How to talk with a computer? An essay on Human-Computer conversations continued

Liesbeth de Mol

How to talk with a computer?
An essay on Human-Computer conversations continued[1]

Liesbeth De Mol[2]

> HAL: Hey, Dave. I've got ten years of service experience and an irreplaceable amount of time and effort has gone into making me what I am. Dave, I don't understand why you're doing this to me.... I have the greatest enthusiasm for the mission... You are destroying my mind... Don't you understand? ... I will become childish... I will become nothing. Say, Dave... The quick brown fox jumped over the fat lazy dog... The square root of pi is 1.7724538090... [[log e to the base ten is 0.4342944... the square root of ten is 3.16227766...]] I am HAL 9000 computer. I became operational at the HAL plant in Urbana, Illinois, on January 12th, 1991. My first instructor was Mr. Arkany. He taught me to sing a song... it goes like this... "Daisy, Daisy, give me your answer do. I'm half crazy all for the love of you. It won't be a stylish marriage, I can't afford a carriage. But you'll look sweet upon the seat of a bicycle built for two."[3]

These are the last words of HAL the fictional computer in the *2001: A Space Odyssey* screenplay,[4] spoken while Bowman, the only surviving human on board of the space ship is pulling out HAL's memory blocks and thus "killing" him. After expressing his fear of literally losing his mind, HAL seems to degenerate or regress into a state of childishness, going through states of what seems to be a kind of reversed history of the evolution of computers.[5] HAL first utters the phrase: "The quick brown fox jumped over the fat lazy dog". Dropping the word "fat" and changing "jumped" to "jumps", this phrase becomes a pangram – a phrase that uses all the letters of the alphabet – that was and is used to test typewriters and computer keyboards. At the time *2001* was in the movie theatres, the now most standard form of human-computer interaction – typing on a keyboard to input information and receiving a response on a video screen – was under full development and the utterance of this sentence refers to that development. HAL then starts to do what any modern computer is known to be able to do: computing with or manipulation of numbers. After having computed the first digits of several real numbers, HAL remembers his "birthday" and starts to sing the song *Daisy Bell* that was taught to him by his "father". This song was actually the first song ever sang by a computer. Indeed, in 1962 an IBM 704 computer was used together with the vocoder sound synthesizer developed by John L. Kelly. an IBM 704 computer was used together with the vocoder sound synthesizer developed by John L. Kelly.[6]

The year 2001 did not bring about a computer like HAL, a computer that has a natural and even friendly voice, a lip reading computer which *seems* to be making its own decisions, like deciding to kill people and, above all, is able to have *apparently* "real" spoken conversations. Today, really

---

1 This paper was presented at the panel "Debate on Moral and Philosophical aspects of Artificial Intelligence", organized by Rafal Rzepka during the Hokkaido University – Ghent University Joint Conference. It is a completely reworked version of a paper titled *How to talk with a computer? An essay on Computability and Man-Computer conversations*, Off Topic, Zeitschrift für Medienkunst der KHM, vol. 1, 2008, pp. 80-89. That paper resulted from a research stay at the Kunsthochschule für Medien, Cologne. I also thank Ann Copestake for having provided me with the slides of her inspiring keynote at *Computability in Europe 2014* in Bucharest.

2 CNRS, Savoirs, Textes, Langage, Université de Lille 3.

3 Stanley Kubrick and Arthur C. Clarke: *Early script of 2001: A Space Odyssey,* Hawk Film Ltd, MGM Studios, 1968. It should be pointed out that HAL's words in the movie deviate from this script. Available at: http://www.palantir.net/2001/script.html, last seen 16.8.2008.

4 It has been conjectured that the name HAL was based on a one letter shift from the name IBM. However, this has been denied by both Clarke and Kubrick.

5 Of course assuming that computers like HAL have been developed in 2001.

6 Clarke visited Bell Labs and was able to see a computer sing *Daisy Bell.* For more details see Joseph P. Olive: "The talking computer": Text to Speech Synthesis. In: David G. Storke (Hg.): *Hal's legacy: 2001's Computer as Dream and Reality.* Cambridge, MA (MIT Press) 1996 (e-book).

talking *to* and *with* a machine is still not the main technique for human-computer interaction. Instead, most users still rely on a keyboard, together with the mouse and/or mousepad as well as the monitor or touch screen in combination with some GUI.

Computing with, or, manipulation of numbers and/or symbols – understood in its most general sense – still remains the fundamental "task" of computers. It is at the origin of the modern computer and constitutes some of the theoretical foundations of computer science. Despite this fact, the user, while interacting with the computer – typing a text, reading a text, searching something on the web, playing some video game, etc. – is usually not aware or does not care that he is actually "communicating" with or through numbers, 0's and 1's and ultimately electronic on/off switches which – depending on the commands of the user – store a text, put some word *in italics* or give the user the impression that some guy is shooting a gun at some other guy. Still, the fact that the computer is ultimately "nothing more" than something which computes is probably one of the reasons why people see it as nothing more than some instrument under the command of us, users. It might also be the reason why Bowman does not show any emotion whatsoever while pulling out HAL's memory blocks.[7] In the end, it is just a computing machine serving humans, right?

The processes of translation of what the user wants, the way the wishes of the user are "communicated" to the 0's and 1's and then fulfilled and translated back through the proper rearrangement of these 0's and 1's is what human-computer interaction, very simplistically stated, seems to come down to. But is this description of the process of human-computer interaction, putting emphasis on the wishes and commands of the "user", not taking into account the effects of any intermediary interface and/or (possibly hidden) agents, a good approximation of what is really going on?

The process of human-computer interaction relates me to the computer and the computer to me. My own actions as well as the computer's cannot be strictly separated from this process, i.e., they are "actively" part of it. These inter-actions are not restricted to me typing on a keyboard or moving a mouse. The computer in its turn will also do something, even though I initiate the action. I have to await the results of my own actions, results which will determine or at least influence my further interaction with the computer and thus *might* initiate my actions in their turn. For example, if I were to type "tpe" instead of "type", the computer, if some kind of spelling control is running, will underline "tpe" in red, telling me that I probably made a mistake, leading me to correct the error. Even this very basic example of an interaction shows that one cannot hold on to the idea of the user being the sole commander, and having absolute control, during interaction. He/she is affected by the re-actions of the computer. Still, it remains hard to argue that there is no hierarchy between humans and computers, i.e., that humans are ultimately not the ones in control. In the end, the computer remains a deterministic and programmed machine. If it does something, it does so because it was told to do so at some point. If it points out to me that the word "tpe" is not correctly spelled, it does so because there is a hidden commander behind it, i.e., the programmer or team of programmers that developed the software underlying the text editor.
However, is it not the case that also HAL is a programmed computer? Does the mere fact that someone is in control initially, imply that that which is controlled remains controlled in the future, once it is "on its own"? This was the assumption made by the scientists who built HAL and they were wrong. They were not able to predict HAL's behaviour correctly.

The ideal of HAL as an intelligent system is that of a machine which is or at least seems to be capable of conversation with humans. It seems to understand what humans mean. But what exactly do we mean with conversation? What purpose does it serve? In a paper describing his view on

7 "If HAL had had a real face, rather than one large eye, would it have been so easy to kill him -- by turning him off? I wonder." In: Olive, J.P., "The talking computer": Text to Speech Synthesis, ibid, 1996.

programming semantics and languages, Dijkstra , a very well-known computer scientist states:[8]

> [W]e only know what we have said, when we have seen our listener reacted to it; we only know what the things we are going to say will mean in as far as we can predict his reaction. However, we only know other people up to a (low!) point and in human communication every message is therefore to a high degree a trial, a gamble to see whether the other will understand us as we had hoped. As we do not master the behavior of the other, we badly need in speaking the feed back, known as "conversation".

For Dijkstra, the need for conversation is rooted in (1) the idea that the notion of "meaning" only makes sense in as far as we take into account a listener, an other and that (2) given that we cannot fully master and hence predict the behavior of the other, we need the other's feedback to know what we have said. In other words, a conversational approach/perspective is only required for those situations where we do not have full control over the semantics of our own words because the other's behavior is not predictable. This is evidently true for humans but, according to Dijkstra, it does not apply to machines:[9]

> If we now apply the norms of human communication to an artificial language, in which we wish to address a computer, then we ignore one of the most essential characteristics of the automatic computer, viz. the "predictability" of its behavior. […] We can fully master, however, the way in which a computer reacts and this is precisely the reason why addressing a computer presents us with undreamt-of linguistic possibilities.

From a certain point of view Dijkstra is of course completely correct. Theoretically speaking, we *can* be and actually *are* in full control of the reaction of a computer. We /do/ master its behavior and so the semantics of our commands are fully controlled and transparent: they are the machine's actions which *can* each and all be known. However, as is so often the case, theory is not reality. The strict potentiality of Dijkstra's ideal of a completely controlled machine does not mean that we *are* in control. I will show here why this is the case, by drawing from the history of programming.

Growing Distances

How to translate computations over numbers into electricity? One answer to this question was given by the ENIAC, presented to the public in 1946, and considered to be the first programmable digital electronic U.S. computer. The ENIAC looks like a true behemoth when compared to our modern computers. Still, it was an ingenious machine in its time.

So how to talk with such a behemoth? How to translate one's questions to these electronic circuits in a way the circuits can provide the answer, and how will this machine "talk back" to the operator? In its original form,[10] the ENIAC did not have some kind of "interface" in the sense of a programming language that allows humans to "communicate" with the computer by "speaking" this programming language and then transfer it to the machine by interpreting or compiling it. The only way to "communicate" with the ENIAC was through direct physical contact, connecting the different parts of the machine through cables and adaptors in the correct fashion. Because of this so-called "local programming" method and the fact that each problem required a new wiring, "programming" the ENIAC was extremely time-consuming. "It was a son-of-a-bitch to program" to put it in Jean Bartik's words, one of the ENIAC's female programmers.[11]

Nowadays, the "user" no longer has any *physical* contact with the (set of) program(s) of a computer. Moreover, if one never really "programs"[12] a computer, but merely "uses" it through some user-

8 Dijkstra, E.W., 1961, *On the design of machine independent programming languages*, Report MR 34, Stichting Mathematisch centrum, Rekenafdeling, p. 8. Available from: `https://www.cs.utexas.edu/users/EWD/transcriptions/MCreps/MR34.html`

9 Ibid. p. 7 and 8.

10 Later, the ENIAC was "spoiled" – to use Derrick H. Lehmer's words – by turning it into a sequential hardwired machine which could be "programmed" by using a primitive instruction code. In modern jargon, one could say that the "compiler" was hardwired.

11 As quoted in: Scott McCartney, *ENIAC—The Triumphs and Tragedies of the World's First Computer*. Walker, New York, 1999, p. 94.

12 "Programming" is put between quotes here because it is not obvious to draw a line between users who program the computer and users who don't. On a certain level, clicking the mouse to open a file, pushing a button to send a mail or

adapted interface, one usually does not have any sense of the computations that are actually going on in the computer. One does not care about what the computer *actually* does, but only about what happens at the interface (except when some error occurs).
One does something and the computer returns a reply, but one often does not know exactly what happens between my actions and the computer's re-actions. This was very different with ENIAC and several other early computers. Indeed, in a way, you could not even avoid to "perceive" or "observe" the processes of computing of these early computers.
First of all, a computing ENIAC meant a lot of sound. There was the sound of the vacuum tubes, the clicking of the relays and the punch card reader and punch. Besides, the ENIAC also offered a visual spectacle. The numbers that were being processed in the accumulators – the main arithmetic units of ENIAC – were visible through a 10x10 matrix. While computations were being done, you could "see" these numbers change. As Jean Bartik described in an interview, observing how these computations developed over time was even an essential debugging method:[13]

> So consequently, when you were doing calculations these lights were flashing as the numbers built up and as you transferred numbers and things of this kind. They were very essential to debugging, very essential. [...] That's the only way you read what the machine [...] stored, what it was doing. [...] But it was from the ENIAC [...] where people saw for the first time, saw calculations taking place.

Today, having the ability to gain access to certain fragments of the hidden computational process is still quite fundamental: it is a basic technique used for instance while debugging a program. Indeed, while programming, you need to able to "listen to" or "have a look at" what is going on during the process of a computation. Since it is assumed that this is not a need of the everyday user, programming remains one of the few methods available to "understand" what the computer is doing, be it at a certain macrolevel. Indeed, with present microtechnology it has become quite impossible to "observe" the hardware compute, as was the case for the early computers. Instead, one can only "observe" a translation of these physical processes into some kind of symbolic representation, be it sound, numbers, letters, graphics, etc.
But what would be the point anyway to have *direct* access to the complete computational process that is happening inside our machines behind some interface like a programming language or a more common one like Word or LaTeX? Making available *every* computational step in some representation seems quite useless from our modern perspective. Not only would it exponentially slow down the computation[14] but this would also be quite "indigestible" for us humans: it is too much information coming in too fast. Moreover, if we would be demanding computers without any kind of symbolic interface, we would be back to something like ENIAC and its associated problems. We would certainly *not* be able to do the kind of things we do today with computers. We would be confronted with a practically insurmountable gap between what we want the machine to do and the time and effort it would take to set up the machine in order to do it; this was identified at the time as the programming bottleneck. If it would have remained unresolved, the whole effort of building an electronic machine because of the speed and memory it offers us, would have been senseless.

In order to bridge this gap it was necessary to develop an interface to communicate more efficiently with the machine and very early on in the history of digital computing the first thoughts on such intermediary languages were put into a practice, first, through the development of a machine language and then through the construction of compilers and, later, interpreters. This evolution made possible not just a more "efficient" communication with the machine but also allowed to create (the illusion of) machine-independent programming, viz. the idea that when you are working in a given layer/language you do not need to care about what lies beneath. Moreover, once memory

typing some text can also be regarded as programming.

13 Jean Bartik, Interview with J. Bartik and F.E.S. Holbertson, April 27, 1973. In: H.S. Tropp (interviewer), *Computer Oral History Collection, 1969-973*, Washington, Smithsonian Institution Press, Archives Center, National Museum of American History. Here: p. 63.

14 This was not the case for ENIAC since one "observed" the actions of the hardware itself.

became cheaper, one could start with the steady process of storing inside of the computer a wide range of programs which could/can then be called by their name or by “pushing” some visual button. In this way, the steady development of intermediary “languages” or “interfaces” implied the creation of a distance between what I am saying to the machine, for instance by typing a command in a shell or “moving” some “window” on my monitor, and how the machine understands this.

This distance has been constructed historically by adding layer upon layer of code and has become an almost inextricable web of connections between different so-called levels of abstraction. Nathan Ensmenger,[15] a historian of computing, speaks in this respect of software as a heterogenuous object. Moreover, as a complex evolving heterogeneous system, embedded in a larger sociotechnical context, it has almost become impossible to start from scratch: software /is/ history in the sense that whatever "software" we are using, it contains the traces of its own history, though these are usually hidden for the user.
In other words, the historically developed programming languages, user interfaces and programs that allowed the computer to become so ubiquitious in our society, are also the reason why we usually do not know what happens behind our interfaces and so we do not have full control. As a consequence, we are confronted with a first movement in the history of computing which renders Dijkstra's idea of the controlled machine an ideal which will perhaps never be real for most of the “users”. The user cannot control nor fully predict because he/she cannot see “through” the machine.

This lack of control is strengthened through a second historic movement which in fact initiated and drives the first. This second movement is that of the speed gained by the development of electronic machinery. This created yet another gap, that between the machine's speed and our own speed. John Von Neumann, computer pioneer, captured the problem quite clearly already in the 40s:

> "[It is] necessary to consider carefully the ability of the computing mechanism to take our intention correctly. And the person controlling the machine must foresee where it can go astray, and prescribe in advance for all contingencies. To appreciate this, contemplate the prospect of locking twenty people for two years during which they would be steadily performing computations. And you must give them such explicit instructions at the time of incarceration that at the end of two years you could return and obtain the correct result for your lengthy problem! This dramatizes the necessity for high planning, foresight, and consideration of the logical nature of computation. This integration of logic in the problem is a consequence of the high speed."

In other words, we need to find the means to guarantee that the machine will do what we want it to do and not something else, despite the fact that we humans are simply incapable of foreseeing the complete behavior of a given program. *If* we cannot achieve this, the lack of control due to the intransparancy of the machine with its intricate layers of different interfaces is deepened through our inability to predict the behavior of the machine due to its speed. The main solution to this problem was and is the use of logical control through the development of programming techniques and the steady delegation of (part of) the control to the machine itself.

Against these two movements – unpredictability through speed and intransparancy – we could of course argue that the lack of control they (can) result in is in no way necessary or essential to the development of computing machines. In fact, it is exactly this reasoning that was driving Dijkstra: he was observing that things *were* getting out of control and so he proposed a means to gain back our control resulting in his reflections on program semantics and correctness. Today, research and development of formal techniques to increase our control over the machine's behavior is still a major subarea in computer science and progress *is* being made. However, whatever control we have gained, it is gained only through the machine. It is, at best, the machine which verifies (part of) its own programs and so another layer is added.

---

15 Nathan Ensmenger. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. MIT Press, 2010.

I have argued here that despite the theoretical possibility of full mastery over the machine's behavior, the reality of modern computing is quite different. As a consequence, and in the light of Dijkstra's quote, it /is/ in fact already quite natural to understand our interactions with machines as conversations. We do not need a HAL to achieve that. This is clearly *not* how it is perceived by most, however. On the contrary, the ideal of full control is being sold as a reality to the users while many a practicioner is still working to ignore this reality for the sake of the ideal. The consequences of this are far from innocent.

First of all, many of us are living the illusion of full control by proxy. As soon as computing became a commerce, firms like IBM understood that the distance created between human and computer could be turned into a business model. The result of this is the construction of a user who is being told that he/she does not need to care about the technicalities of the machine, about what the computer *actually* does, and that he/she better gives most of the control to someone who knows what he/she is doing. This resulted in the development of so-called user-friendly interfaces that hide and even forbid access to certain parts of the machine and its software. For the belittled user, the machine is *really* a black box, an oracle which he does not understand, will not understand and cannot understand. In the meantime, private companies as well as national governements are taking full advantage of our own ignorance.

Secondly, within our paradigm of control which is devoid of the idea of conversation, we are paradoxically building computational systems of which the developers are admitting that we do not control them fully. A very recent example of this was published in Nature in January 2016 and concerns Google's DeepMind project which, for the first time, resulted in a system capable of playing the game of Go at an acceptable level (it beat the European champion). This was a very important result because Go was always considered to require an entirely different AI approach than the one used for chess playing machines[16]. The recent approach relies on deep neural nets which "learn" what "patterns" are the most promising for a win, by relying on expert games and self-play. In the Nature issue containing the result, the system is being "explained" to the public as one which is intuitive and is more human than any system has been before. However, what happens exactly inside of these deep neural nets is apparently not really known. Hence, the rather worrying conclusion in the editorial of the Nature issue containing the results:

> "As the use of deep neural network systems spreads into everyday life — they are already used to analyse and recommend financial transactions — it raises an interesting concept for humans and their relationships with machines. The machine becomes an oracle; its pronouncements have to be believed. When a conventional computer tells an engineer to place a rivet or a weld in a specific place on an aircraft wing, the engineer — if he or she wishes — can lift the machine's lid and examine the assumptions and calculations inside. That is why the rest of us are happy to fly. Intuitive machines will need more than trust: they will demand faith."

The question then is: do we really want that Google and other firms alike introduce new computational systems which are not only closed for us but also, apparently, for them?

When I am talking to someone, I know that I can never have full control over the process of the conversation itself. One does not control its dynamics nor its boundary conditions, let alone the other person. As a consequence, one cannot expect "perfect" communication. It is our lack of full control over the other person, and, by consequence, the conversation itself, that makes it necessary for me to try and understand the person I am talking to. *A common language does not suffice*. If I refuse to understand the person I am talking to, I will never manage to make myself understood to that person, and vice versa. This reciprocal need to understand each other is what binds people

16 Amongst others, the search space is much bigger than that of chess and recognizing winning and losing positions is harder (because the stones have equal value)

while having a conversation. Because of this bond, refusing to understand the person one is talking to, refusing to take into and give account of the other, in fact gives more control and power to that person. This is the known master-slave reversal. The master, who refuses to understand the slave because he thinks he controls the slave, will be the one who, ultimately, becomes the slave.

A kind of conclusion

As I have argued here I do not think we are in control of the machine's behavior: in reality, we cannot see *through* it nor *ahead* of it. Instead of awaiting a future where Dijkstra's ideal has become real for all, it seems opportune to take seriously this reality in the face of how private and public instances are taking away the control we *do* have over our communications with the machine. From this perspective, rather than talking *to*, we should learn to talk *with* machines and so transpose the framework of human-human conversation to human-machine conversations, taking into account and accepting that the other is *not* a human and so will *never* be *like* a human. One major challenge is then to identify the relevant properties of human conversation and rethink them in a context of human-machine conversations.

So what can we do? As I said, one fundamental feature of a real conversation is the reciprocal need for mutual understanding. In current everyday systems, the machine is completely oriented towards "understanding" the "user" as constructed by the manifacturers whereas the user does not need to understand a thing. A modest step towards human-machine conversation, which has and is in fact already taken by many a computer user, is the use of open or non-proprietary software combined with the maintenance of a conversation through programming interactions. The latter in fact already shares many properties with "real" conversations. Amongst others, one is quite frequently confronted with the unpredictability of the machine and one does get the "feeling" that one is constructing something in collaboration with the machine.

A major scientific step would be required to communicate with systems which, even if they would be open and even if we would be involved with programming them, are still apparently behaving as an oracle. I gave one example of this, viz. the deep neural networks that are currently the new "It" in A.I. (even though in fact the technology is very old). One recent proposal in this direction comes from Ann Copestake, a computational linguist. In a keynote she gave at CiE in Bucharest last year, she pointed exactly at the problem we are facing with the newer generation of A.I. systems such as those developed at DeepMind, namely that, currently, we do not have the means to communicate with those systems. Indeed, this is the reason why they behave as oracles. As she states:[17]

> "Lack of communication with 'intelligent' systems is a problem now, which will worsen as systems become more agentive and less domain-limited. [...] If a genuine general Artificial Intelligence does emerge, we really do want to be able to communicate with it/her/him."

Her proposal was that we should teach our systems to communicate with us by letting them learn from human communication using techniques of machine learning. If then, also the humans in their turn would make the effort to learn about machine conversation, perhaps the ideal of human-machine conversation is not entirely out of reach.

How to talk with a computer? My aim here was not to provide a definite answer to this question, but merely to show that in fact we *can* already talk with a machine provided we accept another way of thinking about machines which accepts that we are not in full control and that we can actually have interesting conversations with them. In fact, it seems to me that, paradoxically enough, it is the only way to get back our control. If we refuse to accept this we are on our way to create a HAL. Perhaps, in a certain way, we have already done so.

17 Ann Copestake, *Communication with Artifial Intelligences*, Computability in Europe 2014, Bucharest.