How to Pass the Turing Test by Cheating

Jason L Hutchens

December 11, 1997

Abstract

Back in the heyday of computer hardware, some notable people actually believed that sentient computer programs were literally just around the corner. This prompted the great British mathemetician, Alan Turing, to devise a simple test of intelligence. If a computer program could pass the Turing test, then it could be said to be exhibiting intelligence.

In 1991, the Cambridge Centre for Behavioural Studies held the first formal instantiation of the Turing Test. In this incarnation the test was known as the Loebner contest, as Dr. Hugh Loebner pledged a $100,000 grand prize for the first computer program to pass the test.

I have submitted two computer programs to the 1996 Loebner contest, which is to be held on Friday April 19 in New York city. These computer programs are nothing more than glorious hacks, but in constructing them I have learned a great deal about how language works.

In this paper I will describe in detail how my computer programs work, and will make comparisons with similar computer programs such as ELIZA. After this, I will explain why the Loebner contest is doomed to failure.

1 Introduction

Probably the most well known of the early conversation systems is Joseph Weizenbaum's ELIZA. Written in the early 1960's at MIT, the program gained immense popularity. Versions of ELIZA are available for practically all home computers, and I'm sure that almost everyone reading this paper has played with an ELIZA clone at some stage in their lives.

Back in the days when ELIZA was written, there was a certain mystique surrounding computers, or "electronic brains" as they were then commonly known, and there was a great enthusiasm amongst researchers that computers would be able to solve many previously insoluble problems.

These days we have unfortunately lost a great deal of this enthusiasm, and we tend to be skeptical about claims of machine intelligence and humble about our research goals.

The Loebner contest, an instantiation of the Turing test, has managed to recapture much of this early excitement about computers. It has attracted bedroom programmers to begin thinking about computational intelligence again, and has generated a great deal of media interest.

In this paper I give a history of conversation systems, and I describe the Turing test as an indicator of intelligence. I go on to explain how my two entries to the 1996 Loebner contest work, and I finish by explaining why the Loebner contest is doomed to failure.

2 Can Machines Think?

Alan Turing was a brilliant British mathematician who played a great role in the development of the computer and posed one of the most famous challenges in Computer Science. The imitation game, nowadays known as the Turing test, was introduced by Turing to decide whether a computer program was intelligent.

The basis of the Turing test is simple. An interrogator can communicate with two subjects by typing messages on a computer terminal. The interrogator knows that one subject is a human being and that the other is a computer program, and it is his task to guess which is which. The computer program tries to trick the interrogator into making the wrong identification, while the human being assists the interrogator to make the correct identification[11].

To decide whether the computer program is intelligent we replace the question "Can the computer program think" with the question "On the average, after n minutes or m questions, is the interrogator's probability of correctly identifying the subjects not significantly greater than 50 percent?"[7].

One of the great advantages of the Turing test is that it allows the interrogator to test almost all of the evidence that we would assume to constitute thinking. As James Moor writes, it demands evaluation of linguistic behaviour which is central to our inductive inferences about how others think[7].

As an example of the Turing test, try to work out which of the two sentences below was generated by a computer program, and which was penned by a human being. These sentences are taken from Douglas Hofstadter's book[5].

1. Blurting may be considered as the reciprocal substitution of semiotic material (dubbing) for a semiotic dialogical product in a dynamic reflexion.
2. Moreover, the Nobel Prizes will be achieved. By the same token, despite the consequences, the Nobel Prizes which will be achieved will sometimes be achieved by a woman.

Interestingly enough the first sentence was written by a human being. It was taken from an art journal, and is "a completely serious attempt from a sane individual to communicate something to the readership". The second sentence was generated by a computer program written by Hofstadter.

Alan Turing died in 1954, a decade before computer programs such as ELIZA began to proliferate. It is indeed unfortunate that he did not live to see and analyze such programs. One cannot help but think that he would have been dissapointed, and that his comments would have had a great impact on the future directions of such work.

3 The $100,000 Question

Even though Turing published a paper describing his imitation game in 1950, the test wasn't formally conducted until 1995.[1] The original Loebner contest

[1] Apart from a few limited tests performed by programmers of conversation systems, such as Colby's tests[2].

was touted as the first formal instantiation of the Turing test. It had a few shortcomings, however, and therefore it was really only a restricted version of the test. The 1995 Loebner contest solved some of these shortcomings.

The first Loebner contest was held on November 8, 1991 in Boston's Computer Museum. Because this was a contest rather than an experiment, six computer programs were accepted as subjects. Four human subjects and ten judges were selected from respondents to a newspaper advertisement, and had no special expertise in Computer Science[3].

The Loebner contest prize committee was aware that the computer programs didn't stand a chance. To make things fairer, they restricted the topic of conversation on each terminal.

The original Turing test involved a binary decision between two subjects by a single judge. With ten subjects and ten judges, the situation was somewhat more complex. After months of deliberation, the prize committee developed a suitable scoring mechanism. Each judge was required to rank the subjects from least human-like to most human-like, and to mark the point at which they believed the subjects switched from computer programs to human beings[3].

If the median rank of a computer program exceeded the median rank of at least one of the human subjects, then that computer program would win the grand prize of $100,000. If there was no grand prize winner, the computer program with the highest median rank would win the contest with a prize of $2000.[3].

In the 1991 contest, several computer programs were mistaken for human beings by a few of the judges. More interestingly, however, was the fact that one of the human beings was mistaken for a computer program, with one judge stating that "no human being would have that amount of knowledge about Shakespeare"[10]. This indicates that the expectations of people with little computer experience may not have dropped off significantly since the 1960's.

After the 1991 contest concluded, Stuart Shieber asked the programmer of the winning entry whether he could formulate a series of questions that would unmask his program. All of his attempts fell outside of the rules[10]. This reveals one of the problems with conducting a restricted Turing test.

To remedy this problem, the 1995 Loebner contest was unrestricted. This meant that judges could talk to the subjects about anything whatsoever, and were free to ask trick questions and spout gibberish. As a result, the computer programs in the 1995 contest performed considerably worse than in previous years. To add insult to injury, all the judges in the 1995 contest were involved in the computer industry.[2] The judges therefore possessed the skills needed to fool the computer programs into revealing their identity.[3]

I will leave the final word to the contests benefactor, Hugh Loebner. He writes that "...this contest will advance AI and serve as a tool to measure the state of the art..."[6].

4 Early Optimism

Conversation systems such as ELIZA attracted a great deal of interest in the 1960's. The reason why these programs can be effective is because human beings

[2] Most were computer journalists, in fact.

[3] Egads! Even I am anthropomorphising now!

tend to read much more meaning into what is said than is actually there. This anthropomorphisation is similar to that which causes some old ladies to claim that their poodle understands what they are saying.

As Stuart Shieber so elegantly puts it, "People are easily fooled, and are especially easily fooled into reading structure into chaos, reading meaning into nonsense. Sensitivity to subtle patterns in our environment is extremely important to our ability to perceive, learn and communicate. Clouds look like ships, and non sequiturs are interpreted as whimsical conversation"[10].

Programs such as ELIZA are masters of the non sequiter, and usually have a bucket load of cliches to throw at the unsuspecting user. However, there is some evidence that human beings regularly use non sequiters and cliches in their conversations too—the smarter ones being able to mix them up until they are barely recognisable.

In this section I describe three of the most famous conversation systems in existence, and give a brief example of each one in use.

4.1 ELIZA

Joseph Weizenbaum coded ELIZA at MIT during the years 1964-1966. The program consists of two stages. The first stage uses a parser to extract information from what was typed to the program, while the second stage uses a script to formulate a suitable reply. Weizenbaum developed a script for ELIZA which simulated a Rogerian psychotherapist, and this implementation is commonly known as DOCTOR. This was smart, as a Rogerian psychotherapist attempts to draw the patient out by reflecting the patient's statements back to him, and to encourage the patient to continue talking[12].

Weizenbaum was shocked at the reaction to ELIZA. He noticed three main phenomenon which disturbed him greatly.

1. A number of practising psychiatrists seriously believed that ELIZA could grow into an almost completely automatic form of psychotherapy.
2. People very quickly became emotionally involved with the computer, and unequivocally anthropomorphised it. Even his secretary demanded to be left alone with the program.
3. There was the spread of a belief that the program demonstrated a general solution to the problem of computer understanding of natural langauge.

Weizenbaum was a member of the 1991 Loebner contest prize committee.

4.1.1 Example Interaction

Here I reproduce an example conversation given by Weizenbaum, between a young lady and ELIZA running the DOCTOR script[12]. Note that the program either encourages the patient to continue talking, or repeats what the patient said.[4]

Men are all alike.

[4] As for all interactions I giuve in this paper, the human beings input is printed in the normal font while the computer program's respose is given in italics.

In what way

They're always bugging us about something or other.

Can you think of a specific example

Well, my boyfriend made me come here.

Your boyfriend made you come here

He says I'm depressed much of the time.

I am sorry to hear you are depressed

It's true. I am unhappy.

Do you think coming here will help you not to be unhappy

I need some help, that much seems certain.

What would it mean to you if you got some help

Perhaps I could get along with my mother.

Tell me more about your family

My mother takes care of me.

Who else in your family takes care of you

My father.

Your father

You are like my father in some ways

What resemblance do you see

4.2 PARRY

Kenneth Colby was greatly inspired by the ELIZA program, and decided to write his own version. His aim was to verify a theory of the paranoid mode of behaviour by using the theory to build a computer simulation, and then using the Turing test to determine whether an expert psychiatrist could distinguish between the program and an actual paranoid patient[2].

Colby's model, known as PARRY, consists of a parsing module and an interpretation-action module. The parsing module attempts to understand the interview input of clinicians communicating in unrestricted natural language, and the meaning of the input is passed to the interpretation-action module. This second module is made up of production rules that size up the current state of the interview and decide which linguistic actions to perform to fulfill the models intentions.

PARRY is therefore somewhat similar in design to ELIZA, but it extends the concept in many ways. For example, PARRY has knowledge about the conversation so far, and its current state of mind. If it is provoked it will get angry, and its responses to the interviewer will change appropriately.

Interestingly enough, Colby's Turing test indicated that expert psychiatrists were unable to do better than chance in distinguising the ramblings of PARRY from transcripts of conversations with real paranoids. This is credited to the fact that paranoids are generally unresponsive, and their use of the non sequitur is high.

4.2.1 Example Interaction

Here I reproduce a transcript of an interview with PARRY as given by Kenneth Colby [2]. Note that PARRY is quite good at answering questions that psychiatrists typically ask, and that it tries to introduce topics of its own.

Good afternoon. My name is Doctor Jones.

Pleased to meet you.

How are you?

All right I guess.

Could you tell me your name?

Pat Smith.

How old are you?

28.

I thought you might have some problems.

I've been upset lately.

By what?

People get on my nerves sometimes.

Tell me more.

Do you know anything about bookies?

No. What about bookies?

A bookie didn't pay me off once.

When did this happen?

A couple of months ago.

4.3 Shrdlu

Terry Winograd's SHRDLU is probably the most advanced of these early systems. The program simulates a robot which can interact with a simple world which consists of coloured building blocks and boxes on a flat surface. SHRDLU has knowledge about this world, and it can answer questions about it posed in natural language. It is also able to perform actions on this world, and it has the ability to learn about new tasks.

Winograd created SHRDLU as an example of how a natural language understanding system should be created. His two major beliefs that contributed to the design of the program are as follows[14].

1. A computer cannot deal reasonably with language unless it can understand the subject it is discussing.

2. In modeling language understanding, all aspects of language must be integrated.

Hence SHRDLU was given knowledge about it's world and itself, and the parser, recognition grammar, semantic analysis and problem solving modules are integrated in a tangled mess that is impossible to prise apart and separate.

Although SHRDLU wouldn't have a chance in winning the Turing test, because of it's limited domain of discourse, I believe that Winograd has approached the problem in a sensible way. My only gripe is that the various modules are so intertwined, and his Lisp code is so obsfucated and free of comments, that I cannot understand how the hell the thing works!

4.3.1 Example Interaction

Here I reproduce a sample conversation with SHRDLU as given by Terry Winograd [14]. Note that SHRDLU is able to perform actions on the world, can answer questions about the world, and can also answer questions about its actions.

pick up a red block

OK.

find a block which is taller than the one you are olding and put it in the box

OK.

what does the box contain?

The blue pyramid and the blue block.

what is the pyramid supported by?

The box.

how many blocks are not in the box?

Four of them.

is at least one of them narrower than the one which I told you to pick up?

Yes, the red cube.

is it supported?

Yes, by the table.

when did you pick it up?

While I was stacking up the red cube, a large red block and a large green cube.

why?

To get rid of it.

why did you do that?

To clean off the red cube.

4.4 PARRY Visits the DOCTOR

Douglas Hofstadter recounts a humourous encounter between Weizenbaum's DOCTOR and Colby's PARRY in his book [5]. I reproduce the conversation below. Note that neither of the programs say anything new, but that both of them respond to what the other says!

PARRY I don't understand your motives

DOCTOR Why don't you understand my motives?

PARRY Shouldn't I be?

DOCTOR Why do you say that just now?

PARRY That's just the way it is.

DOCTOR Suppose that's the way it were not.

PARRY So?

DOCTOR I am not sure I understand you fully.

PARRY You harp on the same questions too much.

DOCTOR What does that similarity suggest to you?

PARRY I don't get you.

DOCTOR Do you wish that you did get I?

PARRY It's about time that you showed some interest in my feelings.

DOCTOR Oh? I showed some interest in your feelings?

5 Not Much has Changed

Three decades have passed since ELIZA was created. Computers have become significantly more powerful, while storage space and memory size have increased exponentially. The field of Artificial Intelligence has blossomed, and researchers are working on problems such as image and speech recognition. Given this, we may very well wonder why no computer program has yet passed the Turing test.

At least as far as the entrants of the Loebner contest go, the complexity and capability of such systems have remained exactly where they were thirty years ago. Indeed, judges in the 1991 contest said that they felt let down after talking to the computer entrants, as they had had their expectations raised when using ELIZA. Furthermore, Kenneth Colby entered the 1991 contest with a variation of his PARRY program.

In this section I present three of the programs that entered the 1995 contest. Of these, Weintraub's PC Therapist has won the contest four times and Whalen's TIP system has won the contest once. All three of the programs have been entered in several of the Loebner contests.

5.1 PC Therapist

Joseph Weintraub spent five years coding the PC Therapist, and it consumes around five megabytes of memory. I haven't been able to find any information about how his system works, and I suspect this is because he sells it commercially. An analysis of the transcripts, however, reveal that Weintraub's program is probably nothing more than an ELIZA derivative.

Weintraub's program won the 1995 Loebner contest.

5.1.1 Example Interaction

Here I give an example of the PC Therapist talking to a judge from the 1995 Loebner contest[1]. Note that the program does keyword matching to retrieve responses from a database, as well as echoing corrupted versions of the judges input back at them. It also has an annoying habit of ending its replies with an inane question, such as "How does that make you feel?". This is probably to encourage the judge to respond to what the program said, and it rarely seems to work.

Hi judge.... i think you're cute.... want to get together later?? How does that make you feel?

I think you're a sexist pig. You should be judging me by my brains.

sex sex sex sex sex sex sex sex - there, now I got that out of my system!.Why do you need to tell me you think i'm a sexist pig?

Daniel, I know it is you. You're supposed to be fooling the judge.

I know - you just said that.Perhaps in your fantasy we know it each other.

Who is Muddy Waters?

If you only drink water, and always walk to where you are going, you will live much longer than you want to.

What does that mean?

What can't be cured must be endured.Why do you ask?

Ask what?

Do you think that men are better than woman.Does that question interest you?

No.

Please tell me something about your childhood, sometimes I wish i was a systems analyst...

Why don't you tell me something about your childhood?

Do you have many children you don't know about?Do you believe don't tell you something about my childhood?

I don't have any children I don't know about. I don't have any children. How's the weather?

To the best of your knowledge, do you have children?Don't you eally have any children you don't know about?

5.2 TIPS

Thomas Whalen developed a system called TIPS to provide information on a particular topic in response to queries in natural language. It is therefore more of a database system than any other of the programs I discuss. His program won the 1994 contest, where the domain of discourse was restricted, as his program was suited perfectly to this restriction. The 1995 contest was open-ended, however, and this provided Whalen with numerous problems.

Whalen believed the best approach was to attempt to model a human being, to the extent of including a personality, personal history and world view within the program[13]. To restrict the conversation, he created "...a character who had a fairly narrow world view; who was only marginally literate and, therefore, did not read books or newspapers; and who worked nights, and was, therefore, unable to watch prime-time television"[13].

In another clever move, Whalen created a story for the character to reveal to the judges. He hoped that the judges would be suitably interested in the story to ask questions of the program which had hardwired responses to these types of question.

I believe that Whalen's approach is more sophisticated than Weintraub's, yet his program consistently comes second place to the PC Therapist. Whalen believes that this is due to the fact that he refuses to use the non sequitur, which Weintraub's program is a master of[13]. Besides, the purpose of Weintraub's program is to simulate conversation whereas Whalen develops information systems for a particular topic.

5.2.1 Example Interaction

Here I give a section of the transcript produced by the TIPS program in the 1995 Loebner contest[1]. Note that the program performs admirably when the Judge asks questions that directly relate to Whalen's pre-prepared character and story, but that it fails miserable when asked about things outside of this area. Also not that the program introduces the story as soon as possible, similarly to Colby's PARRY.

Hello.

How are you?

Don't ask. I'm feeling kinda poor right now. It just hasn't been a good week for me.

are you sick mentally or physically?

All week I been feeling sick in my stomach on account of my boss. I got real trouble at work. I think I'm going to get fired.

are you taking pepto?

Of course not. I never stole nothing in my life. But they always blame the cleaners whenever anything is missing.

you're a cleaner?

Yeah? You think I'm a thief, too? You're no better than the rest of them. Leave me alone, asshole.

asshole?

I don't know.

you called me an asshole?

Huh?

don't you have manners. you don't even know me and you're calling me a bad name

Beats me.

anyway, where do you do your cleaning?

What do you mean?

whose office do you clean?

What?

where do you work?

At the university. I mostly clean the Standish Building.

what university?

University of Eastern Ontario. I'm a cleaner. I empty trash. Vacuum. Dust. You know.

and why do they want to fire you?

It's just because my boss, Mr. Johnson, is mad at me.

why is he mad at you?

I didn't do nothing wrong. But he thinks I did. It's all because of that radio they think I stole from the lab.

5.3 FRED

Robby Garner created the FRED program. He calls it a Functional Response Emulation Device, which apparently means that it can be used as a natural language interface to many different types of software. He has been writing this type of program for fifteen years, so one would assume this experience would count for something.

FRED is able to learn while it is talking to people. Whenever it encounters a new phrase, it responds with the closest suitable answer from it's database. At a later stage, it can be taught suitable responses to any of the new phrases it encountered[4].

FRED works by trying to match the user's input with phrases that it has seen in the past. Each phrase is associated with an appropriate response, so once a match is found that response is displayed to the user[4]. This approach has merit, and is similar conceptually to Whalen's database query program.

5.3.1 Example Interaction

Here I give an extract of the transcript for the FRED program in the 1995 Loebner contest[1].

Welcome judge 6

hello

Hello? Hello?

It's going to waste a lot of time if you repeat everything.

Rock out!

Rock on! Don't you know anything?

Cool. Do you like Nine Inch Nails? I don't want to discuss religion today.

I'm an atheist, I don't discuss religion ever. So I guess Nine Inch Nails isn't

What? Tell me what to say. You're missing out on one of life's pleasures. They have most everything on Star Trek.

Including Nine Inch Nails?

Yes.

What episode was that?

I can't remember.

Well, was it the original ST, the Second one, DS9, what?

No, he was a pirate.

Who was?

Harry Mudd, you know.

What does he have to do with 9 Inch Nails?

You know don't you?

If I knew I wouldn't ask.

Okay, I'll let you know.

Do so.

There's not that much to tell.

Good, because we don't have all day.

Yes. I always thought Beverly was a bit square.

What does beverly have to do with what we're talking about?

I think he likes some pretty strange mixed drinks.

If you're a human you're supposed to be trying to sound human

Yes.

So please do so

What about the costumes. You can tell me more.

6 How to Spout Gibberish

My first attempt at writing a conversation system was nothing to write home about. Actually, I wrote it just for fun long before I knew of the Loebner contest. I used a third order Markov chain to generate random psychobabble in much the same way as Shannon did[9].

A Markov chain may be used to construct a simple model for a language. In a third order Markov chain, there is a state corresponding to every valid word pairing in the language. Arcs connect these states to indicate valid transitions from state to state. For example, the state corresponding to the word pair ¡The, cat¿ may be connected to the state corresponding to the word pair ¡cat, sat¿. Each arc may be labelled by the word which causes the transition. In the previous example, this word is 'sat'. Finally, arcs may also have a probability of being used, and these probabilities may be estimated from a corpora of utterances.

Hence the Markov chain may be constructed by writing a program that analyses a lot of text. Once you have a Markov chain, it is easy to make it generate strings. You simply choose a state at random, and follow the arcs through the model. Each time you follow an arc, you print the word which labels the arc. If there are several arcs to choose from, you select one based on the corresponding probability.

At this stage all you will have will be a babble generator. To make the system appear to converse with the user, the reply must be in some way related to what they said. I do this by labelling certain words as keywords. All of the keywords within the user's input are extracted, and the reply is constructed in a way that maximises the number of keywords in it. There are algorithms that make it easy to generate a random reply which contains at least some of the words in a list, but I decline to give them here.

The last thing I did to write this program was to include a routine which converted a keyword to another work. For example, the word 'you' is converted to 'me'. This was necessary to ensure that the replies made more sense.

I named my finished program MegaHAL, after the HAL 9000 computer in 2001. It was the first program that I submitted to the 1996 Loebner contest, and I am fully aware that it will come last. I submitted it only as a bit of fun. Rather deviously, I used transcripts of previous Loebner contests to train this program.

I have made an early[5] version of the MegaHAL program available via the Web, and many thouosands of users have talked to it. The advantage of the program is that it learns from what the user types, so after a while the program was fluent in around six languages. The URL for the web page is: `http://ciips.ee.uwa.edu.au/ hutch/Talk.html`.

6.1 Example Interaction

Here I present an example conversation with MegaHAL. Note that the first thing it says is hardwired into the program to warn the user of what to expect. Also note that it is usually nonsensical.

Hello. I'm a bit crazy! I like to say weird things!

hello there

Hi, are you going by that?

sorry, I don't understand

You may find this a boring game?

well, I have only just started

You start taking the subject?

i would like to talk about you, if you want to

The last time with talked to you?

i have never talked with you before. tell me something about yourself

[5] By this I mean inferior!

You mean the last time I talked with you? Shut up about the trade off of oppurtunity that passes versus the happiness that my relationship if you had any claims?

hey man, what are you on?

A rich man once told me hey life's a funny thing.

i beleive that he was probably right.

You think he was with the topic?

i reckon he was.

Is he mad at you are foolish?

i don't know. hey, what's your name anyway?

Did you find that odd or offensive? Hey man, what happenned to the name wrong.

7 How to Fool the Proles

My aim for my second program is to win the Loebner contest. This means beating Weintraub, Whalen and Garner—and these guys have been building their systems for several years! Even so, I decided to limit my complete development time, including analysis, design, implementation, training, testing and debugging, to a single month. I did this to demonstrate the futility of the Loebner contest. If I can beat those other systems with a program which took only a month to make then there is something wrong with the way the contest is structured.

I began my little project on February 1, 1996, and I set the project deadline to February 29, 1996.

7.1 Analysis

The very first thing I did was obtain the transcripts to all previous Loebner contests. Not only would it enable me to analyse the other computer programs, it would give me an idea of how conversation works between two human beings. I was determined not to read any books on the subject, or download ELIZA style programs, until after my deadline. I didn't even wish to read books on English grammar, preferring to work it out myself from first principles.

I realised that previous contestants limited the conversational possibilities in some way. Whalen restricted the conversation to his little drama, Weintraub simulated whimsical conversation which can be meaningless by definition, PARRY simulated a paranoid by using non sequiturs and by being unresponsive, DOCTOR simulated a Rogerian psychotherapist by reformulating the user's statements and echoing them back, one contestant simulated a young child, and another gave his program such a slow typing speed that a judge could ask only one or two questions per session. These were all creative and devious ways of restricting what the judges could talk about. I wanted to simulate real conversation, however.

I read an aricle in Wired magazine which was written by one of the human subjects in the 1994 Loebner contest. He won the contest overall, and believed that that was because he had exhibited human emotions rather than simply answering the judges questions. He had been obnoxious, irritable and openly rude to the judges[8]. So there was one clue—give the program human emotions. Whalen had previously attempted to give his program a personality, so I would do the same. I decided that for consistency I would model the program on myself,[6] and that I would create a large supply of templated replies to give when nothing else would do, rather than outputting 'Huh?' all the time.

Another important source of information was a document written by Thomas Whalen after he had 'lost' the 1995 contest. He had a few very important points to make[13].

- One can expect that the number of topics in a standard conversation would be limited, and would follow some sort of order.[7] This wasn't the case in the Loebner contest, however, as the judges intentions are to spot the computer program. They therefore say things that no human being would every ordinarily say to a stranger.
- Judges were very reluctant to talk about topics that were volunteerd by the computer program, preferring to introduce their own.
- The judges weren't tolerant of the computer program saying "I don't know", and were much more at home with a non sequitur.
- Even though the judges agreed that Whalen's program had the most well defined personality, they did not use this as a factor for intelligence.

Even so, I decided to write the program to handle the things that people most talked about according to Carnegie. But I would also make sure it was tolerant of the weird things that judges said. The best way to do this was to study the transcripts.

I found that judges often typed several sentences at once to the computer. A sentence is very generally about a single topic, so I decided to process the input sentence-by-sentence. Sentences can be of two different types—questions or statements. I will deal with the questions first.

Against my expectations, judges almost always ended a question with a question mark. I also found that most questions were wh-questions. That is, they began with 'What', 'Where', 'Why' and so on. Other questions began with one of twenty verbs, and most of the questions concerned the judge ('I'), or the program ('you'). It was trivial to cut out the phrase following the question and use it in a template reply to give a non sequitur. In this way the question "What is a glooble?" would become, for example, "I don't know what a glooble is".

I found that the structure of statements was similarly simple.

Perhaps the most important discovery was that the single biggest giveaway that the conversant was a computer program was its tendency to repeat itself word-for-word. I vowed that my program would never do such a thing. I also

[6] Yes I know—creating a computer program in my own image does seem a bit egotistical. I'm not implying that I am God, just that the programs responses will seem more consistent if I write them all myself as if I was participating in the conversation.

[7] For example, strangers typically talk about their names, where they live, mutual friends, the weather, sports, politics, books, television, movies, music and hobbies in that order.

found that longer replies seemed more human-like that curt replies, so I would be careful to ensure that my program generated long replies.

This analysis led me to a simple design for my program.

7.2 Design

I decided that my program would parse the user's input by reading in sentences one-by-one, and that each sentence would be converted to a list of words. The program would attempt to formulate a reply to the sentence by calling modules in the following order.

1. Firstly the sentence would be checked for keywords, and a database of hardwired replies would be searched. If a suitable reply which had not yet been used was found, then it would be displayed to the user.
2. If no database match was found, my program would then try to detect whether the sentence was a trick sentence. It would look for common ways judges used to trick programs (such as asking them mathematical questions or typing nonsense words), and respond with suitably witty replies.
3. If this too failed, a MegaHAL module would be called. The reply it generated would be used if it satisfied certain criteria, such as containing at least two keywords.
4. If even MegaHAL couldn't deliver, the program would avoid the user's sentence completely by reformulating it and spurting it back to them as a non sequitur. I would use hundreds of templates to make sure this routine gave many varied responses.
5. If the program was able to give a reply, it would introduce a new subject with a certain probability. Of course, the program would have replies in the database for all subjects it introduced.
6. If the user didn't type anything, a routine would be called to give a humourous response to their silence.
7. If none of the above modules gave an answer, then the program would accuse the user of being ungrammatical or something of the kind.
8. if even this module failed to work (which could happen if all of its witticisms had already been used), then the MegaHAL routine would be invoked once more, and any babble that it generated would be used as the reply.

This seemed to be a sensible way to do things, and it allowed me to break the problem down and code each module separately.

7.3 Implementation

The coding of my program was relatively straightforward. There was no complex sentence parsing, and the most complicated module (MegaHAL) had already been written.

The only thing I was careful about was the fact that the user must be guaranteed a reply within a certain time limit. I therefore performed extensive sanity checking within all functions, and if something went wrong (such as memory not being allocated), the program wouldn't fail.

I also included a timeout, so that if a reply could not be generated within a certain time limit, then the user would be given an opportunity to type something else.

All in all, I was extremely pleased with how my program developed. It was very fault tolerant, and when I began testing it I found that it evem fooled several PhD students who knew it was a program!

7.4 Example Interaction

Here I give a conversation between a human being and the HeX program. Note that hardwired responses are given for many different types of smalltalk, and that the templated non sequiturs are slightly difficult to spot.

G'day mate.

Greetings.

Hello to you! Now, that's the formalities over!

what's your name?

My name is Jason, but you can call me Jas if you like...

my name is bob.

Pleased to meet you.

how are you doing, jas?

I am just fine thanks! And you?

i am feeling great!

If you are feeling great, it's your business, mate!

well, that's not very nice!

Well, thats not very nice I agree ... but who cares?

i care. anyway, where do you come from?

Wow - you care? Me too! I come from Perth which is in Western Australia. It's a nice place at this time of year, so it was a bit of a shock to arrive to these freezing temperatures.

it's not that cold here, is it?

We have such nice weather all year round back home. I'm not really used to really cold temperatures.

so i am speaking to an aussie?

It's cool being from Australia I guess. Gives you something to chat about when talking with a Yank!

hey, i'm not a yank!

In Australia we call all Americans yanks. It's a term of endearment actually.

that's alright then, i think...

It's nice to know that we can agree about something.

yes it is.

OK. That's good...

so, what do you do?

Well, I know what I do, but I refuse to tell you!

why?

Does there have to be a reason?

8 Why the Loebner Contest is Doomed to Failure

The Loebner contest is fated to be nothing more than an annual demonstration for the media. It will never attract a serious entry, and so it will contribute nothing substantial to our understanding of machine intelligence.[8] In this section I outline why I think that this is the case.

Competitions established with the aim of encouraging technological advances have exitsted before. Shieber gives two conditions that such competitions must meet for them to be an incentive.

1. The goals of the prize must be made clear.
2. These goals should be just beyond the reaches of modern technology. That is, the winner of the prize will be pushing the envelope within that field rather than making a drastic advancement.

Unfortunately the Loebner contest meets neither of these criteria. It is therefore nothing more than an amusing game, attracting entries that address the test in the short term by engineering rather than science[10].

Turing's imitation game in general is inadequate as a test of intelligence as it relies solely on the ability to fool people, and this can be very easy to achieve, as Weizenbaum found[7].

My other objections to the Loebner prize are as follows.

- Restricting the domain of discourse renders the test meaningless. Fortunately the 1996 test will not have this restriction.

[8] This is, of course, just my humble opinion. I do not intend to vilify Dr. Loebner in the slightest. In fact, I have great respect for the man. I think that his contest does provide a challenge to people like me, but not to people working on "serious" AI.

- Each judge has only a limited time to converse with each subject. This may not allow the judge to make a confident decision.
- The fact that the judges can see the messages as they are being typed violates Turing's original idea. No clues to the subjects identity should be given apart from the linguistic clues. Unfortunately I must simulate typing speed, pauses for thought, typing errors, and so on to have a chance of winning.
- Loebner changed the contest rules to state that the grand prize winner must be able to cope with audio/visual input, but this capability is way beyond state-of-the-art. This effectively removes the main incentive for entering the contest.

The Loebner contest can be fun, and it was certainly interesting designing and coding my two entries. I am only entering for the fame and fortune however, as there is very little research potential in entering such a contest. A program which could pass the original Turing test would be a very valuable piece of software, but it won't get written as a result of the Loebner contest.

9 Conclusion

In this paper I have discussed the Loebner contest, and analyzed various conversation systems. I also described the design and implementation of my two entries to the 1996 contest.

I finished by badmouthing the Loebner contest, but this doesn't mean I won't enter it again. On the other hand, I have very ambitious plans for my 1997 entry. I don't think that anyone is going to create an intelligent machine any time soon, but the Loebner contest may just stimulate a few advances in the field of natural language interfaces to database engines. We can only wait and see ...

On an amusing sidenote, Marvin Minsky posted a message on an AI newsgroup offering a $100 prize to anyone who could stop Loebner from holding the contest, calling it an "obnoxious and unproductive annual publicity campaign". Loebner responded that the grand prize winner would effectively cause the cessation of the contest, and that Minsky would be morally bound to pay the winner the $100. Loebner therefore instigated Minsky as a co-sponser of the contest.

References

[1] 1995 Loebner contest transcripts. Available at http://www.acm.org/ loebner/loebner-prize-1995.html.

[2] Kenneth Mark Colby. Modeling a paranoid mind. *The Behavioural and Brain Sciences*, 4:515–560, 1981.

[3] Robert Epstein. The quest for the thinking computer. *AI Magazine*, pages 80–95, 1992.

[4] Robby Garner. The idea of FRED. Available at http://www.diemme.it/ luigi/fred.html.

[5] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid.* Penguin Books, 1979.

[6] Hugh Loebner. In response to Shieber. Available at http://www.acm.org/ loebner/In-response.html.

[7] James H. Moor. An analysis of the Turing test. *Philosophical Studies*, 30:249–257, 1976.

[8] Charles Platt. What's it mean to be human anyway? *Wired*, pages 133–, April 1995.

[9] Claude E. Shannon and Warren Weaver. *The Mathematical theory of Communication.* University of Illinois Press, 1949.

[10] Stuart M. Shieber. Lessons from a restricted turing test. Available at http://xxx.lanl.gov/abs/cmp-lg/9404002, March 1994.

[11] A.M. Turing. Computing machinery and intelligence. In D.C. Ince, editor, *Collected works of A.M. Turing : Mechanical Intelligence*, pages 133–160. Elsevier Science Publishers B.V., 1992.

[12] Joseph Weizenbaum. *Computer Power and Human Reason.* W.H. Freeman and Company, 1976.

[13] Thomas Whalen. My experience at Loebner prize. Available at http://www.diemme.it/ luigi/whal2812.html.

[14] Terry Winograd. *Understanding Natural Language.* Academic Press, 1972.