\textbf{Preliminary Information}

\textit{“An intelligent-agent building system”}

We have developed a cutting-edge intelligent agent dialog system to allow a system to have complex conversations, in a very similar way to real humans, w/o human intervention.

\textbf{The Several Parts of a whole}

Our system consists of several modules, tied together as one; allowing our customer to design, build and debug a complex dialog system in a very fast and efficient way.

\textbf{Preliminary Description}

There is a remote development studio (like any IDE) to allow developers to program the scripts, debug and test/debug them on-line, as simple as that! The developer team gets access to a web-application (license) allowing group-wise developers to edit and test the conversation-controlling programming of several agents (called somehow scripts). The scripts are written in a new language called \textbf{AgentScript}, which resembles much like a human novel-dialogue more than a computer programming system. Instead this dialog packages are enhanced with the power of special scripting (the possibility of embedding complex logic inside). The whole dialog-description-packs are managed by means of a full flagged web-interface, where the users can debug and test in a very sharp way all the features of the conversational building blocks. Even there are statistic modules capable of analyzing the whole development, testing timing for benchmarking and much more.

\textbf{At a first Glance}

The resulting dialog system (called from her out: agent) bears a lot of intelligence built-in, ranging from simple semantic and syntactic recognition to complex co reference (anaphora) resolution, inside the talking-script, like taking about "that" which is translated to the "object" or issue talked about in the last chat-turns.

On the other hand, a novel embeddable \textbf{DDL (Dialog Definition Language)} language called itself \textbf{AgentScript} has more than 120 special functions and logic to deal with linguistics in a simplistic way. Thus our thinking-engine is fulfilled with really a lot of power-NLP processing; it also has a complete plug-in structure to incorporate any external module, containing as well a full flagged morphologic-error-correcting-multilingual-dictionary (over 1 million English words and 2 millions words in Spanish), a shallow-chunking parser, anaphora solver, deixis-switcher, a simple reasoning engine, an many more stuff.

\textbf{Under the hood}

The system contains a real computer language (dialog-script) compiler, to provide high-quality flash-speed processing (has a blazing speed of 40k logic-NLP-pattern test per second, typically using a 2000 lines complex conversational script the typical processing time lies far under 50 milliseconds). Our AI conversational motor is an advanced system containing a lot of state-of the art technologies tied up intelligently together (we don’t use a simple AIML-like motor). Most of our developments have been published under scientific papers, presented towards the AI & NLP community, obtaining several good comments from the scientific stuff.

The Platform, itself its written in C# (.net), has almost 620k lines of code (~40 modules, 600 classes & lots of programming and documentation) including phonetical matching, full flagged spell-corrector, spanish and english morphologic analyzer, both handling even non-existent inferred words, a full GLR parser & parser-generator many AI sections as well and actually it is a native compiler, automatic compiling scripts \textit{(much simpler and far better than hand coded AIML)} into C#, which compiled results in a highly efficient conversational system, which can emulate many ways of thinking and yields deep conversational skills. We allow to add native C# functions and libraries (its extensible itself) also it generates internally MSIL to allow dynamic linking (no need to compile altogether). The whole stuff is like a framework which works in a highly abstract object-like language resembling knowledge elements, strongly similar to real things and concepts, allowing to do math, logic inference and many NLP operations like concordance, generate and number checking, conjugation, ontological inference etc.
Preliminary Information

Architecture
Here in Fig.1 we show a simplified diagram and schematics, stating major blocks and functions. The several web-interfaces are shown as a single screen, but are far more complex and rich.

Chat Interface
Our conversation agent engine has an interface to the almost any protocol. We offer as well a simple and rich web-based control interface called web-Farm to allow any agent to chat simultaneously in any available protocol; we serve natively XML (remote clients), HTTP, SMPP (a mobile SMS standard) and any of the standard Chat-Protocols called itself IRC. The included IRC protocols are almost any of the actual existing ones, including: MSN / AIM / GTalk / ICQ / XMPP / etc.

Hardware/Software Requirements
The whole system is offered as SAS and or/installed into a client (where the volume license is worth it). The software runs on a windows (2000/3/8) server and doesn’t require too much resources, it’s fast & lightweight, the total amount of RAM depends on the programming of each agent and the total simultaneous conversations to be held, as an example a 1800 rule based system uses 60 megabytes of ram. The whole lexical server (dictionary + thesaurus) uses ~50 megabytes for 2 languages (spanish and english).
A Success Story
We have had a big initial success in having over 8.3 million conversational-lines in as low as 4 months for a spanish avatar of a cook in Uruguay, called Maestro Saman, and also an automated tourism advisor agent, called Yorugua serving the Uruguay Department of Tourism during 2009 summer until 7/2010. The mean traffic of both agents was over 10 simultaneous answer/responses per second, during the summer!. This project was built along with a Microsoft Business partner interacting with MS-Advertising LA.

A shallow comparison
The most frequent question around all this technical-linguistic blah-blah is

¿How does the system behave, on the real world?.

The answer is far from being simple, but depends primarily on the skills of the dialog-programming team and the skills and amount of content deployed into the system.

¿But, then: how do you compare to other systems?

Here the answer is strong and direct:

The power of a programmable system lies inside on how you can make it solve complex decisions with very few programming lines wrapped around a logical and simple syntax.

And precisely this was the design goal of our system:

“To make it simple and complex at the same time!”

Let’s see an example:
To program a whole linguistic math calculator (as an answering system) where you might ask: ¿how much is 3 + 8.9 plus nine?, besides allowing the user to input almost any complex math expression, mixing up math operators, and numbers (all they as numerical symbols as well as words), even mixing up romanic numbers (ie.: MCMXII).

To unleash this with AgentScript: you need only 2 lines of simple scripting: one for the pattern matching and other for the response, no parsing needed, no interpretation needed, no nothing! ➔ As simple as that!

But a new answer rises immediately: ¿Is this simple enough to be handled by anyone? or a more complicated question could be ¿Do you need a skilled professional?:
And the answer is simple: someone who can speak, and knows the type of answers the agent should provide, of course knowing the agent’s purpose, is enough!, if he/she knows a little about logic or scripting, more better. No need to know high-level programming languages at all.

Another question may rise: ¿What are the limits of the system?

And the answer is simple:

The limits are the unsolved normal human word-ambiguity problem, and misunderstanding, not even solved in normal conversations among humans, but we provide a lot of tools to solve even co-reference (pronouns, referring before-told entities) with enough success to surprise a dialoging human (again: a single built-in function).

Another limitation is that you cannot make a dialogue with large reflexive-declarative sentences, because the agent (nor a human) cannot easily extract anything useful out of this kind of constructs, because of the many possible co-references and miss-interpretations.

Our system points to concrete simple conversations, with context, using as few words as possible, with fast and intelligent answers, as simple as that.
Even providing a good search machine like Google with too many keywords leads into nowhere useful or even missing the seeking point.

So the comparison center lays less than two items:

1. How simple and powerful is your language, and
2. How fast does the outcome perform under production, with ‘heavy’ programming?

Both answers are positive: (1) the language its very powerful but simple and abstract enough to set up a dialogue easily, the interface and thinking mechanisms to create the dialogue are straightforward and intuitive, and (2) the resulting system performs very fast and well under production, outperforming similar systems like AIML among other brute-force attack brains.

Comparison to Other products

In the meantime, we saw a lot of products on the market and tested them thoroughly, without biasing toward our development (sounds hard, but it is the fact)

The conclusion is sad and disappointing: all they are mere simple pattern-matching systems, having statistical based inference, they seem mostly be based on (or similar to) the old MIT’s Eliza psychoanalyst, and its new nephew: an old-fashioned XML revamp called AIML.

Even big companies like the one providing Star-Alliance, Wal-Mart and eBay the ‘new’ intelligent conversational agents lacks to have a decent conversation and answer nonsense to many good intended questions, not even allow the user to write with a single spelling error: they won't answer at all, or answer nonsense’s or simply pick up randomly a pre-built answer!

We saw a common problem on that they all lack to manage normal dialogue-turn situations, feeling rude inside a conversation. Their behavior is mostly limited only to have memory of the last question or answer, or the user name in turn, answering only to what you say at any time, no context, no memory, no relations, no... nothing.

No one of the former (among many others) remembers anything but the user’s name or age or certain pertinent information. They all have no goal on a conversation nor they plan what to say and when to say it, they lack modal and emotion handling...they are not intelligent, they only seem to be smart... but they are not!

Our Conclusions

Our conclusion is that most of the commercial offered dialogue systems, aren’t dialogue systems at all: they are template-limited question and answer systems trying to make sense of the dialogue. But they are definitively a good approach to dialogue, when there is nothing to compare with, like our system, but for sure: the time and users will tell which system will be the best.

On the other hand, our system is also programmable, with a vast powerful scripting language, and also it handles just ‘out of the box’ a lot of ”internal affaires” such as automated co-reference, pronoun solving, factual memory, math, units, spell correction, and many more to let the dialog programmer to build easily a next generation intelligent dialogue system.

Let’s see what happens next..

Andres Hohendahl
CTO, PandoraBox