human, mostly in a restricted knowledge domain. Since ELIZA [1], text pattern recognition based chatter bots have come a long way [2], [3] . A.L.I.C.E. is an handy crafted chatter bot composed of about 50'000 lexical categories edited by a community of about 500 authors [3], aiming to be unrestricted in its knowledge as a tentative to pass a limited implementation of the Turing test known as the Loebner Prize [4]. A.L.I.C.E.'s lexical knowledge base is described using the Artificial Intelligence Markup Language, AIML [3]. The lexical categories in AIML are defined by means of $(pattern, template)$ tuples in a XML derived syntax:

```
<category>
   <pattern>WHAT IS LINUX</pattern>
      <template>
       Linux is an open-source
       computer operating system
      </template>
</category>
```

Also, different categories with a common semantic background can be linked together by means of a SRAI connection:

```
<category>
   <pattern>WHAT IS GNU LINUX</pattern>
      <template>
       <srai>
        WHAT IS LINUX
       </srai>
      </template>
</category>
```

In this way a tree of SRAI connection can link all of the different lexical forms to their lemma. Using wildcards it is also possible to filter out common words, isolating keywords.

The reasoner, i.e. AIML interpreter, is a LISP program proposed by Richard Wallace [3] designed to search for the best text pattern matching given the user input so to give the most appropriate answer during the written conversation. The IBM question answering system (QAS), known as Watson, won a challenge of the kind human-versus-machine [5], using brute force search algorithm on an unrestricted knowledge domain. However, in this work I concentrate to automate the generation of only **stateless restricted chatter bots**, given that their lexical knowledge can be expressed by means of a combination of a frequently asked question/glossary set, keywords, multiwords and stopwords lists. The input data

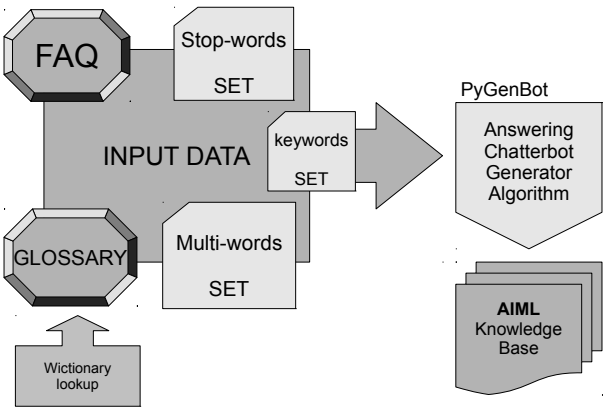

Fig. 1. Input data set and information workflow of the AIML generation process.

set can be arbitrary, so **the methodology is unrestricted**, but not so for the final products, i.e. the FAQ chatter bots. Also, the overall method does not depend on the language, so multilingual IoT applications can be readily designed with parallel free text corpora for each language. Here is shown how to apply the PyGenbot Python program [6], [7] designed to automatically generate Artificial Intelligence Markup Language (AIML) knowledge bases.

II. INPUT DATA SET

The set of textual data in input is defined as the following:

1) a frequently asked questions (FAQ) file F
2) a glossary file G
3) a keywords list file K
4) a stop words list file S
5) a multiwords list file M

All files are simple free text documents, with some basic structure in order to distinguish text of the questions from text of the answers, or glossary items and their respective definition. Keywords, multiwords and stop words are just files containing a word on each line, or separated by comma. The FAQ file F is completely defined by the chatter bot designer. It contains questions and answers in the simple form:

```
Q <question> | {Q <alternative version>}
A <answer  > | {A <alternative version>}
```

Alternative versions of the question are useful to enlarge the possibility to intercept the user input; alternative answers are great to increase the variability of the answers given by the chatter bot in response to the user input. The multiwords list is very important to isolate conceptual entities that uses more than a words, as for example "*operating system*" or "*credit card*".

The input data set and the information workflow can be summarized by the diagram in Fig.1.

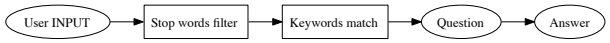

Fig. 2. Ideal textual data processing from the user input to the right answer.

The glossary item definition can be enriched using the free online dictionary [1] or by using the Python NLTK [2]. Glossary items should cover the most significant terms of the restricted knowledge domain about which the final chatter bot is designed to be expert. Keywords should be selected from the text of the questions in order to optimize the pattern matching the user input. The stopwords list are just the most common words of the language, i.e. elements of structural and connective lexicon. The input set does indeed determine:

- the language (English , Italian, German, etc..)
- the restricted knowledge domain
- the vocabulary

The generated chatter bot is considered to be stateless since it can only demonstrate a purely reactive behavior, given a textual stimulation. A more sophisticated prototype could be built upon adopting proactive multi-agent system logic frameworks like DALI [8] or AgenSpeak/Jason [9] middle layer. By the way, a stateless chatter bot is what is needed in the majority of IoT applications were an appliance need to give a correct answer to a user or to set up a working parameter to accomplish a user given task.

III. THE PYGENBOT PROGRAM

PyGenbot is a Python program with about 750 lines of code that takes as input the text files set and produces an AIML file set, ready to be uploaded to any AIML interpreter, which finally implements the actual stateless chatter bot able to interact with the user. The usage scenario is analog to use a compiler to produce machine language (AIML) from high level source files (FAQ, keywords, multiwords, stop words), even if in this case the underlying natural language is not context free as programming languages.

The algorithm as been published in [6], [7]. The reference idea is shown in Fig. 1, by which the construction of this kind of restricted chatter bots is inspired.

PyGenbot generates three set of AIML files:

- the FAQ/keywords/multiwords categories
- the glossary categories and "**WHAT IS ***" question patterns
- the stopwords filtering categories

The FAQ/keywords/multiwords AIML set can grow to several thousands of categories, so the generation algorithm needs to be tuned by the maximum number of categories each AIML file can contain, given the complexity of the FAQ/keywords/multiwords text file set and the final AIML interpreter tool adopted. The output AIML 1.0 file set is then ready to be used in a AIML hosting web services, like http://pandorabots.com .

[1] http://en.wiktionary.org last accessed June 2016
[2] http://nltk.org last accessed June 2016

IV. QUALITY ASSESMENT

It is necessary to introduce a measurable metric of the correctness of the final chatter bot. As already proposed in [7], a three level metric can be adopted:

- **Level 0**: the resulting chatter bot does give a correct answer for all the questions included in the FAQ, with exact text matching
- **Level 1**: the resulting chatter bot does give at least 50% of correct answers, not using the exact wording of the original FAQ questions text, but with the same semantic
- **Level 2**: the resulting chatter bot does give at least 50% of correct answers using questions with completely different wording, but same semantic of the original FAQ questions

It has been experimentally proven that all chatter bots generated with PyGenbot are at least of Level 0 quality, and very often can reach Level 1 quality if the FAQ/keywords/glossary set is accurately designed and well written. The demonstrator at the WOA 2016 Workshop is aimed to confirm experimentally this statement.

V. CONCLUSION

The proposed demonstrator, the PyGenbot program, is capable of generating lexical knowledge bases for AIML based stateless chatter bots. This work illustrated the underline engineered knowledge-unrestricted methodology, also proposing a quality assessment procedure that should objectively demonstrate that restricted chatter bot can be generated starting from arbitrary text files, independent from the language.

REFERENCES

[1] J. Weizenbaum, "Eliza a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[2] R. Epstein, G. Roberts, and G. Beber, *Parsing the turing test : philosophical and methodological issues in the quest for the thinking computer*. New York: Springer, 2008.

[3] R. S. Wallace, *The Anatomy of A.L.I.C.E*, ser. Parsing the Turing Test. New York: Springer, 2008, pp. 181–210.

[4] M. Mauldin, *Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition*, ser. AAAI '94 Proceedings of the twelfth national conference on Artificial intelligence. AAAI Press, 1994, vol. 1, pp. 16–21.

[5] S. Baker, *Final Jeopardy: Man vs. Machine and the Quest to Know Everything*. New York: Houghton Mifflin Harcourt Publishing Company, 2011.

[6] G. De Gasperis, "Building an aiml chatter bot knowledge-base starting from a faq and a glossary," *Journal of e-Learning and Knowledge Society-English Version*, vol. 6, no. 2, 2010.

[7] G. De Gasperis, I. Chiari, and N. Florio, "AIML knowledge base construction from text corpora," in *Artificial intelligence, evolutionary computing and metaheuristics*. Springer, 2013, pp. 287–318.

[8] G. De Gasperis, S. Costantini, and G. Nazzicone, "Dali multi agent systems framework, doi 10.5281/zenodo.11042," DALI GitHub Software Repository, July 2014, DALI: http://github.com/AAAI-DISIM-UnivAQ/DALI.

[9] R. H. Bordini and J. F. Hübner, "BDI agent programming in agentspeak using *Jason* (tutorial paper)," in *Computational Logic in Multi-Agent Systems, 6th International Workshop, CLIMA VI, Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science, F. Toni and P. Torroni, Eds., vol. 3900. Springer, 2006, pp. 143–164.