YouBot: A Simple Framework for Building Virtual Networking Agents

Seiji Takegata, Kumiko Tanaka-Ishii
Graduate School of Information Science and Technology, University of Tokyo
13F Akihabara Daibiru, 1-18-13 SotoKanda Chiyoda-ku, Tokyo, Japan
takegata@cl.ci.i.u-tokyo.ac.jp, kumiko@i.u-tokyo.ac.jp

Abstract

This paper proposes a simple framework for building 'virtual networking agents'; programs that can communicate with users and collect information through the internet. These virtual agents can also communicate with each other to share information that one agent does not have. The framework - 'YouBot' - provides basic functions such as protocol handling, authentication, and data storage. The behavior of the virtual agents is defined by a task processor ('TP') which can be written in a light-weight language such as JavaScript. It is very easy to add new functions to a virtual agent. The last part of this paper discusses the micro-blog system 'twitter' and other web services as information sources that a virtual agent can utilise to make its behavior more suited to the user.

1 Introduction

Recently, communicating in short sentences, such as via Instant Messenger or SMS, has become more common; the use of 'Twitter', especially, is spreading very quickly and widely. These networking tools are not only for chatting, but also for gathering information on and discussing a world of topics. Short sentences are suitable for Natural Language Interface processes like question-answering, recommendation, or reservation systems; thus, Natural Language Interfaces are becoming increasingly important in this area of communications.

There are many dialogue systems that process natural language as a user-input, like 'UC' (Wilensky 1987), 'tour guide' (Prodanov et.al. 2002), but most of them are designed for a specific individual purpose, so, have to locate different systems for different purposes. This problem has been one of the main barriers preventing dialogue systems from being adopted more widely.

Our framework -'YouBots'- can accept the user's messages as input, and respond in natural language. The behavior of these agents is defined by task processors ('TPs') which can be written in a light-weight language, eg. JavaScript. It is very easy to add new TPs to a virtual agent. Web-browsers like Firefox have a similar add-on mechanism and, through open-source collaboration, now have thousands of types of extension. We hope that, in the same way, developers will be encouraged to write new TPs for our YouBot framework.

Personal Digital Assistant is an example of this kind of application. Its schedule manager, contact manager and to-do list are easily implemented on this framework. Q&A system is another example; it would be realized by cooperating with webservice or other external system.

The framework also has a unique networking feature to help the bots communicate with each other: It is called 'Inter-bot communication', a feature which expands the ways in which the virtual agent can get preferred information for the user.

2 Outline of the Networking Bot

Most existing dialogue systems only use their internal data. So their application is often limited to a specific purpose, as in domain-specific expert systems. Using a network feature enabling bots to communicate with each other, our system can obtain many types of information from other, external systems. Figure 1 shows users communicating with their own bots, and bots communicating with each other to collect information for their users. Each connection in the figure is conducted by XMPP protocol[1].

[1] http://www.xmpp.org/

Proceedings of SIGDIAL 2010: the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 273–276,
The University of Tokyo, September 24-25, 2010.

Information in each bot can be linked in the same manner as web pages, and combine to form semantic structures in the way of the Semantic Web (Berners-Lee 2001), this can improve the bots behaviour.

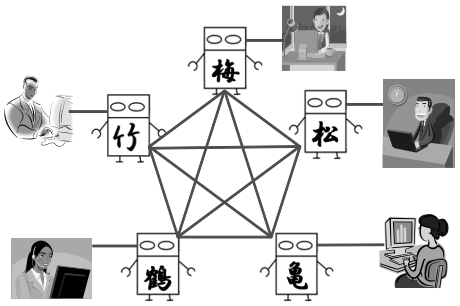

Figure 1: Network of Users and Bots.

If TPs are designed to share information through the network, a user need not know which system contains the information he or she needs. They only need to talk to their own personal bot, then the bot will find the information for them. Each user has their own bot, and can share information through these bots. The modes of interaction with other users and modes of information gathering depend on how the TPs are written.

3 Task Managing

Within our framework, a 'task manager' invokes a 'task processor' as shown in Figure 2:-

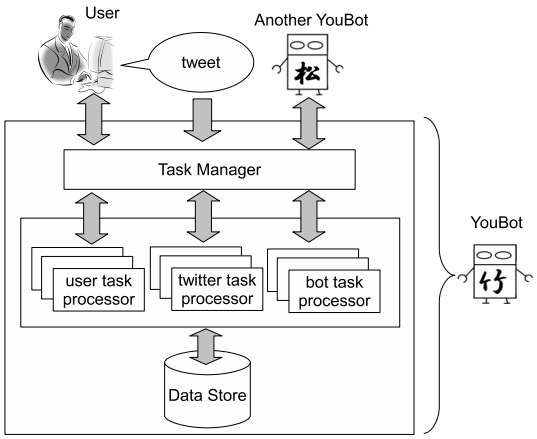

Figure 2: Task-managing

There are existing systems that process tasks with modular components - TPs; among these, we find two approaches, one is centralized and the other distributed. In the centralized approach, a user-message is analyzed by a central component of the system, often called the 'dialogue manager'. Then the dialogue manager decides which TP to invoke. The 'Smart Personal Assistant' (Nguyen et al. 2005) uses 'BDI Theory' (Bratman 1987) to determine the user's intention in the dialogue manager. Then, a TP which satisfies the user's demand can be selected. In this approach, interpretation can be carried out efficiently, but the task manager needs to be revised every time a new TP is added. This is not an easy operation unless the task manager is configured to recognize the functionality of a new TP automatically. This may be viewed as a serious weakness of the centralized approach.

On the other hand, there is 'RIME' (Nakano et al. 2008) which adopts a distributed approach - where the user-message is sent to each of the TPs, which interpret it and return a 'score' indicating how well they can handle the message. Consequently, the TP returning the highest score will process the user's message. This approach suffers from the inefficiency of having to interpret the user's message many times in each TP. On the positive side, there is no need to revise or redesign central components when a new TP is added.

We have decided to adopt the distributed approach because we think expandability is more important than speed. Our framework uses 'Scripting Engine' in which JavaScript codes can run. JavaScript is very easy to write, owing to which, many people write extensions for Firefox in which JavaScript codes can also run. How simply a TP can be written is a very important factor in the attraction of developers.

4 How to Write Task Processors

There are three types of designated TP in the YouBot system: a 'user task processor', a 'bot task processor', and a 'twitter task processor'. The 'user-TP' is for processing messages from the user - explained in the 'Basic Task Processor' subsection (see below); the 'bot-TP' is for processing inquiries from other bots - explained in the 'Inter-bot communication' subsection (see below); and the 'twitter-TP' reads the user's tweets at the Twitter site - explained in the 'Cooperation with External Services' subsection (also see below). Each TP is saved to an individual JavaScript file in the 'task' folder with a .js extension. The YouBot Framework reads these files when the program starts and when a 'reload' command is issued.

4.1 Basic Task Processors

The JavaScript code for a basic TP needs at least one variable and two functions. The variable 'type' indicates the type of task - which can either be a user-task, bot-task, or twitter-task. The mandatory functions are 'estimate' and 'process', an approach introduced in the 'Blackboard' multi-agent system (Corkill 1991). The 'estimate' function receives a user-message from the task manager and returns its score, which shows how likely it is that this TP will be the best among the other TPs to process the message. For example, when a TP uses pattern-matching for message interpretion, the score may be higher if the matched pattern is more complicated, or may be zero if no pattern matched the user message. The 'estimate' function can use not only pattern-matching, but also various data calculated or stored in different ways; such as the dialogue history or information from external systems. The YouBot Framework gathers and compares the scores returned from the TPs, then selects the processor which returned the highest score to process the message. The 'process' function of the TP handles the user-message and makes a response to the user. During the processing, this function can access the internal data store or an external system to get or save various information.

4.2 Pattern Matching

Our framework provides a handy way to do pattern-matching, using four types of placeholder:

An OR conditional placeholder is defined by "{abc |def }" format.

```
I {will go|went }to school.
```

matches both "I will go to school." and "I went to school." Optional selection can be defined with this "(abc|def)" format.

```
Yes (I do |it is).
```

matches "Yes I do", "Yes it is" and just "Yes" Using "[abc]" format, the content of the placeholder can be retrieved. For example, the pattern:

```
I went to [place].
```

matches the sentence "I went to school." or "I went to see a doctor." If the pattern matches the user's message, an object holding the contents of the placeholder will be returned. You can get the contents with the "get" function, specifying the placeholder - in this case "[place]"

To define a placeholder which matches only one specified pattern, "<abc>" format is used. For example, the placeholder "<date>" can be defined so that it matches a date expression such as 'yesterday' or 'on Sunday'. Then the pattern:

```
I went to [place] <date>.
```

matches "I went to school on Sunday", but does not match "I went to school with my brother". The content of <date> placeholder can also be retrieved with "get" function. Retrieved data can be kept in the data store and used in interaction with the user later.

4.3 The Data store

Many chatter bots don't remember what they have said before. 'A.L.I.C.E' (Wallace 2008) has a short memory - just one single interaction. Unusually, YouBot has a long-term data store for its memory. It holds key=value style properties which can be defined by the TP. To save schedule data, as in:-

```
type="schedule"
date="2010/05/14"
item="Submission dead-line"
```

- we create a new data object, set its properties, and use the 'save' function. To retrieve specific data from the data store, a 'data selector' object is provided. If the following condition is set up in the data selector:-

```
type="schedule"
date="2010/05/09"
```

- then a list of matching data is retrieved from the data store. The Youbot framework also provides a facility for responding to inquiries from other bots, and this raises security issues. In this framework, a default security filter is installed in the data selector to send information only to privileged bots. Data objects saved in the data store have security attributes for which the default is 'secret', and only the owner of the bot can access this information. This attribute can be set to 'private' or 'official' - then, the information will only be accessible to the bots which have 'private' or 'official' privilege. Developers do not have to worry about this data security setting during inter-bot communication.

4.4 Inter-bot communication

A user-TP can send an inquiry to another bot - about, for example, the user's schedule or knowledge and expertise. The TP generates an 'Inquiry Sender' object , sets the inquiry and the target bot's address, then uses the 'send' function. This

inquiry is formatted as an inter-bot message so that the receiving bot can distinguish it from user-messages. The receiving bot generates an 'Inquiry Responder' object for each of the incoming inter-bot messages; then the Task Manager sends the messages to the bot-TP. Next, the bot-TPs estimate the likelihood of processing the message and return scores - with the bot-TP which returns the highest score being selected to respond. A responding message is sent back to the inquiring bot in the inter-bot message format. then a function named 'convey' - within the inquiring TP - is invoked to make a response to the user. A function named 'timeout' is invoked when no response has been returned.

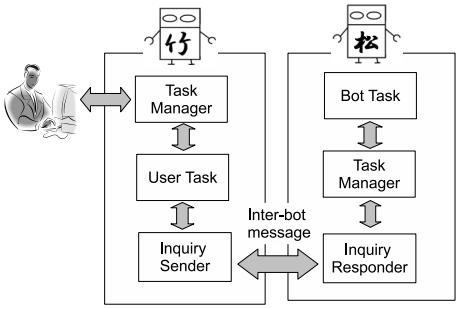

Figure 3: Inter-Bot Communication

To respond to an inquiry from another bot, a bot-TP for that inquiry has to be defined. Besides which, remote bots have to be given privilege to collect information which has a security attribute restricting access. If a TP developer fails to specify a security attribute for the data, no access will be allowed without the right privilege, because the default setting is secret.

4.5 Cooperation with External Systems

A bot can read the user's tweets at the twitter site at specified intervals. The User's tweets are sent to twitter-TPs, then estimated and processed in a same manner as user-TPs and bot-TPs. A bot can get information about a user's status, interests, and favorites; these data are useful for generating preferable responses for the user.

The Youbot framework also provides a utility function which takes URI and retrieves HTML code. This function can be used to access search engines or news sites. Services such as online shopping or recommendation engines represent the type of business model that would be suited to the application of the Youbot framework.

5 Interaction Example

The following are examples of interactions which YouBot might handle:

```
USER: I will meet John at 9 tomorrow.
SYSTEM: Is that A.M or P.M?
USER: pm
SYSTEM: There's a meeting with Mr. Smith at 8pm.
USER: It's been canceled.
SYSTEM: I see.
```

6 Conclusion

We proposed a simple framework for virtual agents. Its functionality can be easily extended by adding task processing modules written in JavaScript. The Youbot framework provides utility objects which make task processing even easier. Networking ability is also provided to expand the networked information's reach, while data security is maintained. Future work will include normalizing the estimation score. Another challenge is how best to share contextual information among TPs so they can interact to generate better responses for the user.

References

Robert Wilensky. Ther Berkley UNIX Consultant Project. *Informatik-Fachberichte*, volume 155, pages 286–296, Springer, 1987.

P. J. Prodanov, A. Drygajlo, G. Ramel, M. Meisser, and R. Siegwart. Voice enabled interface for interactive tour-guided robots. *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1332–1337, 2002.

T. Berners-Lee and J. Hendler and O. Lassila. The Semantic Web. *In Scientific American*, pages 34–43, May 2001.

R.S. Wallace. The Anatomy of A.L.I.C.E. *Parsing the Turing Test*, pages 181–210, Springer Netherlands, 2008.

A. Nguyen. An agent-based approach to dialogue management in personal assistants. *In Proceedings of IUI-2005*, pages 137–144. ACM Press, 2005.

M. Bratman. Intentions, Plans, and Practical Reason. Harvard University Press, 1987.

M. Nakano, K. Funakoshi, Y. Hasegawa, and H. Tsujino. A Framework for Building Conversational Agents Based on a Multi-Expert Model. *In Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 88–91. ACL, 2008.

Daniel D. Corkill. Blackboard systems. *AI Expert*, volume 6, pages 40–47, 2008.