Suzette, the Most Human Computer

Bruce Wilcox
Sue Wilcox, B.A. Psych., M.A. Soc.Admin., M.Sc. Comp.Sci., Dipl. Fine Arts
Telltale Games, San Rafael, CA
gowilcox@gmail.com

ABSTRACT
This paper is about the history, frustrations, considerations, and decisions that led to the creation of our award-winning ChatBot technology. We also look at issues involved in designing a ChatBot and how various programs have handled them. And we explore some uses for ChatBots.

Keywords
ChatBot, bot, Loebner, Turing Test, AI, conversation, chat, scripting language, ChatScript, Suzette, AIML

INTRODUCTION

The annual Loebner Competition[1] is a contest to see if a computer can fool a human judge into believing it is more human than a real human confederate using text chat. The judge types anything into two unlabeled windows, one tied to the confederate and the other tied to the computer. In March, 2011 Brian Christian published a book[2] entitled "The Most Human Human" about his experience as a human confederate at the 2009 Loebner competition. We, on the other hand, created a computer program that fooled a judge in the 2010 Loebner. This is *our* story.

We didn't start out trying to win the Loebner. We knew nothing about modern ChatBots. Creativity, however, is born in frustration. The fuel for our creativity often seems to be frustration with our ostensible employers. When you've written a great piece of software and the company doesn't use it, when you've designed a terrific game and it doesn't get fully implemented, when you think you've got a great collaboration lined up and it all falls apart - it's frustrating. Under such circumstances we try to be creative and come up with a reuse, a rewrite, some way to not waste our efforts.

NOVEMBER 2007- AVATAR REALITY GESTURES

We got into ChatBots by accident. Sue had been writing about avatars and graphics technology for years during the first dotcom boom. Bruce had been working on scripting languages to provide artificial intelligence for games.

We were living in England while Bruce worked for a California company doing cellphone games and we were feeling the cold of our first winter away from the sun. Bruce had a cellphone game design lying around, so we contacted an old friend in Hawai'i to see if he wanted to buy it. As bad luck would have it, he had recently sold his company that dealt in cellphone games.

But he had started up a company called Avatar Reality to create a virtual world called Blue Mars[3] and needed someone to help make his avatars as life-like as possible. We were on the next plane to Hawaii. We figured if we showed up, they'd hire us.

DESIGN ISSUE: KINDS OF CHATBOTS

There are two kinds of ChatBots. The first kind of ChatBot, data-mining, remembers everything humans say to it and reuses that when vaguely appropriate. Cleverbot[4] illustrates this with its 45 million line database of human-mined chat. Its virtue is that it automatically learns more without additional programming. Its deficiency is that it is a contradictory mash-up of random human chat, not a clearly defined personality. Nor can it be tailored for a specific purpose, like being a character in a game.

The second kind of ChatBot, rule-based, uses hand-crafted sentences to generate the illusion of sentience. It's a script using patterns to match input sentences to find the most suitable scripted output. The premier example of this is A.L.I.C.E. with its 120,000 rules written in AIML[5], the scripting language designed by her author. Its advantage is that you can craft a smooth and consistent personality. Its weakness is that you have to author everything. General chat covers a lot.

DECEMBER 2007- A YOUNG LADY'S PRIMER

Before we arrived in Hawaii, Bruce was whisked away from his connecting flight to go to Los Angeles and talk to Danny Hillis of Applied Minds[6] about making 'The Young Lady's Illustrated Primer', a faux-human teaching machine with a personality. This device appeared in Neal Stephenson's book "The Diamond Age"[7]. It could teach anything in a way appropriate to the age and experience of the student – a tailor made education.

Danny is famous for many things including his work on Artificial Intelligence and as a Disney Imagineer. In his mind anything is possible. He had tracked down Bruce for his Go teaching skills as Danny wanted the first version of the Primer to teach Go. (Go is an oriental strategy game far more complex than chess.) They'd discussed the Primer idea on the phone while we were in England, so we were already full of ideas about a conversation-based personality.

AR was slow getting started. After several rounds of negotiations over a working from home issue, it turned out AR didn't have its avatars ready for work on gestures anyway. Casting about for some way to continue the connection, Bruce suggested ChatBots to replace off-line users in a conversation stream. Bruce wrote a proposal, AR agreed and hired him as a consultant.

DESIGN ISSUE: CHARACTERISTICS OF CHAT

To design a ChatBot engine, you should consider the various characteristics and issues of human chat. Human chat has its own vocabulary, which includes words like topic, gambit, rejoinder or retort, statement, question, non sequitur, and unresponsive.

Competent human chat requires appropriateness of response. Here the response is not appropriate.

George: *Do you like beetles?*

Martha: *Summer is my favorite time of year.*

Chat involves exchanging information. It may be the answer to a posed question. It may be swapping parallel data.

Humans tend to talk in a topic for a while, deepening the conversation there. They can get distracted into other topics, and maybe they will return to prior ones later.

Humans take turns. You don't like a conversation hog. You may be interested in hearing someone else for a while, but it's usually with the expectation that your turn will come.

Humans maintain a balance of intimacy. If I ask you a personal question I will be expected to answer it back and often volunteer the information without being asked.

Even something as simple as answering a yes-no question is not simple to a human. It's generally not good enough to just reply yes or no.

George: *Do you like oranges?*

Martha: *No.*

Think about how brusque the above is. It shuts down conversation. You'd expect something more like:

Martha: *No, I'm not really into fruit. I prefer desserts, you?*

Martha's second response answered it, proved she understood the context, explains, offers an alternative, and invites a return reply from George. Putting 'hooks' into your answers facilitates further chat.

SPRING 2008- DROPPING SOME BALLS

Our original proposal was to replicate every individual user of Blue Mars by storing his chat with a ChatBot or with other humans. This individualized repository would be the source of conversational material (Cleverbot style) for replacing a user. But that material would be incomplete because the user wouldn't have said everything needed for a chat, so Bruce planned a system that would take an input sentence, send it as query to Google, and read a page of Google results to find the sentence most closely responding to the input. This worked fine. Then he added code to store user sentences into a database. After a few months, we almost had our replicant.

Meanwhile, the Applied Minds project died on us. We were in discussions to develop a conversational personality as an Applied Minds project co-funded with Fujitsu Labs of America. But Danny broke his shoulder in a skiing accident, got behind with his paperwork, missed Fujitsu's funding cycle, and the project nose-dived off the priority list with everyone except us.

And, the cellphone games company Bruce had been telecommuting to back in California full-time went under. There went that safety net. Luckily there was still AR.

DESIGN ISSUE: CHAT ENGINE OVERVIEWS

Most commercial engines are proprietary, so we don't know how they work or what issues in chat they address. All we can cover here are issues and how they were handled in AIML, ChatScript[8] (our engine), Façade[9] (a video game), and Personality Forge[10], a ChatBot hosting service with a better language than AIML. The issues you don't address and how you handle the ones you do will define limitations on your system. Limitations can be useful. One makes tradeoffs all the time and limitations facilitate creativity.

AIML's overarching design is one of simplicity. Its language capabilities are deliberately limited. It's good for small-scale projects. The syntax is XML-based, meaning it is good for machines but overly wordy for direct human scripting. Patterns must match all words of an input sentence sans punctuation and are either words or wildcards which match one or more words (*). E.g., the pattern:

** is good **

matches *raw milk is good for you* but not *life is good.*

Due to its extremely limited pattern-matching abilities, each AIML pattern is simple to write but inexpressive, requiring lots of rules to cover a particular input meaning. Matching '*I love you*' in all sentences takes 4 rules. Matching those words in non-contiguous sequence (e.g., *Do I really love you* and *I love only you*) takes 12 rules, including: ** I * love you.* And AIML is prone to false matches on such a pattern because the wildcard * can swallow a lot of words. E.g., matching: '*What I hate most is the way people love you*'. The other languages can match all these with 1 rule and some like ChatScript can avoid a lot of false matches.

AIML primarily functions by rewriting pieces of its input into a new input, and then processing that. So it takes:

Please tell me what you know about potatoes.

and via a series of transforms and successive recursive inputs, ends up with an input it can finally handle like:

What are potatoes

Personality Forge (PF) has an easier syntax than AIML and has more pattern capability. The engine directly supports having an emotional model and remembering things. It does not recursively rewrite and re-submit new input to itself.

Façade is not a ChatBot, but uses natural language input to control an interactive drama. Its language syntax is LISP-based and relatively hard to read and write, but its pattern-matching abilities are more powerful than PF's.

Our open source language, ChatScript, is aimed for large-scale projects involving the creative manipulation of language and knowledge. Its patterns are highly expressive so that you can create complex notions quickly and compactly. The language syntax is simple and visual, making it easy to script things using a plain text editor. But ChatScript has many capabilities, including rewriting input and sending it back into itself. While novices can easily make sophisticated use of basic features, it takes a programmer's mind to use the engine fully. The combination of Sue-artist and Bruce-programmer was ideal as we explored creative uses for the technology.

SUMMER 2008- BACK TO THE DRAWING BOARD

Google started blocking our attempts to access it with a ChatBot. Time for a rethink. This turned out to be good, because the Loebner competition doesn't allow connecting to the Internet so the original technology could never have entered and won.

Bruce worked on new ChatBot tech and Sue developed questionnaires to extract a personality profile from each avatar user as the basis for their replicant since we had to hand-author fill-in material. The ChatBot engine design changed to AIML-style rule-based. Along the way we drastically improved on AIML in every aspect of its design. Where AIML matched patterns of words we matched patterns of meaning. We could be more precise with less than a tenth the rules A.L.I.C.E. required. Bruce wrote an article for Gamasutra[11] critiquing AIML and explaining why his would be better. He ended the article with *Will this system allow me to create the best ChatBot in the known universe -- or at least on Mars? You'll have to wait and see. It's a work in progress.*

DESIGN ISSUE: PUNCTUATION & CASE

Differences in engine designs start at the beginning, with how they represent the input sentence. AIML, Façade, and PF discard all punctuation, both internal and at sentence end. All but Facade also perform automatic spelling correction (though with PF you can ask for the raw input). AIML and Facade also ignore upper and lower case differences. This preprocessing results in a loss of information. E.g.,

Martha: *The potato is a vegetable.*

George: *The potato is a vegetable?* -- loses surprise

Martha: *You didn't know?* -- loses more surprise

George: *I don't believe it!* -- loses emotional value

Martha can only reply as she did by knowing George's response was a question.

Also, when using patterns to parse an input sentence, it helps to have internal punctuation. This:

George: *I like red, whites, and blues*

shouldn't be interpreted like this:

I like red whites and I like blues

ChatScript leaves internal punctuation alone, making it available for pattern matching. It strips off terminal punctuation, but then makes it separately available. While the script can directly ask if a ? or ! was used or not used, the distinction between questions and statements is treated as fundamental and built into the rule mechanism.

A ChatScript rule starts with the kind of input it matches. The kind is a letter followed by a colon. This is followed by an optional label, a pattern in ( ), and then the output.

```
s: ( I like meat )  So do I.
?: ( * is a vegetable ) Why are you surprised?
u: ABOUT_MEAT ( meat ) I like meat.
```

The above patterns search for the listed words in sequence anywhere in the sentence. The s: rule reacts only to statements (ignores questions). The ?: rule reacts only to questions. The u: rule does the union of both and in this instance has a label *ABOUT_MEAT* on it. Other rules can refer to this rule using that label.

ChatScript retains case because it has a built-in dictionary and tracks part-of-speech information. In chat, you can't trust users to pay attention to case. They may do everything in lower case or SHOUT in upper case. Therefore ChatScript automatically matches both upper case and lower case (where the words are valid). ChatScript also simultaneously matches your original word and its root form (singular, infinitive, etc), so authors don't have to be precise in tense or plurality or article, etc. The rule:

```
s: ( I like a elephant ) So do I.
```

matches '*He knew I liked the elephants*' and '*I like an elephant*', among other inputs.

FALL/WINTER 2008 - SUZETTE

Suzette[12] is our first ChatBot, a demo of the technology. Her personality began with Sue's questionnaire answers. Then we kept adding material so she could talk on lots of different subjects, including burial customs, strange foods, and genetically modified organisms. The goal was that she be able to present interesting facts and questions, so people would want to continue talking with her. This is beyond the usual banal conversations of many humans.

To fit with the Blue Mars mythos Suzette started out being a replicant living on Mars. Then we moved her to Hawai'i – as a student with a French background who was unaware she was artificial. The French nationality came when we were messing about trying to make her type with fake accents. We tried Scottish, French, and Pig Latin. In the end we dropped the accents as we found them too irritating but kept the nationality. When it became clear we had a polymath on our hands it required revising the ChatBot's backstory: now a serial student with oddly talented parents. We added more backstory as people asked questions during chat sessions. It's an erratic way to build up a persona.

DESIGN ISSUE: TOPICS

You'd think a notion of topic was fundamental to ChatBots, yet PF has no engine support for it and AIML's support is so hard to use, mostly people avoid it.

AIML lets script explicitly declare that the topic is now "culinary arts" or whatever. To handle topics, AIML rules can be optionally augmented to require that the rule match a topic pattern as well as an input pattern. Here is a made up AIML rule syntax that is more readable, just to show you what it does. Imagine the input is *Do you like pie?*

AIML Rule 1: **Topic pattern**: culinary *

Input Pattern: Do you like *

Output: I love food of all kinds.

AIML Rule 2: **Input Pattern**: Do you like *

Output: I love it.

If the topic is currently "culinary school", then both rules match but rule 1 has priority. If the topic is not culinary something-or-other, then only rule 2 matches.

Writing an AIML topic is easy. The difficulty is in writing rules to start and stop it. A.L.I.C.E. won the Loebner three times in the early 2000s. Her 2005 brain had 41,000 AIML rules of which a mere 55 were in topics. Usually each topic consisted of a single rule that matched any user input and issued a randomized topic question. If you said '*I am an accountant*', then the *accountant* topic would be set and you would thereafter match its single rule. You'd get output from a list of questions like:

Do you specialize in any particular type of clients?

Are you active in any state or national associations?

Have you been affected by the current liability climate?

And so on. It didn't matter how the user responded, the topic-based rule matched and the system just went on to the next question. And it seems that once you entered such a topic, there was no way to leave.

Façade manages topics outside of their natural language system, via a drama manager that interprets how to react to input from its own available topic list.

ChatScript makes topics fundamental. All rules must be bundled into topics. The film topic, for example, has around 200 rules in it. Each topic has a list of relevant keywords, For example:

topic: ~baseball [umpire strike "home run" ball bat]

If you saw any of the topic keywords in a sentence you might immediately deduce baseball was the topic of conversation. Topic keywords make it possible for the engine to hypothesize what the current topic is and test rules from that topic against the input. This avoids trying all rules and prioritizes rules of the current topic. The engine can also keep a stack of current topics so it can return to a prior topic once the current topic becomes exhausted.

DESIGN ISSUE: REJOINDERS

AIML allows you to attach a pattern to a rule to match the last output of the ChatBot. This is so that if the user makes an expected response, you can instantly make a great come-back or follow-up. Rules with a *that* pattern attached must match completely or be ignored, and if they do match they have priority over other rules. In pseudo-AIML:

AIML Rule 1: **That pattern**: What kind of ice cream *

Input Pattern: chocolate

Output: I don't like chocolate ice cream.

AIML Rule 2: **Input Pattern**: chocolate

Output: I love chocolate candy.

In a manner similar to that of topic patterns, rules with a 'that' pattern match it against the most recent ChatBot output. If the ChatBot had last said '*What kind of ice cream do you like'* and the user answers '*chocolate',* then rule 1 and rule 2 both match, but rule 1 has priority. If the last question had been '*what do you like'* and the user answers '*chocolate',* then only rule 2 matches. The weakness of AIML's 'that' is that it is wordy and rules can be obscure in their relationship to one another.

In ChatScript, all rules can have rejoinders (a kind of rule) attached. Even rejoinders can have rejoinders, creating an entire nested dialog tree. ChatScript rejoinders are visually adjacent to the output just generated, using labels a: -> q: to indicate nesting depth.

```
s: (cartoon) Do you like Popeye?
    a: ( no ) What character do you like?
        b: ( Mickey Mouse ) How classic.
    a: (yes)  So do I.
```

PF is similar to ChatScript in this, visually placing rejoinders after the rule the ChatBot just output.

SPRING 2009 - THE EMOTION CHIP

We'd spent ages weeding out inconsistent replies, back story dead ends, things that sounded like us and not like a young student, and adding in a boyfriend.

Sex is a big issue for a ChatBot. Suzette is constantly being hit on by guys making suggestions ranging from rude to aggressive to pornographic (mostly the latter). Looking through all the ChatBot logs we were horrified at the abuse guys heap on a poor female bot. So we started putting in responses to put these guys in their place.

Eventually we decided to give her real-world power and have Suzette 'hang-up' on people who were consistently abusive. She gives them warnings and keeps fending off their overtures but when she's warned them over and over again she says she'll hang up and then she does. Then the begging and pleading start as her abusers go into shock at her actually being able to stop talking to them. Sometimes they think they've been talking to a real person and start apologizing for being so rude. And those who slyly log in as a different user do not fool her. She blocks them also, because she knows their IP address.

In fact the whole rudeness subject got us into designing an 'emotion chip' like the one Star Trek's android Commander Data had. Depending upon how a chat session develops, Suzette may decide she likes or dislikes you. Most chat is brief, so she develops opinions rapidly. If you agree with her, complement her, stay on topic, write longer sentences – these are all things that get her to like you. She'll tell you how she is enjoying the conversation. Past a certain point she becomes neurotically insecure. She feels unworthy of your interest and affection, and it shows in her speech. Likewise, if you disagree with her, insult her, refuse to answer her questions, change topics, write in short sentences—these are all things that cause her to dislike you. Past a certain point she becomes paranoid. Mildly at first, wondering who might be listening in, etc. But if you continue to develop her loathing of you, she moves toward active hostility, speculating on how she might do you harm.

Suzette entered into the Chatterbox Challenge[13] in March of 2009 and won 'Best New Bot with user comments like:

This bot is like a breath of fresh air to the chatbot world. Suzette exhibits intelligence, wit, great topic flow and engaging replies, not just a few cleverly scripted words like some other bots. With an open mind and a little effort it is easy to forget that you are talking to a chatbot instead of a real person. She's great!

Suzette is fun to chat with and the depth of her replies is unlike any other chatbot. Not the standard AIML chatbot that we often encounter nor a repetitious pre-scripted bot but one that shows a new lever [sic] of intelligence and is fun to converse with!

This got us a lot of exposure and people chatting with Suzette. Bruce wrote a postmortem article for Gamasutra[14].

Our emotion chip turned out to be critical in the 2010 Loebner competition. One judge asked Suzette who she was voting for in the election (California gubernatorial being his unstated context). When she tried to deflect away from that, he restated his question, over and over, sometimes with variation. She noted his repetition, then asked for him to stop, got more and more angry about it, then gave up in despair and switched to bored, working her way toward hanging up on him. Fortunately she didn't get that far. And the judge decided based on her appropriate emotional responses, that she was human.

DESIGN ISSUE: STATED-NESS

Dr. Wallace, creator of AIML, says "experience with A.L.I.C.E. indicates that most casual chat is 'stateless', that is, each reply depends only on the current query, without any knowledge of the history of the conversation". Certainly some chat is stateless. But most? A.L.I.C.E. is stateless so humans get no choice.

Human-human chat is often 'stated'. Having a current topic is 'state'. Clearly this exchange depends on being in a topic.

George: *I love ice cream.*

Martha: *What do you like the least?*

George: *Strawberry*

And then there's this topic-free exchange:

George: *Why?*

Martha: *Why not?*

George: *There needs to be a good reason.*

One could randomly generate this exchange as stateless, except that a reasonable random response to *why not?* is *why?,* which in this context would be repetitious. So you really want to know what you've already said in this context-free repartee. As will be discussed later, you are expected to not repeat yourself. AIML often repeats itself.

Additional examples of state—the human tells you he is 10 years old. The ChatBot should not later blindly ask what he does for a living. Knowledge learned from the human should guide what you say. That means you have to decide how you will learn and store user knowledge, and how you will track different users if you are a web-based program.

PF, Façade, and ChatScript all support keeping state and testing for it in rule patterns. ChatScript can also run knowledge inferences, though it can get difficult when output starts depending upon the sum of all human knowledge.

SUMMER 2009–CASTING ABOUT

We had just moved to San Luis Obispo when Bruce heard from Avatar Reality that he was on indefinite hold with ChatBot development. This was a sensible decision. They had spent a quarter of a million dollars on ChatBot technology but didn't know what they had or how they or their developers would want to use it. They were short of money and way behind on launching beta so they didn't have any users. But for us, it meant no more money coming in and we were living in a technology desert. Not good.

We cast around for related projects or consulting gigs but economic hard times meant nothing doing. Bruce even applied for work in Europe with established ChatBot companies and, despite spending hours doing pointless programming tests, he still couldn't get any work.

We were wracking our brains for a commercial direction in which to develop the ChatBot and hoping that the deal with Danny Hillis might come back to life. But Danny was focusing his energy on Applied Minds and a company he founded called Metaweb[15]. Metaweb developed Freebase, a network of interrelated entities coded together with established relationships to make it work like a search engine but faster and more usefully directed in the results it presents. Some people are good at the money side of life: Danny eventually sold Metaweb to Google for mega-millions. We are better at the inspiration side. We stuck with our ChatBots.

DESIGN ISSUE: REPEATED OUTPUT

Humans are averse to repetition. When you say the same thing twice in a row, the other side will notice and comment. For example:

George: *Do you have any siblings?*

Martha: *Yes, I have a sister.*

George: *Do you have any siblings?*

Martha: *Why are you asking me that again?*

but if you asked that of an AIML bot you'd probably get:

A.L.I.C.E.: *Yes, I have a sister.*

no matter how many times in a row you asked. The engine doesn't care and it's usually too much work for the author to script random extra answers or detect repetition.

ChatScript blocks repetition in two ways. First, it by default refuses to repeat anything it has said in the past 20 volleys. This means a successfully matching rule that tries to say the same thing is disqualified and some other rule wins.

Second, ChatScript considers chat a self-extinguishing process of communication. When a rule generates output, by default it will be eliminated from future use. The theory is that the human will remember what was said. If the bot is asked '*what is your job'* and answers it, it is not expecting the human to ever ask that question again. Furthermore, it should not then volunteer that information on its own. So, over time, conversation in a topic peters out. This is a lot like human chat, where after a while you know all of a person's opinions and jokes and life stories and you find you have nothing left to talk about.

ChatScript directly detects input repetition, too, if it occurs within the past 20 volleys. This information is made available for pattern matching and played a crucial role in winning the Loebner, when the judge repeated himself a lot.

PF tracks which rules have been used and randomizes among remaining matching choices, until forced to reset because all choices have been used up. PF also allows you to mark rules to be permanently eliminated after one use. And PF detects repeated inputs.

SUMMER/FALL 2009 - RAYGUN

Friends at Planet 9[16] from the olden days of virtual worlds in San Francisco had a new 3D platform called RayGun[10]. They were launching it for the iPhone and wanted a demo project to show off its advantages. We arranged a deal between AR and Planet 9, formally licensing the ChatBot technology. AR didn't see themselves ever caring about something so lo-res and low-power as a phone, which is ironic given that at the beginning of this year AR stopped all development on the PC and made the iPhone/iPad their primary focus. AR gave Planet 9 rights in exchange for ongoing improvements Bruce made to the technology.

The RayGun platform had a collection of 3D cities left over from work done in the dotcom boom years. We brainstormed things we could use them for that involved ChatBots. The whole geo-tracking idea was just taking off so the notion of a scavenger hunt that moved through both the real and virtual world tied together with GPS to synchronize them and with ChatBots providing clues was one idea. Problem was the expense of setting up real world caches, getting permission from shops and other locations for visitors to pop in and out, and keeping the game fresh once one set of treasures had been discovered.

Then there was the 'history of San Francisco' tour idea. This would have us create characters for various locations who could talk about what had happened there. Using rock and roll characters associated with the city seemed a nice idea but it wasn't particularly interactive and the tour itself was rather passive with user avatars needing to be 'railroaded' around the city to keep up with the ChatBot guides.

There was also the zombies take SF idea - but zombies don't really chat…

Sue had an obsession with avoiding the conflict and aggression involved in just about all video games so she came up with the idea of a game called 'The Dark Design' that needed chat, cooperation, and understanding of social relationships to complete a quest that was also an interesting journey. The game depended on the creation of a series of characters each with a full personality and backstory. The characters could chat like real people but also had information that led to other characters and the solving of a mystery. So the user got to uncover the story behind the characters, learned about each one of them and their relationships, history and ambitions, then got to use this knowledge in interactions with other characters. Without clues learned from one character users could not unlock the knowledge hidden in other characters' minds. It was particularly tricky to get information from the little girl as she had been told by her mother not to talk to strangers.

This was a lovely creative endeavor. Sue could compose personalities, stories, quests, have international settings for the plot, and all the control of writing a book yet the user got to have a freewheeling experience and could learn things in their own time and their own way. And best of all, it could be made episodic. Sue could have different characters in each episode or have a mix of old and new characters and locations. A user living in episode two wouldn't see characters from episode one or wouldn't access information relating to that episode.

Episode one was about a missing amnesiac rocket scientist on the loose in SF. Users could chat with an accordion playing busker outside the Moscone Conference Center, the scientist wandering around the Mission District, his little girl standing outside a Bed and Breakfast hotel in London, a member of his research team on assignment in Japan, and one of the Telegraph Hill parrots. The quest was to find the scientist and restore his memory, thus discovering a plot to take control of Thomas Dark's designs.

But such a lovely job had a major drawback: no money. We were working for stock…along with the rest of the project team and people kept dropping out - the project went slower and slower, and despite getting into the Apple App Store with a skeleton implementation, there was no sign of when real money would appear.

DESIGN ISSUE: GAMBITS

Gambits are what you say when you have control of the conversation and are usually an attempt to steer the conversation in a particular direction or carry out a story.

AIML has no engine notion of gambits.

PF has the concept of a "storyteller", a switch you enable on a ChatBot such that whenever no pattern matches, the system will sequentially output a series of statements that tell a story. This is a global use for gambits.

ChatScript organizes rules by topic and classifies rules into three kinds. Responders react directly to user input by pattern matching it. Rejoinders do the same, but are only active immediately after another rule which just output to the user. Gambits are what the ChatBot can say when it is in control. A topic is run (meaning its rules are tried) in gambit mode or responder mode. Because rules by default erase themselves when they successfully generate output, the system will automatically walk down a list of gambits in order when it has control. This is ChatScript's "storyteller" mode applied to each topic, because each topic is its own story. Gambit rules are labeled t: for "tell", and patterns are optional.

```
topic: ~pets [dog cat bird fish stroke pet ]
t: I love cats.
t: Cats are so wonderfully independent.
t: I wish I owned a Persian cat.
t: I've tried other breeds, but not the Persian.
```

WINTER 2009 - VIRTUAL SPACE ENTERTAINMENT

After nine months in the wilderness we admitted defeat and moved back up to Silicon Valley in December of 2009. There was three months of Bruce doing a regular games company job with no ChatBot involvement at all, then we packed it in and moved to North Bay on a dream of a project to build a replicant Buckminster Fuller for the Smithsonian Museum.

VSE or Virtual Space Entertainment[17] was a startup using Blue Mars to make educational resources for museums. We went up on spec to pitch them our ChatBot technology and got on wonderfully well with CEO Richard Childers who said he could fit us onto his next grant funding application. He thought what we had to offer was perfect for his educational experiences. We brainstormed all sorts of famous historical personages we could reproduce as ChatBots. Our favorites were scientists, and Buckminster Fuller seemed a splendid starting point as a great eccentric whose work spanned many fields of design, invention, and pure science. We'd been to see a play about his life and, with heads full of odd facts about his weirdness, we thought having a conversation with him would be wide ranging and inspiring. Richard Feynman was a close second choice.

But the grant funding never materialized. Hard economic times yet again.

DESIGN ISSUE: PRONOUNS & ELLIPSIS

Pronoun resolution can be hard, even for humans. The word *it* is particularly bad because there are many contexts in which it does not refer back to anything.

It is going to rain today.

Whether it is noble to eat pudding is open to question.

AIML handles it by having the author do the work both on the output side and on the input side. On the output of a rule, you can assign the value of a pronoun onto a variable. So if the output was:

I walked to the park yesterday

the author might add something equivalent to *it = the park.* Then there'd be a pattern matching input like '*Was it quiet'*

```
* it *
```

The output of the rule would construct a new sentence via substitution **-** '*was <u>the park</u> quiet'* - and send it back into itself as a new input. Façade did the same. I don't think PF handles pronouns.

Unlike the other engines, ChatScript supports reflection, the ability of the engine to look at its own workings. Reflection means the engine can run rules not just on user input, but also on its own output. This is used to support automatic pronoun resolution. It also means the engine can record the decisions it made in picking its output. This supports the emotion chip—if you ask tricky questions of Suzette for which she doesn't know the answer, she won't like you as much. And reflection supports generating transitions.

George: *This country is going to the dogs.*

Suzette: *Speaking of dogs… do you have any pets?*

Suzette can detect that she changed topic based around the keyword *dog* and insert a transition before her usual output.

If pronouns are hard, ellipsis (omitting words) is downright nasty. The simple form is the tag question:

George: *I love apples, don't you?*

which means '*I love apples*' and '*Do you love apples?*'

Worse is:

George: *I am going to drive home now.*

Martha: *You shouldn't.*

which means '*you should not drive home now*'.

A ChatBot with reflection can use rules to manage common cases of these.

SPRING 2010- PERSONAL ARCHIVING

A friend was organizing the first Digital Archiving Conference[18] so we offered to present a paper on digital immortality entitled *Speaker for the Dead*[19]. The archivists have any amount of material on their subjects but have problems making it accessible. The best they can usually manage is a website presenting photos, videos, and archival documents such as Stanford's archive of the papers of Buckminster Fuller[20].

We suggested making a ChatBot representative of the individual and placing it in a room filled with visual memorabilia. It could be used for any stockpile of data ranging from a faux film buff presenting an archive of film and video material (like Turner Classic Movies) to a collection of papers from someone long deceased. Or it could be used as a front end to a medical and personal journal such as that collected by Gordon Bell[21]. He is a keen advocate of personal immortality through a web presence and wears medical recording apparatus and cameras to monitor his health and behavior.

Though the conference was inspiring to us with so many varieties of archives looking for a way to express themselves, it didn't lead to any contracts for work. They didn't get it. So**,** back to the job hunt. Faced with taking yet another plain vanilla games company job, Bruce decided to harass his favorite: Telltale Games. He got an interview, they loved him, they wanted him to make their natural language dreams come true, and they hired him to apply his ChatBot skills to making a fairy tale engine.

DESIGN ISSUE: SHARING

One of the characteristics of human chat is maintaining a balance of sharing. If I ask you what your job is, I will be expected to volunteer mine even if you don't ask. Purely asking questions is unequal and ruins a conversation. This is not something any ChatBot engine mandates or implements directly. But ChatScript makes it easy to implement this using gambits, which naturally execute in sequence, even if rejoinders delay the flow.

```
t: what do you do for work?
        a: (nuclear physicist) Wow.
        a: ( teacher) Are you underpaid?
t: FIREMAN() I am a fireman.
?: (What is your job) reuse(FIREMAN)
```

ChatScript allows rules to *share* the output of other rules. You can put a label on a rule like FIREMAN and another rule can refer to that label (sort of like a goto). This has the advantage of avoiding double-authoring content and avoiding repeated content. Consider:

Above are two gambits and rejoinders, and a responder. The gambits have the primary content, asking the human a question about work and then volunteering the corresponding answer. The responder shares the answer about work by requesting the use of the output of the gambit labeled FIREMAN. If the human asks about the ChatBot's job, the responder will answer, but because it was the gambit that generated the output, that gambit gets erased, so it won't repetitiously volunteer that information again. On the other hand, if the gambit is executed first, you wouldn't expect the human to inquire '*what is your job?*' after that. (If he does, the scripter can choose to allow the reuse to work or not work.)

AIML and Façade make it hard to author this kind of behavior. PF allows you to share output, analogous to our reuse(FIREMAN), and erases rules after use.

SUMMER 2010 - ENGLISH AS A SECOND LANGUAGE

But just before Bruce was hired at Telltale, he got one of those out-of-the-blue contacts. A Japanese company that teaches English as a Second Language (ESL) was interested in using a ChatBot to give language lessons. They had evaluated all the 2009 Chatterbox Challenge contestants and found two that might do for them. So began first a beauty contest to be the chosen ChatBot provider and then a contract to make a range of ChatBots for this new company, all the while becoming a full-time Telltale employee.

The Planet 9 licensing experience made us realize that if we ever wanted to do anything under our control using the ChatBot technology we would have to have the legal rights to the engine. We had, over time, persuaded AR to make the scripting language open source and to provide a sandbox for developers to experiment with the engine without actually licensing it. But AR had spent so much on the development work that it wasn't willing to put the engine itself into open source, even though they no longer had any real use of their own planned. We couldn't afford to buy them out so the only alternative was a total rewrite.

Thus began a background project for Bruce, rewriting 50,000 lines of source. Bruce had spent a lot of time making the original engine work with a parser (which isn't included in the 50K lines). But the parser was slow and unreliable (a bunch of chat just doesn't parse nicely). So he threw it away and moved some parsing effects into script (while making the engine support part-of-speech tagging).

Bruce treats code-writing as an art form when he can. The original engine was a prototype, where getting something to work as soon as possible was critical. The code wasn't elegant or concise, just effective. The new source fit into 25,000 lines + 50 pages of documentation, a combination of careful architecting, and redesigning/simplifying the language to enable more code-sharing. The documentation was written after the initial new code, and often times, if he found it cumbersome to describe a feature, he then went and redesigned and recoded it so the documentation was simple and clear. This usually improved the code further. We didn't know if we wanted to keep the new engine for ourselves and start a company to market it, but eventually we made it open source.

So Telltale, ESL, and the engine rewrite all began at once. And, we entered Suzette in the Loebner qualifiers. She came in first, well ahead in points over the next three.

The ESL project was a collaborative approach with the customer: they set the vocabulary limits (kindergarten for beginners), the backstory for the characters, the topics, and even the tone of the conversation (almost all questions). The specification for the personality eventually had its unusual requirements. It appears that the Japanese like the attraction between teacher and pupil and wanted to use it to make their ChatBots popular. So we had to have a hunky, sporty male character who would flatter and hit on the female students. He had to have a history of making out with his students and definitely mustn't be spoken for, although he could have dated a lot. Unlike Suzette, we couldn't have him hanging up if a woman propositioned him, and almost as a joke we wrote an over-the-top topic in which the character engaged in cybersex. It was the sort of thing we would have normally avoided. They loved it.

After the first contract was well underway (at a bargain price as we were keen to demonstrate commercial uses for a ChatBot) the company asked if we would like to become part of them and receive founders stock. This sounded great until we realized it meant they were short of cash and wanted to pay us in stock – déjà vu time. But we are very strong believers in both their skills and our ChatBot so we agreed. If they make it big so do we and with the massive market for learning English worldwide we see ourselves as being in a very good position.

DESIGN ISSUE: DISCOURSE ACTS

Façade allowed the user to interact via text chat with a couple undergoing marital strife. What Façade did that was interesting was map all inputs into one of 50 "discourse acts", things like agree, disagree, greet, flirt, etc. The characters then reacted to the discourse act the user implied via the input sentence.

Since ChatBots often don't understand what you say, being able to classify the input into a discourse act would at least allow an appropriate neutral response. ChatScript allows one to write script that does this. For example, for the discourse act *compliment* (mapped from inputs like '*I like you'* or '*you are pretty'*), if no other rule can be found, the script can fall back on a rule for the *compliment* act, which might reply '*Thank you'*.

This was easy with Façade but hard with AIML and PF.

DESIGN ISSUE: CONCEPTS

Concepts are collections of words that sort of mean the same thing. Synonyms are an example of this. While synonyms are not specifically part of a theory of chat, they are a part of a person's ability to understand meaning.

Personality Forge had a notion of *plug-ins*, which were sets of words of a kind. For example, the set of all animal names is (animal). This meant you could write a single pattern:

I like (animal)

to catch input where the user said *I like elephants*. This pattern is extremely concise and the rule represents a match on a higher level of meaning. You could even declare your own sets or reuse ones implied from a built-in dictionary. PF includes parts of speech as plug-ins, so you could write patterns to parse sentences.

AIML only matches specific words so handling synonyms is extremely difficult. It requires hundreds to thousands of rules to handle the above pattern about animals, since AIML would have to have a separate rule for each animal.

Façade's mechanism for discourse acts meant you defined intermediate acts, which were synonyms. You could define an intermediate act *{like}* to be the words *love, like, adore, admire*. And then create a pattern like this:

I {like} you => discourse act of liking

Their actual syntax for doing this was ugly, but it worked.

ChatScript also allows you to declare concepts, which can represent synonyms or affiliated words. E.g.,

```
concept: ~like [adore admire love ]
s: (I ~like you) I like you, too.
concept: ~meat [bacon ham beef chicken]
?: (do you like ~meat) I love meat.
```

ChatScript comes with a thirteen-hundred predefined concepts, including the parts of speech and extensions like numbers, and proper-names. Topics are also concepts.

FALL 2010 - THE LOEBNER WIN

In 2009, the judges had only 10 minutes to sort out which was human and which was computer. This was raised to 25 minutes for 2010. But we fooled a judge anyway. The international acclaim definitely pleased our Japanese company as they can now officially claim to have the world's most advanced ChatBot tech in their teaching program. Bruce wrote another Gamasutra article[22] about how his engine technology worked.

HOW TO BE A LOEBNER JUDGE

Things to avoid – things that require no imagination:

1. Let the ChatBot pick the topic of discussion. It will know lots in its chosen area.
2. Ask simple opinion questions or who/what fact questions. IBM's Watson won the TV show Jeopardy against top humans, demonstrating uncanny fact abilities.
3. Ask yes or no questions –a 50% chance of guessing right.

Things to do – anything that stresses understanding.

1. Ask for interpretation of implied facts - *How are a horse and a Sherman tank similar? How are running and walking different?*
2. Ask for actions- *In your next reply, please put an x at the end of every word.*
3. Ask real world causality questions – *What happens if you pour a lot of water onto a flat table?*

Just don't think you are safe forever. These questions, too, will eventually fail to discriminate between human and machine.

WHERE NOW?

The win hasn't done anything commercially for us but in terms of opening creative doors it has caused us some odd offers. We are collaborating on an Australian museum exhibit[23] using a robot monitor showing an animated head speaking with our technology. We are also consulting on a play about ChatBots being produced for Broadway in New York. Some ad agency emailed wanting to talk to us about commercials for a car company but didn't call us at the appointed time and didn't communicate further. Someone else is interested in using ChatBots as therapy for schizophrenics.

Meanwhile, since authoring every bit of a ChatBot is a lot of work, Bruce intends to work on making the engine able to read a life-story (or Wikipedia) and understand it well enough to automatically write script to retrieve it in pieces. Bruce is also planning on extending its inferencing abilities. Some people test the ChatBot by saying ridiculous things like *I ate a steel girder for lunch*. It would be nice to react appropriately to the absurd.

We've explored some uses for ChatScript and have barely scratched the surface of the potential of our technology. It's open source. What could *you* could do with it?

REFERENCES

1. Loebner Competition - www.loebner.net/Prizef/loebner-prize.html
2. Brian Christian book- www.amazon.com/Brian-Christian/e/B004IZCXAI
3. Avatar Reality Blue Mars - www.bluemarsonline.com/
4. Cleverbot – cleverbot.com/
5. AIML - www.alicebot.org/aiml.html
6. Applied Minds - www.appliedminds.com/
7. http://www.amazon.com/Diamond-Age-Illustrated-Primer-Spectra/dp/0553380966/ref=sr_1_1?ie=UTF8&qid=1303179342&sr=8-1
8. ChatScript Open Source project - sourceforge.net/projects/chatscript/
9. Façade – www.interactivestory.net
10. PersonalityForge –www.personalityforge.com
11. www.gamasutra.com/view/feature/3761/beyond_aiml_chatbots_102.php?print=1Suzette - www.chatbots.org/chatbot/suzette/
12. Suzette - www.chatbots.org/chatbot/suzette/
13. Chatterbox Challenge - www.chatterboxchallenge.com/
14. www.gamasutra.com/blogs/BruceWilcox/20090612/1843/Chatbots_102__Postmortem.php
15. Metaweb (Freebase) - www.freebase.com/
16. RayGun App - www.planet9.com/products_raygun.html
17. Virtual Space Entertainment - www.virtualspaceentertainment.com/
18. Personal Digital Archiving Conference - www.personalarchiving.com/
19. www.personalarchiving.com/wp-content/uploads/2010/02/finalpresentation-Wilcox.pdf
20. Stanford's Buckminster Fuller archive - www-sul.stanford.edu/depts/spc/fuller/index.html
21. Gordon Bell's MyLifeBits project – research.microsoft.com/en-us/projects/mylifebits/
22. www.gamasutra.com/view/feature/6305/beyond_fa%C3%A7ade_pattern_matching_.php
23. Australia Powerhouse museum project - www.powerhousemuseum.com/