

Grove Hill House,
Grove Hill
Harrow.

May 15th 1951.

Dear Turing,

I have just been listening to your talk on the Third Programme. Most stimulating and, I suspect to many people provocative, but it fits extraordinarily well with what I have been thinking on the subject. In particular your remark (almost in passing) that the programme for making a machine think would probably have great similarities with the process of teaching; this seems to me absolutely fundamental.

It seems to me ~~at~~ that the most important difference between the sort of programme I can envisage for a machine and the sort of way in which a brain thinks, is that the brain is capable of discovering relationships and recognizing their importance. A machine, on the other hand, can only look for relations if it has been instructed what sort of relations to look for. It is obviously a fairly simple matter to instruct a machine to look for any one of a specified number of relations, but with the sort of programmes I can envisage so far, it is not possible to get it to look for new ones. Now the whole process of teaching - at any that of teaching the young - is one of showing them new relationships and getting them to recognize them. If it were possible to make the machine copy this process - in other words to make ~~the~~ it learn - then we should, I think, be a stage nearer the programme 'Think'.

I am convinced that the crux of problem of learning is recognizing relationships and being able to ~~use~~ use them. Of course there is a large amount of matter which has to be simply remembered in the form of facts - the multiplication table, for instance, or the words of the language - but without the assistance of a scheme of relations this knowledge is not much use. This was brought home to me in a very striking manner when I was investigating the behaviour of various possible types of game-

playing machines. I had the opportunity of teaching the game of Nim to a friend of mine who had not met the game before, but has a very good general intelligence. At the time I was looking in to a machine which recorded any winning position it reached and played to it if possible. If it could not reach a ~~position~~ winning position in one or two moves, it played at random. In other words the machine was given the rules of the game, and a memory, but not the mathematical theory, so that ~~in~~ at the start it was in exactly the same position as Anthony (who is no mathematician, and would certainly never be able to discover the mathematical theory). We played four games - I taking the part of the machine - and the machine won all four, but at the end, Anthony had discovered the general relation which held for one set of winning positions (the configuration $n, n, 0$) while the machine had only discovered three particular winning positions (1,1,0; 2,2,0; and 3,3,0). The man had therefore learned sufficient about the game in this time to beat the machine on every subsequent occasion.

This shows very clearly, I think, that one of the most important features of thinking is the ~~ability~~ ability to spot new relationships when presented with unfamiliar material; new, that is to say, in that they are new to that particular context. This process is obviously a rather advanced stage of learning - that of learning for oneself by experience, without the aid of a teacher and is by no means necessary except when attempting new or original thought. On subsequent occasions, a simple feat of memory is sufficient to couple the game 'Nim' with the operation 'look for the relation $n, n, 0$ ', so that a large measure of apparent thought might be possible without this facility. I think, however, that I have the glimmerings of an idea of the way in which a machine might be made to do it.

First, however, it would obviously be necessary to get the machine to learn in the way a child learns, with the aid of a teacher.

- 2 -

There are, I think, three main stages in learning from a teacher. The first is the exhibition of a few special cases of the rule to be learned. The second is the process of generalisation - i.e. the underlining of the important features that these cases have in common. The third is that of verifying the rule in further special cases and asking questions about it. I have ~~omitted~~ omitted any mention of 'understanding' the rule, because this is not appropriate at the moment to the action of a machine. I think, as a matter of fact, that the process of understanding a rule is connected with finding relationships between it and other rules - i.e. second (or higher) order relations between relations and this might well become important for a machine later.

As an example, when learning to differentiate powers of x , one first differentiates x , x^2 , and x^3 (stage 1) then points out the significant connection between the coefficient and the power (stage 2) and finally tries the rule out on further powers. At this stage various questions crop up - does the rule work for x^0 ? and for $x^{\frac{1}{2}}$? and for x^{-1} ? and for x^x ? The whole of this third stage of questioning and verifying, which at first sight seems quite unnecessary, is actually of the very greatest importance as it allows the rule to be altered and improved gradually with use.

Now I think it might well be possible to programme the Manchester machine to do all of these stages, though how much it would be able to learn in this way before the storage became inadequate remains to be seen. The essential thing which would have to be done first, would be to get the machine to programme itself from very simple and general input data. The problem seems to me to be largely one of devising a suitable notation for the input. (I am only considering teaching the machine mathematics - the reduction of other branches of learning to a suitable symbolic form seems a formidable task). It would be a great convenience to say the least if the notation chosen were intelligible as mathematics when printed by the output. This raised the difficulty of doing mathematics on the typewriter in

a rather acute form as the teletyper has no back space and no ~~xx~~ means of turning the lines downwards, so that powers and fractions are almost impossible to do satisfactorily. In addition various signs such as ~~xxx~~ $=$, $($, $)$, π , $_$, would be needed,

My idea is that once the suitable notation is decided, all that would be necessary would be to type more or less ordinary mathematics and a special routine called, say, 'Programme' would convert this into the necessary ~~xxx~~ instructions to make the machine carry out the operations indicated. This may sound rather Utopian, but I think it, or something like it should be possible, and I think it would open the way to making a simple learning programme. I have not thought very seriously about this for long, but as soon as I have finished the Draughts programme I intend to have a shot at it.

I have completed my first effort at the Draughts, but I am not yet satisfied with it. It is far too slow, and does not make full use of the great storage capacity available, being only a slight modification of the one I designed for the ACE. Moreover I now think that it would play very bad draughts, as the valuation method I adopted is much too crude. I have been having some fun however with the input and output arrangements. Not having the proper ones, I have devised my own (incidentally I should much prefer instruction TJ to be $A \neq S_+$ - the fact that it is actually S_+ always seems to involve using an extra instruction when checking the printing in order to make sure that M is clear) and have amused myself by making the machine get impatient if its opponent is too slow, or makes mistakes in feeding in his move, and finally, if he makes too many mistakes, in making it loose its temper completely and refuse to play any more. I will send you the draughts programme itself as soon as I am moderately satisfied with it.

Please excuse such a long letter - I am quite sure you are far too busy to answer it - you must blame your talk for being too stimulating.

Yours sincerely

Christopher Strachey