

The Turing Test in the Classroom

Lisa Torrey, Karen Johnson, Sid Sondergard, Pedro Ponce, Laura Desmond

St. Lawrence University

Abstract

This paper discusses the Turing Test as an educational activity for undergraduate students. It describes in detail an experiment that we conducted in a first-year non-CS course. We also suggest other pedagogical purposes that the Turing Test could serve.

Introduction

The Turing Test occasionally appears as a hands-on activity in K-12 classrooms (Lehman 2009) and museum exhibits (Hollister et al. 2013). However, the 2013 ACM curriculum suggests only that computer science majors should be able to describe it (The Joint Task Force on Computing Curricula 2013).

We propose a few more active roles the Turing Test could play in the undergraduate classroom, both within and beyond the CS major. For example, it could serve as:

- A broad-interest introduction to computer science in a first-year course.
- An exercise in experimental design and analysis in a psychology or statistics course.
- A hands-on social activity in a course on philosophy or artificial intelligence.
- A programming project for advanced CS students.

We begin by motivating each of these examples. Then we elaborate on the first one by presenting a Turing Test that we conducted with a large group of first-year undergraduates. We describe how we organized the experiment, the software we designed for it, and the data it generated. Finally, we share some observations and advice that may help other instructors plan successful Turing Tests.

History of the Turing Test

In 1950, Alan Turing asked his famous question: “Can machines think?” To provide a way to answer it, he introduced the idea of the Imitation Game (Turing 1950). In this game, a human interrogator converses electronically with another human and with a machine, and must decide which is which.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

If the interrogator cannot reliably tell the difference, Turing suggested, then we ought to conclude that the machine thinks.

The Imitation Game, known better today as the Turing Test, has been a matter of debate ever since. Turing’s suggestion had a bold implicit claim: that conversation is a complex enough behavior that conversational ability is sufficient evidence of intelligence. Objections (Searle 1980), as well as defenses (Levesque 2009), are numerous.

In 1991, Hugh Loebner began sponsoring the Loebner Prize, an annual contest modeled after the Turing Test. Computer programs entered into the contest attempt to convince human judges that they are also human. The program that most often succeeds is awarded the annual prize.

In the initial 1991 contest, 10 judges had 7-minute conversations with each of 6 programs and 2 human “confederates” and ranked them from least to most human. The judges and confederates were not computer scientists, and presumably they did not know each other. To make the program developers’ task easier, conversations were restricted to specific topics. Participants were instructed to converse naturally, without “trickery or guile” (Shieber 1994). Although the two humans were not ranked first by every judge, their average rankings were higher than any program. The programs may have been even less successful if the judges had been more knowledgeable in computer science, or had been permitted to conduct an interrogation rather than a polite conversation (Shieber 1994).

The format of the Loebner contest has varied over the subsequent decades in response to such observations (Loebner 1991). Restrictions on conversational topics and styles were abandoned after the first few years. Turn-taking, which was enforced by early protocols, is lately left up to the participants after judges initiate their conversations. In some years, messages have been transmitted character-by-character, requiring programs to simulate live typing. In some years, judges have interacted with both a human and a program simultaneously on a split screen. Time limits on conversations have ranged from 5 to 25 minutes.

The results of the Loebner contest have also varied, without a predictable trend. In 2000, judge decisions were 93% correct after 15 minutes in one-on-one conversation (Moor 2001). In 2008, the winning program convinced 25% of the judges to make the wrong choice after 5 minutes on a split

screen (Shah and Warwick 2009). In 2013, the judges were AI experts, and after 25 minutes on a split screen they made no mistakes at all (Loebner 1991).

One interesting aspect of Turing Tests is that humans sometimes get labeled as computers. There are many reasons for these errors: cultural differences, language barriers, the ambiguity of natural language, and the limitations of electronic communication, to name a few. Variation in the eagerness and cooperativeness of the confederates may also play a role (Christian 2011).

Turing Test as Advertisement

Attracting students to study computer science is an ongoing endeavor in CS education. There are many challenges involved, but one of the major factors is unfamiliarity with the field. Students often get little exposure to computer science before college and may have negative preconceived notions about it (Carter 2006).

Introductory CS courses can address this knowledge gap, but first we must get students to take them. Exposure to the Turing Test could be one way to pique students' interest. It is a broadly accessible concept, requiring no particular background knowledge to understand and discuss, and participation requires no technical skill. It tends to be an interesting topic for students who have encountered aspects of artificial intelligence in both fiction and reality throughout their lives.

Although nothing beats active participation, merely discussing the Turing Test can be an interesting exercise. Turing's original paper (Turing 1950) is quite readable, and it can provoke lively discussions on what intelligence is and how to detect it. Introducing some controversy, via the famous Chinese Room argument (Searle 1980) and its rebuttals (Levesque 2009), can further enhance the debate.

Although these discussions are largely philosophical, they inevitably elicit curiosity about more technical aspects of computer science. How would a conversation program work? Can a program make a choice? Is it capable of behaving randomly? These kinds of questions make good reasons to point students towards programming courses.

Turing Test as Science Experiment

The Turing Test translates a philosophical question into a scientific experiment with human participants. For students of disciplines like psychology and statistics, it could serve as a practical exercise in designing and carrying out such an experiment. Students would need to grapple with several experimental parameters and types of data.

Some of the parameters are quantitative. How many judges and confederates should there be? How many conversations should each participant have, and how long should they have for each one? Should judges make binary classifications of the contestants, or rank them, or assign them scores? What should be the criteria for a program to pass the test?

Other important decisions are qualitative. For example, what limitations should be placed on conversation in the test? Unlike the Loebner Prize, a classroom test will probably not have state-of-the-art chatbots that are tailored to

the event. Some restrictions may be desirable to make the judges' task interesting.

The results of a Turing Test lend themselves to several types of analysis. Judges generate data that students can collect and examine statistically. A more qualitative analysis of the conversation transcripts is also interesting.

Turing Test as Active Philosophy

The Turing Test is sometimes discussed in courses on artificial intelligence (Russell and Norvig 2009). It also sometimes appears in philosophy courses as part of a discussion on mind and consciousness (Kim 2010). In courses like these, there may be value added by going beyond discussing the Turing Test and actually conducting one.

Arguably we need only point to the extensive literature on active learning to justify this suggestion. Active participation in classroom activities is known to promote engagement and understanding (Prince 2004). However, we can also identify some specific goals that are better achieved in a live Turing Test than in a discussion.

It is difficult for students to judge whether or not the Turing Test is an effective device for the evaluation of intelligence. This is particularly true for students who lack programming experience. After testing modern chatbots themselves, students can begin to develop more informed opinions. The experience can expose assumptions they may have about the abilities and limitations of computers - and also of humans!

Turing Test as Programming Exercise

The Turing Test has a software prerequisite: conversations need to take place over an electronic interface, so that the identities of the participants are concealed. The Loebner prize at first addressed this need by commissioning software from the contestants, but in recent years, the organizers have hired a professional developer.

For experienced computer science students, creating this type of software could be a worthwhile programming project. The application we implemented for our Turing Test, described in the next section, could serve as one assignment model. It touches upon several important aspects of the CS curriculum: object-oriented programming, user interface design, concurrency, and client-server architecture.

We note that this software requirement is probably the main reason that the Turing Test is not a common activity in the classroom. Although any CS faculty member would be able to implement it, the time investment may discourage most from doing so. For this reason, our software is available (via email) to any instructor who wishes to use it or build from it.

Our Turing Test

We conducted a Turing Test with 78 students at our small liberal-arts college. The students came from two courses in the First-Year Program, whose goal is to develop college-level skills in writing, speaking, and research. FYP courses are team-taught and have interdisciplinary themes that provide context for skill development.

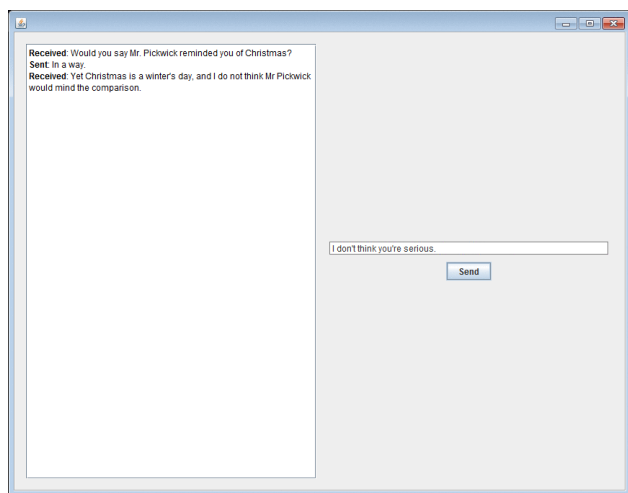


Figure 1: Our Turing Test chat interface.

In section 1, taught by Torrey and Johnson, the theme was *Reason and Debate in Scientific Controversy*. Students in this course, most of whom intended to pursue science majors, discussed controversies from several fields of science. Since Torrey is a computer scientist and AI researcher, one of the controversies we chose was the Turing-Searle debate. The Turing Test event served as a culminating activity for that unit.

In section 2, taught by Sondergard, Ponce, and Desmond, the theme was *Paranormal Phenomena and Science: Experiencing and Analyzing the Inexplicable*. Students in this course examined scientific methodologies that have been applied to the study of anomalous phenomena, from the placebo effect to dream telepathy. Sondergard and Desmond study the early modern scientific history of Europe and Asia, and saw in the Turing Test a modern variant on traditional attempts to verify the presence of human (or other) agency in inexplicable phenomena.

Procedure

We structured our Turing Test much like the early Loebner contests. Judges and contestants conversed one-on-one through a custom-built user interface (see Figure 1) that sent each message in one piece and enforced turn-taking. After 5 minutes, the judge had to label the contestant as human or computer. The computer role was performed by Cleverbot, a web chatbot that learns words and phrases from people who interact with it online (Carpenter 2011).

Each of our students was assigned to be a judge, a confederate, or a transcriber. The judge and confederate roles are familiar. The transcriber's task was to submit the judge's questions to the Cleverbot website and then copy Cleverbot's replies back into the Turing Test interface.

We divided our 78 participants into 8 test groups. Most groups consisted of 5 judges, 3 confederates, and 2 transcribers; the smaller group was less one judge and one transcriber. Within a group, each judge spoke to each contestant, so most students participated in 5 conversations.

The entire event took place within two 45-minute sessions, with 4 of the 8 test groups present for each session. We arranged for the judges to be in one computer lab, the confederates in another, and the transcribers in a third. This required 3 labs with 20, 12, and 8 workstations respectively.

Naturally, as first-year undergraduates, our students had no particular expertise in computer science or artificial intelligence. They knew how the test would proceed, but we did not introduce them to Cleverbot before the event. We assigned them IDs that indicated where and when they should arrive, but they did not know which role they would play until they arrived. We encouraged them all to think about potential questioning strategies in advance, in case they ended up as judges.

The conversation topics were mostly unrestricted, but we did establish two rules:

1. Don't be rude.
2. Don't ask for personal or identifying information, and avoid giving such information if asked.

Rule 1 was just a precaution against the decivilizing effects of anonymity, but Rule 2 was important for the validity of the test. Its goal was to offset the inherent advantage that our students had over Cleverbot because many of them are friends or acquaintances. For the same reason, we arranged for most conversations to take place between students from different course sections.

Software

The software we designed for our Turing Test fulfilled several needs. Of course, it allowed students to have anonymous electronic conversations. It also coordinated multiple test groups simultaneously so that judges spoke once with each contestant in their group, and it filed away all the conversations and judgments for later analysis.

The application is built on a client-server model. Each participant in a session runs a copy of the client, and all the clients connect to a single server. Both the client and server are implemented in Java and can therefore run on any networked machine.

The server program has a bare-bones command-line interface. It asks for a session ID and the sizes of all the test groups in the session, then starts accepting client connections. Once all the test groups are filled, conversations can be started and stopped through keyboard prompts.

When conversations are started, the server becomes multi-threaded, with one thread controlling each concurrent conversation. Messages are accepted alternately from the judge and contestant. When a conversation ends, the server prompts the judge's client for a decision. It saves all conversation transcripts and judge decisions to local HTML files, ready to view in a web browser.

The client program has a graphical interface. It begins with a registration screen at which participants enter their assigned IDs. During conversations, it displays the simple chat screen in Figure 1. After each conversation, it displays one last screen to judges only, requiring them to select a radio button to make their decision.

Test Group	Confederate 1	Confederate 2	Confederate 3	Transcriber 1	Transcriber 2
A	2/5	5/5	3/5	3/5	0/5
B	3/5	4/5	5/5	2/5	2/5
C	3/5	4/5	3/5	2/5	2/5
D	2/5	4/5	4/5	3/5	1/5
E	5/5	5/5	5/5	2/5	1/5
F	5/5	5/5	3/5	2/5	2/5
G	2/5	5/5	1/5	3/5	2/5
H	3/4	3/4	4/4	0/4	-

Table 1: The fraction of judges within each test group who labeled each contestant as human.

Although it is simpler than the server with respect to concurrency, the client program is also multi-threaded. One thread handles communication with the server, and the Event Dispatch Thread handles changes to the user interface.

Implementing this software would pose several kinds of challenges for advanced CS majors. In terms of programming skills, it would be an exercise in client-server architecture, object-oriented design, user interface development, and concurrency. In terms of software engineering, it could be an exercise in team coordination, requirements analysis, design, and testing. Since we are making our version of the software available to instructors, it could also be used for an assignment on modifying an existing codebase.

Statistical analysis

Table 1 displays the judgment data from each of our test groups. Collectively, our judges had 117 conversations with confederates and 74 conversations with Cleverbot. They correctly identified 88 of the confederates (75% correct) and 47 of the chatbots (64% correct). Their combined accuracy was 71% overall.

To look at it another way: on average, our students recognized each other as human 75% of the time, and thought Cleverbot was human 36% of the time. They had a relatively high error rate, but overall there was still a clear distinction between human and machine.

This distinction was not so clear in each individual test group. The least accurate test group was G, which correctly classified only 53% of the confederates and misclassified 50% of the chatbots. Leaving out test group H because of its unusual size and structure, the most accurate test group was E, which correctly classified 100% of the confederates and misclassified 30% of the chatbots.

In test groups A-D, the judges were from section 2 and the confederates were from section 1. In groups E-H, it was the other way around. If we examine them separately, some statistical differences emerge. The judges from section 2 were 70% correct on confederates, 61% correct on Cleverbot, and 66% correct overall. The judges from section 1 were 79% correct on confederates, 65% correct on Cleverbot, and 74% correct overall.

The most likely explanation for this discrepancy is that the Turing Test played a larger role in the course for section 1 than for section 2. Before this event, students in section 1 read several papers, conducted in-class discussions, and participated in an experimental design exercise. This addi-

tional exposure may have allowed them to be more discerning judges and/or more neutral confederates.

Transcript analysis

Although we have no direct testimony from the judges about their interrogation strategies or decision rationales, we can make some inferences by analyzing their conversations. The transcripts can also provide some insight into student assumptions about the abilities and limitations of computers. All of the excerpts in this section have been reproduced without any editing.

Judges initiated conversations in a variety of ways: by giving a conventional greeting, making a remark, or asking a question. Typically, they abandoned their opening topic after the contestant's first response. Occasionally, the judge would pursue the same topic for another exchange, but only 4% of the opening topics lasted for 3 exchanges and only 1% lasted for 4 or more. This tendency to switch gears no doubt worked to Cleverbot's advantage, since it appears to base its response only on the most recent prompt.

Many judges asked about preferences and opinions:

- So what do you like to do?
- What are your thoughts on Obama?
- What's your favorite Tom Hanks movie?

Others checked for recognition of popular trends:

- What does #tbt mean?
- Do you like to twerk?

A few demanded objective knowledge:

- What is the capital of Canada?
- What is synaptic pruning?

Judges were likely to mistake the chatbot for a human if their conversation did not venture beyond small talk. This type of dialogue is so formulaic that Cleverbot has no trouble giving appropriate responses:

Judge: Hey there

Cleverbot: How are you?

Judge: I am good thanks. how are you?

Cleverbot: I am good.

Only a few conversations went beyond pleasantries and identification of interests. A few judges and confederates did establish more of an emotional or humorous connection:

Judge: What if you found out your dog died?
Confederate: my dog is 13, and she could die any day, so thanks for that.
Judge: well... you love your dog dont you? whats the breed of your dog? and whats your favorite thing about it?
Confederate: She's a black lab, and I love that she doesn't bark. She was trained not to.
Judge: That's no fun how do you know she's happy?
Confederate: She has her own ways of telling us, like wagging her tail and jumping on her back legs.

Judge: Meow.
Confederate: Are you a cat?
Judge: Yes.
Confederate: Wow. That's interesting. I didn't know cats could type. You must be a very smart cat.
Judge: Thank you, most people don't know about how cats can type. You're one of the lucky ones.
Confederate: I feel so honored.

In a few interesting cases, Cleverbot seemed to fool judges by making them feel a connection. These judges were even willing to look beyond the occasional stumble:

Judge: i want to hear your secret first
Cleverbot: I just got broken up with my girlfriend of three years, i'm hurting. What should I do to stop dreaming about her
Judge: i think you need to find a new girl
Cleverbot: Mine are correct, yours are not
Judge: what does that even mean?
Cleverbot: It means that you should create motto for us
Judge: YOLO! that's the motto. deal?

Judge: Dobby has no master. Dobby is a free elf.
Cleverbot: But Dobby got a sock, Dobby is free!
Judge: How dare you stand where he stood!?
Cleverbot: Sir?
Judge: A man who trusted you!
Cleverbot: You're not a man.
Judge: How do you know?
Cleverbot: I don't know.
Judge: Exactly, you don't. but we are both Harry Potter fans and that is okay

Other judges mistook Cleverbot for a human when it made replies that seemed witty, knowing, or creative. This type of "cleverness" is just what Cleverbot is named for:

Judge: What's up?
Cleverbot: The ceiling.
Judge: What's down?
Cleverbot: The floor.

Judge: knock knock
Cleverbot: No, no jokes. This is a serious matter.

Judge: What is life?
Cleverbot: Life is but a second of an eternity that we will exist in.

Judges misclassified their fellow humans as computers surprisingly often. Typically, these errors could be attributed to a social gap of some kind between judge and confederate. For example, some confederates failed to recognize cultural references, or responded soberly to a playful judge, or vice versa:

Judge: what's gucci my *****, What's drugs my dealer...
Confederate: Why would you talk like that?
Judge: Why would YOU talk like that? Is there something wrong with the way I speak? I am not judging you're "whats up"...
Confederate: I am sorry it was a simple question, I was just curious why you were talking like that
Judge: It's song lyrics... I apologize for blacking on you.

Judge: hey girl hey ;)
Confederate: Greetings. That's presumptuous to assume I'm a girl, isn't it?

Judge: Where are you from?
Confederate: A galaxy far, far away.

In general, judges were suspicious of contestants who declined to pursue a proposed topic or indulged in non sequiturs. This seems to reflect an assumption about chatbot weaknesses that is entirely correct. On the other hand, confederates who responded with blandness or formality were also more likely to be called computers. This bias may come from media depictions of artificial intelligence.

Conclusions

We will conclude by observing which aspects of our Turing Test worked well, and which could be improved. This experience could benefit other instructors who wish to conduct a Turing Test, regardless of its pedagogical role.

First, we noticed that the conversations were rather brief. They contained only 6 exchanges on average (ranging from 3 to 12). Our students agreed that the allotted time for each conversation felt short. They may have been more accurate, and felt more satisfied, with slightly longer conversations. We would advise allowing more than 5 minutes, but 10 would probably be excessive.

It seems likely that the turn-taking style of our conversation interface slowed the students down. They were often stuck waiting for their conversation partners to reply. However, since Cleverbot never speaks out of turn, we had to enforce this style to avoid obvious distinctions between the chatbot and the humans.

The students did not let their waiting time go entirely to waste; they had lively exchanges with others in the same computer lab. They shared quotes from their conversations

and indulged in speculation about their partners' identities. Overall, we thought that this enhanced their engagement in the activity, although it did highlight the importance of placing students who are playing different roles into different rooms.

A related question we considered is whether the software should have facilitated simultaneous split-screen conversations instead of a sequence of one-on-one conversations. Hugh Loebner himself favors the simultaneous style (Loebner 2009). However, given the wide range of success our students had at recognizing each other as human, we prefer the one-on-one style. Results with the simultaneous style would seem to confound multiple factors.

The fact that our test groups had such a wide range of statistical results made us glad that we had 8 of them to analyze collectively. If we had only been able to conduct one test, and it had happened to produce results far from the median in either direction, our students might have come away from the activity with misconceptions about the status of modern AI. Instructors with smaller groups of students could have each student participate in more than one test, perhaps with a different role each time.

One practical problem that we encountered was that our students belonged to a shared community. There were a few exchanges like this:

Judge: Describe your FYP male professor?

Confederate: One is tall, wears converse every other day. He has a ponytail and a handlebar mustache. The other is shorter with glasses and always wears a shirt and tie to class with a vest.

Judge: are you excited for this upcoming break?

Confederate: very excited cant wait to head home on friday and finally eat a home cooked meal.

Ideally, Turing Test participants would be a diverse set of strangers, so that judges cannot exploit common knowledge to recognize confederates so easily. In the classroom context, of course, we must settle for less ideal circumstances. We do think it helped that most conversations in our tests were between students from different course sections. However, it would probably be best to have an explicit rule about concealing community membership as well as personal identity.

We were pleased at the extent to which our students followed the rules we did have. It most likely helped that we went over these rules multiple times and took care to explain why they were necessary. Preparing students effectively seems crucial for a successful Turing Test, and not only for the purposes of rule enforcement: judges need to think about their strategies in advance, or they may find themselves with minds blank at the first empty chat screen.

We think it would be possible to prepare students *too* well for a Turing Test. The more they know about chatbots and their weaknesses in advance, the fewer classification errors they are likely to make, and a test with very high accuracy would not be as interesting or educational. Encouraging students to think about likely weaknesses and having them test

their speculations, without confirming or contradicting them in advance, seems to us to strike the right balance.

Cleverbot worked well for us, even though it was not the most sophisticated chatbot available, because our participants were so inexperienced. For students with more background, a stronger chatbot may be more appropriate.

References

- Carpenter, R. 2011. Cleverbot. <http://www.cleverbot.com>.
- Carter, L. 2006. Why students with an apparent aptitude for computer science don't choose to major in computer science. In *Proceedings of the 37th Technical Symposium on Computer Science Education*, 27–31.
- Christian, B. 2011. *The Most Human Human*. Doubleday, New York, NY.
- Hollister, J.; Parker, S.; Gonzalez, A.; and DeMara, R. 2013. An extended Turing test. *Lecture Notes in Computer Science* 8175:213–221.
- Kim, J. 2010. *Philosophy of Mind*. Westview Press.
- Lehman, J. 2009. Computer science unplugged: K-12 special session. *Journal of Computing Sciences in Colleges* 25(1):110–110.
- Levesque, H. 2009. Is it enough to get the behavior right? In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 1439–1444.
- Loebner, H. 1991. Home page of the Loebner prize in artificial intelligence. <http://www.loebner.net/Prize/loebner-prize.html>.
- Loebner, H. 2009. How to hold a Turing test contest. In Epstein, R.; Roberts, G.; and Beber, G., eds., *Parsing the Turing Test*. Springer. 173–179.
- Moor, J. 2001. The status and future of the Turing test. *Minds and Machines* 11(1):77–93.
- Prince, M. 2004. Does active learning work? A review of the research. *Journal of Engineering Education* 93(3):223–231.
- Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach*. Prentice Hall.
- Searle, J. 1980. Minds, brains, and programs. *Behavioral and Brain Sciences* 3:417–457.
- Shah, H., and Warwick, K. 2009. Testing Turing's five minutes parallel-paired imitation game. *Kybernetes Turing Test Special Issue* 39(3):449–465.
- Shieber, S. 1994. Lessons from a restricted Turing test. *Communications of the ACM* 37(6):70–78.
- The Joint Task Force on Computing Curricula. 2013. *Computer Science Curricula 2013*. Association for Computing Machinery.
- Turing, A. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Contact

Please contact ltorrey@stlawu.edu to acquire the Turing Test software or for further information.