

A Novel Personal Agent Framework for Web Services and Commercial Systems

Angie F. Shia

Softgent LLC, California State University, Chico—
Department of Computer Science
angie@softgent.com

***Abstract** – Softgent Personal Agent system (hereafter refer to as PA system or framework) is a platform independent, hybrid artificial intelligent agent framework. The application is based on the Knowledge Worker agent (hereafter refer to as KW agent) framework [8]. The PA framework varies slightly from the proposed KW agent framework. The reason for the variance is a combination of software engineering principles and commercial application constraints. This paper focuses on the PA framework. The reader can assume the design follows the proposed KW agent framework unless otherwise noted.*

Keywords: Agents, web services, commercial systems

1 Introduction

Softgent Personal Agent system (hereafter refer to as PA system when discussing it as an application and PA framework when discussing its architecture) is a platform independent, hybrid artificial intelligent agent framework. Its purpose is to assist a user with processing and retrieving emails, files, data (from databases), and recreational requests such as searching for an appropriate restaurant or booking a movie—all through a single user interface (portal). The PA system is particularly useful for non-computer savvy or physically disabled users.

The PA framework is based on the Knowledge Worker agent (hereafter refer to as KW agent) framework [8] proposed by this author. The PA framework varies slightly from the proposed KW agent framework. The reason for the variance is a combination of software engineering principles and commercial application constraints. This paper focuses on the PA framework. The reader can assume the design follows the proposed KW agent framework unless otherwise noted.

The PA framework is open and platform-independent. It can support different technologies, such as email system, file system, databases, web services, or any custom application in which a suitable adapter can be created. This in turn makes the PA user interface a single point of entry (portal) to access data on different platforms and delivery mechanism.

The web version of the interface is a thin client that is particularly useful as an interactive web agent to assist user in finding information that is traditionally served by FAQ or knowledge base. If the FAQ and knowledge bases are supported by keyword scoring type search engine [1], [2], [3], that is, for each word that a document matches, it gets a score, then any document that contains all of those keywords gets a high score. This potentially returns a long list of results which the user must sieve through to get to what they want. The problem with using keyword scoring is that it does not focus on semantics, but on syntax. Even though many keyword-based search engines today consider the location and frequency of the words in the document, the context in which the words are under is still not fully addressed [4]. The Natural Language Processing (NLP) and fuzzy logic sections in PA attempt to fill the semantic void. They break the data into knowledge trees and analyze the trees. The heuristic is based on how the tree is structured and what is described in its knowledge base. The structures of the trees are dependent on user input, which are indeterministic. Therefore, the decision making can be considered dynamic; even though it is bounded by pre-programmed semantic and heuristic rules.

2 Architecture

Figure 1 shows the high level architecture of the PA framework. Figure 2 shows the proposed KW agent architecture. The overall idea is still intact, with some differences on how the CBR systems are

represented by the adapter-to-external application modules.

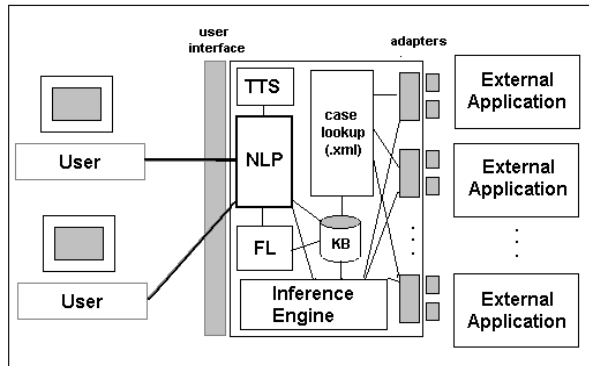


Fig. 1 PA Framework Architecture – High Level

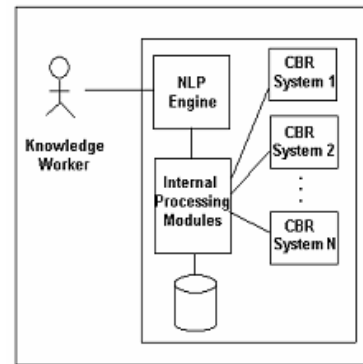


Fig. 2 KW Agent Architecture – High Level

3 Language and platform

PA system is a platform independent application. The code is written entirely in the Java language. It uses many Java based third party libraries that are not part of J2SE. Table 1 lists these mandatory external libraries. There are a few samples that use other third party libraries, but they are not part of the core. The application can run on Unix or Windows platform as long as the correct third party libraries are downloaded for each platform. The application uses XML (Extensible Markup Language) as its data representation. There are around 40 classes and over 60 supporting files in the PA system. Table 2 summaries the list and shows only the file structure in the PA system.

TABLE 1

List of 3rd party libraries

Name	Description
mail.jar, smtp.jar, imap.jar, mailapi.jar, pop.jar	JavaMail APIs for email functionality
activation.jar	Java Activation Framework library from Sun Microsystems. It is a supporting library for email and TTS functionalities.
freetts.jar, cmu_time_awb.jar, cmu_us_kal.jar, cmulex.jar, cmutimelex.jar, en_us.jar	FreeTTS is a free Text-To-Speech library. It is a joint effort from Carnegie Mellon University and Sun Microsystems.
Tomcat Servlet Engine and libraries	Tomcat is from Apache, but is part of Sun's Java Servlet technology. This is used for PA system's web interface only.

TABLE 2

File structure in the PA system

Name	Description
pa	Contains pa.jar, shell scripts, and startup files
pa/agent	Contains PA interface and inference modules
pa/cbr/cases	Contains adapters to the external applications
pa/cbr/utills	Contains commonly used methods

Name	Description
pa/fl	Contains the fuzzy logic module
pa/nlp	Contains the code to process user input in NLP
pa/tts	Contains PA's client to the freeTTS engine
pa/sf	Contains the knowledge base for the whole system. All are text based files.
pa/shared	Contains mainly configuration files
pa/logging	Provides basic logging capability
pa/doc	Contains documentation
pa/web	Contains the alternative web interface classes

4 Technologies involved

The PA system uses many commercial-based technologies. The application can be broken into commercial and AI modules.

The commercial and industrial technology modules include:

- Java programming language (J2SE), XML, Tomcat Servlet Engine, Java Activation Framework (JAF) for speech and Axis support, JavaMail, JSP, and Servlets. Optional libraries: SOAP, Java Web Services, and Axis SOAP Toolkit.

The AI modules include:

- Natural Language Processing, Fuzzy logic, Text-To-Speech (TTS) technologies from CMU and Sun.

5 Design considerations

The PA system can be implemented in numerous combinations of programming language and data representation, and paradigm. This section discusses why the system is implemented in the current manner.

5.1 Language and Data Representation

The foremost implementation goal is portability and interoperability. In order to support this goal among a myriad of different application systems and platforms, the system uses platform independent language and data representation. XML is the natural choice for data structure representation due to its wide support; both in academia and industry. Text based or flat files are chosen as the knowledge base for maximum portability.

Java is chosen as the programming language because the language abstracts the details or the underlying operating systems and allows the same code to run on different platforms. This makes the code maintenance easier. For the same reason, only Java-based third party applications or libraries are used.

5.2 Introduction of Adapters

The expert system modules shown in the KW agent system are only representative. In a real system, these applications are external to the framework. In order to allow the system to know they exist, yet not make them part of the system, *adapters* are introduced. Furthermore, these adapters must not be hard-coded into the system. This requires that the external application systems be abstracted into a common interface. The paper discusses this issue later on. The adapters will facilitate the communication between the PA system and the application themselves. The system currently has six basic modules or adapters built in for demonstrative purpose only. Table 3 lists these adapters. None of them represents their external counterpart fully. Each of these modules are independent from one another. As mentioned earlier, these adapters are not hard-wired into the system. They can be swapped in and out. The inference engine will choose the

appropriate adapter at run-time.

TABLE 3
Sample adapters in the PA system

Name	Description
MessagingMgr/EmailManager	Sample adapter to an email server
LDAPMgr/FileManager	Sample adapters to a file system
QueryMgr	Sample adapter to query database or web pages
ScheduleMgr	Sample adapter to task management
EntertainmentMgr	Sample adapter to entertainment system
WebServiceMgr	Sample adapter to a remote web service

5.3 Object-oriented programming

The PA system uses object oriented programming (OOP) principle [5] to resolve many issues in the system. One of them is the runtime discovery issue outlined in section 8.1. The runtime discovery and instantiation is resolved by using Java's Reflection and introspection capability. The common interface issue is resolved by introducing a hierarchical class structure. The Application Programming Interface (API) are defined in a parent class and inherited by each adapter so that every adapter instance is invoked with their code logic. The results are also normalized so that no matter how the original data looked like, it is normalized to a single format.

Each module or adapter inherits the GenericManager abstract class. The GenericManager class defines a “contractual agreement” between the inference engine and the subclass. It states that any class that inherits it will have certain methods. The engine uses object reflection and introspection to determine which actual child class it should instantiate and call (caselookup.xml provides the name). The actual child object only needs to be identified and instantiated at runtime, not at load time. This means an administrator can add a new adapter and restart the server and it will be available for the user. The current implementation compromises on this goal. Section 8 explains it in detail.

6 Adaptation and addition

Certain suggestions are implemented differently from the suggested approach in the KW agent paper due to engineering constraints. One such example is the suggestion of Fuzzy IF-Then statement. Instead of using long if-then else statements, the implementation introduces the idea of “constraint” objects. A constraint is a restriction of the data set. This restriction is derived from the fuzzy logic module (constraints are configurable by the system administrator). For example, when a user asks for something *good and nearby* restaurant, the system creates two constraints objects on the universal restaurant data set.

Another adaptation is the calculation of the fuzzy graph. Fig 3a below is an original graph from the KW agent paper. The system represents it with a trapezoid instead, as in Fig 3b. Since Java Standard Edition (J2SE) does not come with graph functions, the fuzzy module calculates the trapezoid function by using positive and negative linear functions for the two ends. The MembershipFunctions.java class contains these formulas.

The PA system also adds an extra layer – cryptography. This is added because the knowledge base contains all “flat” files. Flat files are smaller (less overhead) and easier to port than databases, which makes them more suitable on systems with limited resource, but they have no security. Hence the data in the flat files are encrypted by default and decrypted only during system initialization. The decrypted data exists only in the memory cache.

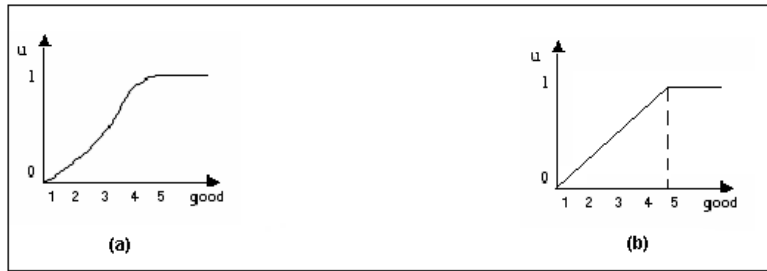


Fig 3. Sample membership functions of the word "good"

7 Functionality

The current version contains these functionalities:

- Natural language recognition as input to query or retrieve information.
- Fuzzy logic based heuristic to handle ambiguous input.
- Portability and interoperability
- Email Management (create, store - IMAP only, retrieve, send)
- File Management (copy, move, delete, find)
- Query Management (runtime discovery of database schema and web pages)
- Task Management (reminders)
- Entertainment Management (offers restaurant suggestions to user)
- Web Service Management (able to access remote services)

8 Limitation and issues

Due to time constraint, certain suggestions in the KW agent are not implemented. There are also some issues that arise from engineering standpoint. Even though most of these issues are not critical, but they point out certain deficiency in the current system.

8.1 Runtime instantiation

As stated in section V, the system should identify and instantiate adapter objects at runtime. However, the current implementation did not do that. In order to improve efficiency, the PA system did preload some of these adapter objects and kept them in cache, which the engine then identifies and requests for them at runtime. These objects exist throughout a session; instead of just a request (a session can have multiple requests). The reason for this compromise is because adapter objects are created for each request. Once the request is over, these objects lose their references when they exit a method or class. They then end up as orphan objects and float in the system until the garbage collection kicks in. Each of these objects take up memory and the memory is not returned until they are garbage collected. Even though you can call the garbage collector and set the memory level in which it should run, there is no such guarantee [6], [7]. This can bogged down the system if large amount of requests comes within a short time frame. Furthermore, garbage collection is an expensive operation as the garbage collection does a full memory sweep. Hence, in order to combat this potential issue, a compromise had to be made. The PA system can bypass preloading adapters when the above issue is resolved later on.

8.2 Multi-sentence support

The current NLP engine cannot handle multi-sentence input. Each request can only come in the form of a single sentence.

8.3 Context-based input

The current system cannot handle interactive dialogue. Interactive dialogue requires the system to keep track of context. Context based input means a user enters a sentence with the assumption of a previous input. There are two issues with this: one, the input may not be a sentence. Two, the data is continuous until the user ends the context. In a way, it is similar to many network protocols such as TCP (Transfer Control Protocol) where data is sent continuously between a start and end flag. Like a network protocol, flags must be defined for the system. Otherwise, the system doesn't know when or how to process the data. However, this is difficult as users do not work with protocols. Also, some level of intelligent data association may have to be made between user inputs.

8.4 Memory management and optimization

The system is somewhat lax in memory management as it is expected to run in a standard Intel Pentium or higher computer in a non-critical environment. However, if the system were to be used in an embedded environment, such as in robots or PDA, the system must be optimized. Many sections of the code may also have to be rewritten using the real time specification for Java (RTSJ).

9 Conclusion

The PA framework provides artificial intelligence into commercial system. This harnesses the value of commercial system and not alienating them, as a purely AI system might. The module based approach is not only appropriate from a software engineering point of view, but it also allows proven AI technologies to be added as appropriate, for example, neural network and Automatic Speech Recognition (ASR). On the other hand, it minimizes the impact of unproven or speculative AI technologies or theories and allows them to be tested in piecemeal in a real environment. The platform and language portability is an added benefit from not only a interoperability point of view, but an investment as well. The system is now an open source project at SourceForge.net, renamed to Java Personal Agent (JPA). The website address is <http://www.sourceforge.net/projects/spa-ai>.

10 References

- [1] E Liddy, "How a Search Engine Works", *Searcher*, vol. 9:5, 2001.
- [2] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", *WWW7 / Computer Networks 30*, pp.1-7:107-117, 1998.
- [3] L. Barlow, "How To Use Web Search Engines ", *Monash.com*, 2004, <http://www.monash.com/spidap3.html>, retrieved March 31 2006.
- [4] D. Sullivan, "Major Search Engines and Directories", *SearchEngineWatch.com*, 2004, <http://searchenginewatch.com/links/article.php/2156221>, retrieved March 31, 2006
- [5] R. Pressman, "Object-Oriented Concepts and Principles", *Software Engineering: A Practitioner's Approach*, pp. 550, 2001.
- [6] S. Wilson, J. Kesselman, "The Guarantees of GC", *Java™ Platform Performance - Strategies and Tactics, 1st ed.*, Section A.2, 2001
- [7] T. Lindholm, F. Yellin, *The Java™ Virtual Machine Specification, 2nd Ed*, Sec.3.5.3, 1999.
- [8] A. Shia, "To Illustrate the Benefit of Using Fuzzy Logic As Part of the Reasoning Model For Knowledge Worker Assisting Agents", *Proc. from International Conference on Computational Intelligence for Modeling Control and Automation (CIMCA 2005), 28-30 November 2005, Vienna, Austria, IEEE/CIMCA, 2005*.