A computer program capable of passing I.Q. tests

Pritika Sanghi (psan5@student.monash.edu)
School of Computer Science and Software Engineering
Monash University, Clayton, VIC 3800 Australia

David L. Dowe
School of Computer Science and Software Engineering
Monash University, Clayton, VIC 3800 Australia

Abstract

The Imitation Game (Alan M. Turing, 1950), now commonly known as the Turing Test (see, e.g., Oppy and Dowe, http://plato.stanford.edu/entries/turing-test, 2003), was proposed as a way in which thinking or intelligence could be ascribed to any agent - including a computer program or machine - able to play the game. People routinely ascribe intelligence to humans and other animals by a variety of means, including those discussed by Turing. But when humans wish to specifically quantify intelligence, this is most commonly done by means of an intelligence quotient (I.Q.) or other aptitude test. Many such aptitude test questions fit in to Turing's (1950) framework of being able to be typewritten to a teleprinter communicating between two rooms - or, using modern technology still well within the spirit of Turing's game, being able to be typed as text into a World Wide Web (WWW) page applet. Sequences of such questions - such as an entire I.Q. test of them - may well form a strict subset of Turing imitation games, since they typically are independent of one another and do not take any advantage (or even account) of the contextual (conversational) framework of Turing's game. We present here a fairly elementary WWW-based computer program (shown in large part at http://www-personal.monash.edu.au/~psan5) which, on a variety of I.Q. tests, regularly obtains a score close to the purported average human score of 100. One conclusion that some might make is to ascribe intelligence to the program. Another conclusion to make is that the reason that I.Q. test success can be automated with comparative ease is that administering an I.Q. test requires little intelligence - it requires comparatively little more than giving a list of questions with known answers. Turing's imitation game test requires greater intelligence to pass largely because of the flexibility it permits to an intelligent questioner - such as in the use of language and in taking into account the responses to previous questions before continuing the line of questioning. We also briefly consider administration of the imitation game "test" via "detection programs" as a test in its own right (Dowe and Hajek, 1998; CalTech Turing Tournament, http://turing.ssel.caltech.edu, 2003). All other things being equal, a more intelligent administrator can administer a more challenging test - and this notion can be continued recursively (Dowe and Hajek, 1998).

1. Introduction

Alan Turing suggested the Imitation Game (Alan M. Turing, 1950), now known as the Turing Test, in 1950 as a way of ascribing thinking, or intelligence, to machines. It involves the interrogation by a judge of a human and a machine using teletype from behind a screen where the judge knows that one is a human and one is a machine, but the judge does not know a priori which is which. The test requires the machine to fool the judge into believing that it is human and that the human is the machine - and the Turing Test has certainly been frequently discussed and surveyed (Moor, 2000; Saygin et al., 2000; Copeland, 2000; V. Akman and P. Blackburn, 2000; Oppy and Dowe, 2003). I.Q. tests are used to measure the level of intelligence of humans. If a computer is made to take an I.Q. test, it, too, can be given a score based on its performance. Given that such scores are used as a measure of human intelligence, it seems plausible that such a score might be used as an indication of the level of intelligence of the computer.

We present here a program which takes I.Q. tests in the form of questions typed in the text box of a simple webpage. The score can be calculated based on the number of questions it answers correctly. In cases where the question cannot (yet) be represented to the program (e.g., picture questions, such as in Figure 1) and there are n options, the program gets a score of 1/n for the question. This is because the probability of getting the answer right is 1/n and so the long run average score from guessing is (n-1)/n x 0 + 1/n x 1 = 1/n.

More is said about I.Q. tests and their constituent questions in Section 2, the program in Section 3, and the program's performance in Section 4; and in Section 5 we discuss administration of the Turing Test as a test in its own right and the relevance of information-theoretic compression to inductive learning and intelligence.

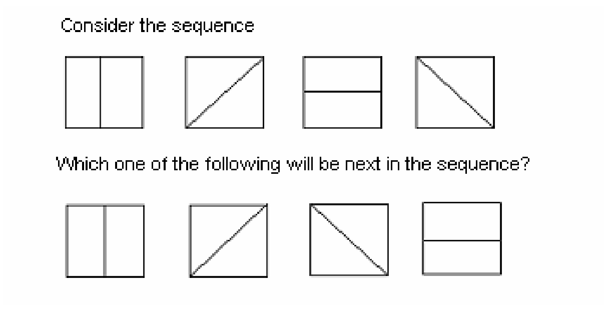


Figure 1: A picture question.

2. I.Q. Tests

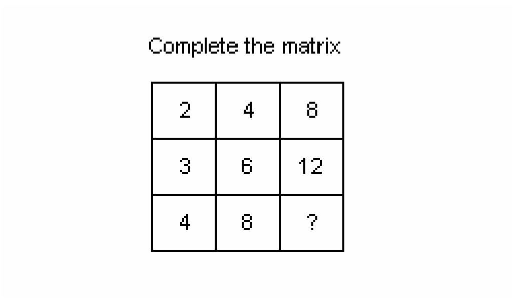
Intelligence Quotient or I.Q. (A.C.E., unknown; H. J. Eysenck, 1988; Helenelund HB, 2001; http://heim.ifi.uio.no/~davidra/IQ/, unknown; I.Q. Test Labs, 2003; KHAN-UUL Institute, 2001; Mensa, 2000; Testedich.de, 2002) is used as a measure of human intelligence. The first I.Q. test was conducted by a French scholar, Alfred Binet, around 1906 (KHAN-UUL Institute, 2001; Enyclopedia Britannica, 2000). The original purpose of the test was to identify slow learners in school. I.Q. is the ratio of mental age over chronological age. For example, consider a child of age 12. A mental age of 10 will suggest an I.Q. of 10/12 x 100 = 83; and a mental age of 14 will suggest an I.Q. of 14/12 x 100 = 117. An I.Q. of 100 implies that the mental age is same as the chronological age.

If a computer program can achieve a score between 90 and 110, it can arguably be said to have average intelligence (H. J. Eysenck, 1988). The program should ideally not be test specific, as it would be rather abnormal to get a score of over 120 on one I.Q. test and under 50 on another test.

2.1 Forms of frequently asked I.Q. questions

Most I.Q. tests seem to have certain similar characteristics. An analysis of various I.Q. tests (such as those listed above) shows that the following types of questions are common:

1. Insert missing number
   a. At end
   b. In middle
2. Insert missing letter
   a. At end
   b. In middle
3. Insert suffix/prefix to complete two or more words
4. Complete matrix of numbers/characters (see Figure 2)
5. Questions involving directions
6. Questions involving comparison
7. Picture questions (see Figure 1)
8. Pick the odd man out (word or picture)
9. Coding


Complete the matrix

2	4	8
3	6	12
4	8	?

Figure 2: A matrix question

3. The Program

The program, which can be accessed from http://www-personal.monash.edu.au/~psan5, recognizes these characteristics (forms of frequently asked I.Q. questions) from Section 2.1 and tries to find the best-suited solution. It is written in Perl and is about 960 lines of code.

A parser with a restricted vocabulary and a basic string search for keywords can be implemented to recognise the type of question. It can then be simple for the program to calculate the answer based on some pre-defined rules.

A trivial way of recognising the question would be to look for patterns such as “What is the next number in the sequence”. A small change in the format of the question will cause the current version of the program to fall over. We can overcome this by looking for certain keywords (e.g., number + sequence). A comprehensive list of keywords can be made for each type of question. If two questions have similar or identical keywords, extra keywords or patterns need to be included to differentiate between them.

A thorough list of possible keywords for a certain type of question can be made by analysing a large number of I.Q. tests. Even then there may be cases where it cannot identify the question or identifies it incorrectly. Provided the keyword list is made properly, this should be rare.

Once the type of question is established, it can be simple for the program to find the answer. An algorithm that does calculations and/or searches can easily be made that can find most of the answers to the questions. A large number of questions that involve simple logic can be programmed.

We now discuss the program’s answers to I.Q. questions of frequently asked forms, such as those from Section 2.1.

3.1 Insert missing number or letter at end

Consider the question – “Insert the next number in the sequence - 1 2 3 4 5”. ‘Number’ could be replaced by ‘digit’, and ‘sequence’ could be replaced by ‘series’. It could also be phrased as “Which number follows logically - 1 2 3 4 5” or “What is next in the sequence - 1 2 3 4 5”. Keywords for this type of question could be ((insert || what || which) && (number || digit || sequence || series)), where ‘&&’ denotes ‘and’ and ‘||’ denotes ‘or’. An example of phrasing that can be used for more than one type of question would be – “What is next in the sequence - 1 2 3 4 5”, or “What is next in the sequence - A B C D E”. They can be differentiated by the fact that one has a sequence of numbers and the other has letters. The keywords for number sequences may be extended to check that there are only digits (0-9) and separators in the sequence section. The keyword for character sequences will not be extended as they can contain numbers and characters in the sequence section.

In the case of sequence questions, certain types of sequences (Arithmetic Progression, Geometric Progression, Fibonacci Series, Powers of a series, etc.) are used frequently in I.Q. tests - see, e.g., (A.C.E., unknown; H. J. Eysenck, 1988; Helenelund HB, 2001; http://heim.ifi.uio.no/~davidra/IQ/, unknown; I.Q. Test Labs, 2003; KHAN-UUL Institute, 2001; Mensa, 2000; Testedich.de, 2002). If the given sequence is any of the types of sequence described above, it can be checked with ease. If it is, the next number/character can be calculated simply according to the properties of the sequence, although at least three numbers/letters are required to find a pattern. The program can solve this type of question from I.Q. tests most of the time. There will be cases when an answer cannot be found or the answer found is incorrect. Since it is rare for humans to get all answers correct, it is not considered a big problem.

3.2 Insert missing number or letter in middle

This type of question will be recognised using a technique similar to the one used in Section 3.1. An example would be ‘Insert the missing number: 10 20 ? 40 50’. The program will look for a special character (x || _ || ?) inside the sequence, where, again, ‘||’ denotes ‘or’. The number/s or letter/s for that position/s can be guessed using the properties for sequences mentioned in Section 3.1. The entire sequence is then checked. If valid, the answer is found. This is not yet implemented in the program.

3.3 Insert suffix/prefix to complete two or more words

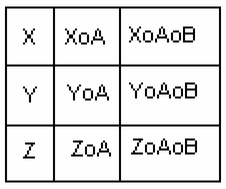
This kind of question can again be recognised by keywords. For questions involving suffix and prefix (e.g., What completes the first word and starts the second: wi..nt), keywords could be ((suffix && prefix) || (complete && word)), where, ‘||’ denotes ‘or’ and ‘&&’ denotes ‘and’. Brute force is then used to search for a suffix for the prefix from the word list. Next it checks if that suffix is a prefix for the suffix. Sometimes, only a suffix is requested for more than one word (e.g., Insert a word of size 2 that completes the words: ma, fa, chara (..)). The technique remains much the same. Instead of checking if the prefix is valid the next time it will check if it is a valid suffix for the rest of the words. A repeated prefix can also be found using the technique with slight modifications (e.g., Insert the word of size 2 that completes the words: (..) de, ke, lt, trix). Most of this is implemented in the program. Currently, the program requires the question to be re-formatted from ‘wi..nt’ (it is found on I.Q. tests in this or similar format) to ‘2-wi-nt’. This can be implemented but it hasn’t yet been done in the program. With the re-formatted input the program generally finds the solution.

3.4 Complete matrix of numbers/characters

In order to complete the matrix, patterns (e.g., for Figure 2, a column is double the previous column) existing inside the matrix have to be identified. Once they are identified, the pattern can be applied to find the most appropriate value. Patterns can be found using the following techniques:

1. The sum of rows can be the same.
2. It could be in the form described in Figure 3, where ‘o’ represents an arithmetic function such as ‘+’, ‘-’, ‘*’, ‘/’, ‘^’, etc.
3. Operators (‘+’, ‘-’, ‘*’, ‘/’, ‘^’, etc. and one ‘=’) are inserted between columns. If the same combination is valid for each row, that is the pattern.
4. Zig-zag through rows to find a pattern.
5. The matrix is transposed and above steps are repeated.
6. Columns are shuffled and steps 3, 4 are repeated.

This is not yet implemented in the program



X	XoA	XoAoB
Y	YoA	YoAoB
Z	ZoA	ZoAoB

Figure 3: Patterns in matrix questions

3.5 Questions involving directions

Keywords like (left || right || east || west || north || south) can be used to identify questions involving directions. Directions can be represented quite easily using degrees. Assuming north to be 0^o and moving clockwise, east becomes 90^o. The points can be represented relative to the starting point. The distance between them can be calculated using methods such as Pythagoras's theorem. For example, 'Joe moves three blocks east, takes a right turn and walks further four steps. How far is he from his original position?' From the first part of the sentence it will take 'three' and 'east' and make the new position 3 units 90^o right of north. Next, 'right turn' and 'four' will be taken into consideration, making the new position 90^o right, 4 units away from previous position. The distance between points can be calculated using properties of a triangle. This is not yet implemented in the program.

3.6 Questions involving comparison

First, all the elements being compared should be listed. Positions are given to elements relative to others based on initial sentences. The list is parsed for elements, which can be given position relative to other elements. Consider, e.g., "A is taller than B, B is taller than C". In the second parse, A, B, and C will be given positions relative to one another (rather than merely the two initial separate comparisons not relating A to C). The two extremes and the complete, total ranking can then be found. This is not yet implemented in the program.

3.7 Picture Questions

The aim of picture question is to check for pattern recognition. It will be hard for a program to solve picture questions (see, e.g., Figure 1). This is mainly because it is tough to represent them in a teletype environment. If the picture were defined symbolically, perhaps too much work would have been done in overly assisting the program. If the picture is described (e.g., the second element in the sequence is an unshaded square with a diagonal top right to bottom left), then parsing will be a challenge (see also Section 5.1). This is not yet implemented in the program.

3.8 Odd man out

Odd man out questions are commonly used in I.Q. tests (A.C.E., unknown; H. J. Eysenck, 1988; Helenelund HB, 2001; http://heim.ifi.uio.no/~davidra/IQ/, unknown; I.Q. Test Labs, 2003; KHAN-UUL Institute, 2001; Mensa, 2000; Testedich.de, 2002). They require one to differentiate names of countries, cities, vegetables, fruits, etc. To differentiate between objects, one probably needs to understand their concept. But, there may be many categories for objects (e.g., for picture questions from Section 3.7, a circle is in both the family of geometric shapes, and also geometric shapes with no edges). It could be a challenge for some time yet to make a computer program understand the concept behind different pictorial objects. However, odd man out questions not involving pictures may well be amenable before too long to a search analogous to a much more complicated version of that required in Section 3.3. This is not yet implemented in the program.

3.9 Coding

These questions generally involve coding from alphabets to numbers or vice versa (e.g., If KNOW is 20 23 24 32, what is CODE?). These questions are of the kind
(if && (a-z || A-Z || 0-9)+ && is && (a-z || A-Z || 0-9)+ && what && is && (a-z || A-Z || 0-9)+), where '+' denotes one or more and, as before, '||' denotes 'or' and '&&' denotes 'and'. A relation is found between 'KNOW' and '20 23 24 32' based on ACSII values. The same relation is then used to find the code for 'CODE'. This is not yet implemented in the program.

3.10 Other kinds of questions

An I.Q. test that steers away from the usual way of testing (certain types of questions can be expected most of the time) or a non-standard test will be cause for concern. Not being able to identify the questions, the program will fail the test. It is highly unlikely that a human can't comprehend any or most of the questions on an I.Q. test.

4. Results – the program's "I.Q."

We present here the results of the program on various I.Q. tests. Most of the questions from the I.Q. tests were re-formatted before being entered to the program (see Section 3).

Table 1: I.Q. Scores on various tests.

Test	I.Q. Score	Human Average
A.C.E. I.Q. Test	108	100
Eysenck Test 1	107.5	90-110
Eysenck Test 2	107.5	90-110
Eysenck Test 3	101	90-110
Eysenck Test 4	103.25	90-110
Eysenck Test 5	107.5	90-110
Eysenck Test 6	95	90-110
Eysenck Test 7	112.5	90-110
Eysenck Test 8	110	90-110
I.Q. Test Labs	59	80-120
Testedich.de – The I.Q. Test	84	100
I.Q. Test from Norway	60	100
Average	96.27	92-108

As seen from Table 1, the program scores high on some tests and low on others. A link can be seen between the score and the type of I.Q. test. The program can attain a high score with ease on I.Q. tests which are more focussed on mathematics, pattern recognition, logical reasoning and computation. On the other hand, I.Q. tests that are based on general knowledge, language skills and understanding are a challenge to the program. The case is often the reverse for humans. Of course, human I.Q. tests are anthropomorphic (or "chauvinistic"), and an intelligent non-English speaker, non-human earthling or extraterrestrial could be expected to struggle on an English-language I.Q. test.

5. Some thoughts and discussion

First, apart from relatively minor issues such as memory and running speed, the "intelligence" of a computer presumably depends almost solely on the software program it is running. That said, we now ask some additional rhetorical questions.

Are I.Q. tests really a measure of our intelligence? Will getting a higher score than a human mean that the computer program is more intelligent than that human?

Without necessarily fully answering either or both of these two questions, let us take this discussion in two directions. In so doing, we shall consider two possible modifications to the Turing test.

5.1 Administering the Turing test

Recalling Sections 2.1 and 3, most I.Q. test questions fit straightforwardly into Turing's original conversational framework. With however much more work (to a human or possibly a non-human possibly poised with pen(cil) and paper), picture/diagram questions can probably also – by careful description – be brought within Turing's conversational framework. So, I.Q. test questions seem, by and large, to fit neatly into Turing's conversational framework. Whereas the judge (or administrator) of Turing's imitation game test can lead the conversation in any number of directions given the conversation so far, I.Q. test questions asked largely or totally neglect the answers to previous questions. In other words, an I.Q. test requires less intelligence to administer than a Turing imitation game test – and this is essentially why it is less challenging and easier (for a computer program) to pass. This raises the issue that intelligence is required to administer a Turing test – and *this* ability could be used as a measure in a test for intelligence. Of course, we can continue this recursively (Dowe and Hajek, 1997; Dowe and Hajek, 1998). More explicitly, some intelligence is required to pass a Turing test (TT0). If we set up a new test, TT1, which is to correctly administer/judge TT0, then that presumably or seemingly requires more intelligence to pass. The test for "detection programs" in the Caltech Turing Tournament (CalTech, 2003) is a case in point. And, of course, we can continue this recursively (Dowe and Hajek, 1997; Dowe and Hajek, 1998) and, e.g., set up a new test, TT2, which is to correctly administer/judge/detect in TT1. (Passing the Turing Test, TT0, is analogous to writing a good academic paper. Passing TT(1) is analogous to being a good referee. Passing TT(2) is analogous to being a good member of a program committee or editorial board – one must be able to choose appropriate referees.) Continuing recursively by induction, given test TT(i), we can set up a new test, TT(i+1), which is to correctly administer/judge/detect in TT(i). Etc. While all of the above is true and TT(1), TT(2), …, TT(i), … are all interesting, salient and worthwhile directions in which to re-examine the Turing Test (TT0), it would also at least appear to follow by mathematical induction that each of TT(1), TT(2), …, TT(i), … can be expressed – albeit seemingly in increasing order of difficulty – in Turing's original conversational framework, TT0.

5.2 Inductive learning, compression and MML

The other direction in which we take this discussion is the observation – see (Dowe and Hajek, 1998) and elsewhere – that traits which seem to be necessary for (human) intelligence and which certainly are assessed in human I.Q. tests include rote learning (and memory), deductive learning (e.g., via modus ponens) and inductive learning. Inductive learning is perhaps the most important, significant and impressive of these. When asked for a list of great thinkers and minds, the primary reason for the inclusion of Newton, Darwin, Einstein and others is because of their inductive inferences - or inductive learning - of theories (gravity and laws of motion, evolution, relativity, etc.).

The relevance of compression to learning languages is discussed in (Wolff, 1995). The Minimum Message Length (MML) theory of inductive inference (Wallace and Boulton, 1968; Wallace and Freeman, 1987; Wallace and Dowe, 1999) states that the best theory, H, to infer from data, D, is that which minimises the (compressed) length of a two-part message transmitting H followed by D given H. MML is a quantitative form of Ockham's razor (Needham and Dowe, 2001), rewarding simple theories which fit the data well. The relationship between MML, Kolmogorov complexity (Kolmogorov, 1965) and related (information-theoretic) works (e.g., (Solomonoff, 1964)) is thoroughly discussed in (Wallace and Dowe, 1999). MML is relevant to inductively learning all range of models – not just languages – from data.

Given the relevance of two-part MML compression to inductive learning, Dowe and Hajek have argued (Dowe and Hajek, 1997; Dowe and Hajek, 1998) that an "additional requirement on Turing's test is to insist that the agent being subjected to the Turing test not only pass the test but also have a concise, compressed representation of the subject domain". Independently but quite relatedly, Hernandez-Orallo and Minaya-Collado (1998) also propose that “intelligence is the ability of explanatory compression”. They then go on to propose a variation of the I.Q. test based on Kolmogorov complexity.

6. Conclusion

While Section 5 considers two interesting possible modifications to the Turing (imitation game) test, the bulk of the paper concerns the performance of a comparatively small computer program (approximately 960 lines of Perl code) on human I.Q. tests. The program obtains very close to the purported human average score both on a variety of I.Q. tests and on average. Although some human pre-processing admittedly takes place with some forms of questions, it should be emphasised (see Section 3) that both viable improvements in the parser(s) and an increase in the number of question forms attempted should enhance the program’s score while reducing or even eliminating human pre-processing requirements. In addition and at the very least, even the current preliminary version of the program could assist and augment the score of a human able to parse the questions.

7. Acknowledgments

The second author expresses his life-long gratitude, encouragement, best wishes and more to a life-long special friend.

8. References

A.C.E. (unknown). Official A.C.E. I.Q. Test, *Association of Christian Era (A.C.E.)*, http://www.aceviper.net/.

V. Akman and P. Blackburn (2000). Editorial: Alan Turing and Artificial Intelligence. *Journal of Logic, Language & Information* (vol. 9, no. 4, pp 391-395).

CalTech Turing Tournament (2003). *Turing Tournament at California Institute of Technology (CALTECH)*, http://turing.ssel.caltech.edu.

B.J. Copeland (2000). The Turing Test, *Minds and Machines* (vol. 10, no. 4, pp 519-539).

D.L. Dowe and A.R. Hajek (1997). A computational extension to the Turing Test, *Technical Report #97/322*, Department of Computer Science, Monash University, Clayton 3168, Australia.

D.L. Dowe and A.R. Hajek (1998). A non-behavioural, computational extension to the Turing Test, *Proceedings of the International Conference on Computational Intelligence & Multimedia Applications (ICCIMA'98)* (pp 101-106), World Scientific publishers (ISBN 981-02-3352-3), Feb. 1998, Gippsland, Australia.

Enyclopedia Britannica (2000). *Britannica Encyclopedia 2000 Deluxe Edition CD-ROM.*

H.J. Eysenck (1962). *“Know your own I.Q”*, Penguin, U.K.

Helenelund HB (2001). *I.Q. Tests*, http://www.2h.com/iq-tests.html.

J. Hernandez-Orallo and N. Minaya-Collado (1998), A Formal Definition of Intelligence Based on an Intensional Variant of Algorithmic Complexity, *Proceedings of International Symposium of Engineering of Intelligent Systems (EIS’98)* (pp 146-163), Tenerife, Spain.

http://heim.ifi.uio.no/~davidra/IQ/ (unknown). *I.Q. Test*, Norway.

I.Q. Test Labs (2003). *I.Q. Test*, www.intelligencetest.com/quizzes/quiz1/index.htm.

KHAN-UUL Institute (2001), *IQ center*, http://khan-uul.mn/Eng/IQ_centre.html#4.

A.N. Kolmogorov (1965). Three approaches to the quantitative definition of information, *Problems in Information Transmission* (vol. 1, no. 1, pp 1-7).

Mensa (2000). *Mensa Workout*, http://www.mensa.org/workout.html.

J.H. Moor (2000). Alan Turing (1912-1954), *Minds and Machines* (vol. 10, no. 4, p 461).

S.L. Needham and D.L. Dowe (2001). Message Length as an Effective Ockham’s Razor in Decision Tree Induction, *Proc. 8th International Workshop on Artificial Intelligence and Statistics (AI+STATS 2001)* (pp 253-260), Key West, Florida, U.S.A.

Oppy, G.R. and D.L. Dowe (2003). *Stanford Encyclopedia of Philosophy* (http://plato.stanford.edu) entry on the Turing Test (http://plato.stanford.edu/entries/turing-test), Thu. 10 Apr. 2003.

A.P. Saygin, I. Cicekli and V. Akman (2000). Turing Test: 50 Years Later, *Minds and Machines* (vol. 10, no. 4, pp 463-518).

R.J. Solomonoff (1964). A Formal Theory of Inductive Inference I and II, *Information and Control*, *7* (pp 1-22 & 224-254).

Testedich.de (2002). *The I.Q. Test*, http://allthetests.com/tests/iqtest.php3.

Alan M. Turing (1950). Computing Machinery and Intelligence, *Mind*, Vol 59, (pp 433-460); also at http://www.loebner.net/Prizef/TuringArticle.html.

C.S. Wallace and D.M. Boulton (1968). An information measure for classification, *Computer Journal* (vol. 11, pp 185-194).

C.S. Wallace and D.L. Dowe (1999). Minimum Message Length and Kolmogorov Complexity, *Computer Journal* (vol. 42, no. 4, pp 270-283).

C.S. Wallace and P.R. Freeman (1987). Estimation and inference by compact coding, *Journal of the Royal Statistical Society (Series B)* (vol. 49, pp 240-252).

J.G. Wolff (1995). Learning and reasoning as information compression by multiple alignment, unification and search, In A. Gammerman (ed.), *Computational Learning and Probabilistic Reasoning* (pp 223-236), Wiley, New York.