Text Compression as a Test for Artificial Intelligence

Matthew Mahoney

Florida Institute of Technology
Computer Science Dept.
150 W. University Blvd., Melbourne FL 32901
matmahoney@aol.com

Abstract

The Turing test for artificial intelligence is widely accepted, but is subjective, qualitative, non-repeatable, and difficult to implement. An alternative test without these drawbacks is to insert a machine's language model into a predictive encoder and compress a corpus of natural language text. A ratio of 1.3 bits per character or less indicates that the machine has AI. Three pieces of evidence support this claim. First, text compression is shown to be more stringent than the Turing test under reasonable assumptions. Second, humans use high-level knowledge in character prediction tests. Third, compression, like AI, is unsolved: under conditions in which human text-prediction tests show an entropy of 1.3 bits per character or less, the best compression algorithm known achieves 1.87 bits per character.

Introduction

We propose using data compression as a measure of artificial intelligence (AI), rather than the Turing test. The widely-accepted Turing test says that a machine has AI if it cannot be distinguished from a human by another human, based on communication through a terminal. Unfortunately the test is subjective; the outcome depends not just on the machine, but on the expertise and motivations of both humans. Furthermore, the qualitative and non-repeatable nature of the test makes it difficult to evaluate and compare candidate systems.

Data compression has none of these drawbacks. The test is quick, quantitative, objective, and repeatable. A candidate system is tested by inserting its language model (the estimated probability distribution of input/output pairs) into a predictive encoder and compressing an appropriate corpus of natural language text. A compression ratio of 1.3 bits per character or less represents AI. This is the upper bound on the entropy of written 27-character literature (monocase letters and spaces) as measured using human character-prediction tests.

Prior work with stochastic language acquisition suggests that AI and compression are related (Hutchens and Alder 1997, 1998a). We present three arguments in favor of this position. First, it can be shown that, using a reasonable model, a compression test is more stringent than the Turing test. Second, people use all levels of observable knowledge in character prediction: lexical, syntactic, semantic, and real-world. And third, compression, like AI, is unsolved. We find that the best data compression algorithm achieves only 1.87 bits per character on 27-character English literature.

Limitations of the Turing Test

If machines could think, how would we know? In 1950, Alan Turing brushed aside philosophical arguments and proposed a strictly behavioral test. It is generally accepted that people can think, whatever thinking is, and furthermore, that people can recognize intelligent behavior in others. Therefore, if a machine cannot be distinguished from a human, based on communication via a terminal, then the machine exhibits artificial intelligence (Shieber 1994). The *imitation game*, better known as the *Turing test* is now a widely accepted definition of AI (Rich and Knight 1991). Passing the Turing test is informally considered an *AI-complete* problem, the hardest problem in natural language processing (Raymond 1997). In order for a machine to appear human, it must perform just as well at natural language subtasks such as language translation, information retrieval, and proofreading. Turing predicted that by 2000, we would have machines that would be mistaken for humans 30% of the time after 5 minutes of conversation, a goal yet to be met.

As a practical tool for evaluating and comparing AI systems, the Turing test is expensive, subjective and gives variable results. The outcome could depend greatly on whether we use computer experts, psychologists, or children in the test. Also, the test rewards machines for reproducing human weaknesses as well as strengths, such as simulating human error.

These shortcomings are best illustrated by the Loebner competition, held annually since 1990 (Flinders University 1998, Hutchens 1998b, Shieber 1994). In a typical competition, 4 to 8 machines and 2 to 6 human confederates (collectively called *agents*) communicate with 10 judges via text-oriented terminals. The judges independently rank the agents from most human to least human, and also classify each agent as human or machine. The machine with the highest median ranking (using the mean ranking to break ties) wins $2000. If the highest ranking machine is ranked higher than the lowest ranking confederate, it wins $25,000. No machine has won this prize, but in 1998 (6 machines, 4 confederates) the winning machine was ranked higher than a human in 15% of the 40 possible judge-confederate pairings.

Examination of the transcripts reveals some interesting phenomena that have nothing to do with AI. Some of the human confederates also tried to mislead the judges with statements such as *I am a computer*. One confederate adopted the maddening habit of always responding to the second to last input from the judge, as if to simulate some type of computer error. The judges were not fooled. A simple distinguishing test would be to look for spelling and typing errors, which were absent in most machines.

Shieber criticized the goals of the Loebner competition as being at odds with AI research. Instead of rewarding advances in understanding intelligent behavior, the contest rewarded machines that exploited human weaknesses, such as the tendency to interpret random text as non-sequitor, to assign meaning where none existed. Some machines would steer the conversation to some narrow domain, for instance, a fictional life story. Others exploited the *Eliza effect*, named after a 1971 program that imitated a Rogerian psychotherapist by echoing back the user's statements, swapping *you* for *I*, etc. One entry, *Parry*, imitated a paranoid schizophrenic, spewing random, incoherent text. Claude Shannon illustrated how an order-2 approximation to English (sequences of 2 words have the same probability distribution) appears halfway readable, with the following example (Shannon and Weaver 1949, p. 14):

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

Hutchens' entry, MegaHAL, which took second place in 1998, used a similar strategy with higher-order statistics, in addition to some canned responses. Considering that Shannon's example was generated in 1949 without a computer, it is remarkable how little progress we have made.

The Compression Test for AI

In order to determine if a machine has artificial intelligence, we propose the following test. We take the language model, the estimated probability distribution of input/response pairs, and measure the cross entropy of a corpus of natural language text relative to the model. The result is just the compression ratio that we would get if we used the model to assign codes in a predictive encoder. Lower ratios are better, and 1.3 bits per character (the entropy measured using humans) indicates AI.

A language model is a generalization of the formal model of computation to interactive, stochastic systems. In the formal model, a machine accepts a string (a finite sequence of characters drawn from a finite alphabet) as input, and produces a string as output. The machine's behavior can be described using a language L, a set of strings. If the machine responds to input string x with output string y, then the string x/y/ is an element of L. We will use the distinguished character "/" to denote the end of a message.

An interactive system alternates between input and output. A machine responds to successive inputs x_1, x_2, ..., x_n with outputs y_1, y_2, ..., y_n if and only if the string $x_1/y_1/x_2/y_2/.../x_n/y_n/$ is in L. We call an alternating sequence of zero or more input and output messages a *dialog*.

A stochastic system does not always respond the same way to a given input. Instead, a language L represents a probability distribution over the set of all dialogs. We define L(x) as the probability that a dialog will have prefix x:

$$L(x) \equiv \sum_y P(xy|L)$$

Thus, the probability of responding to input x with output y is

$$P(y|x,L) = \frac{L(x/y/)}{L(x/)}$$

Similarly, given the language L and an arbitrary string x = $x_1x_2...x_n$ of length |x| = n characters, then the conditional probability that the next character will be x_{n+1} is

$$P(x_{n+1}|x,L) = \frac{L(xx_{n+1})}{L(x)}$$

Thus, every system can be modeled as one which generates output by selecting characters x_{n+1} with this probability.

When we build a machine in an attempt to copy the behavior of an existing system, we may not know the exact probability distribution of its language. If M is an estimate or model of a language L, then one measure of the accuracy of M is the *cross entropy* of L with respect to M, denoted $H_M(L)$, and defined as

$$H_M(L) \equiv \lim_{n\to\infty} \frac{1}{n} \sum_{x,|x|=n} L(x)\log_2 \frac{1}{M(x)}$$

This is the expected compression ratio (in bits per character) for strings of length n, if each string x is assigned a code of length $\log_2 1/M(x)$ bits and x occurs with probability L(x). By the Kraft inequality (Abramson 1963), this code length is the smallest we can use so that every string x is assigned a unique code. By Shannon's first theorem, or the noiseless discrete channel capacity theorem (Shannon and Weaver 1949), $H_M(L)$ is minimized (achieving optimal compression) when M exactly equals L. We call $H(L) \equiv H_L(L)$ the *entropy* of L.

We still cannot find $H_M(L)$ if L is not known. Suppose instead that we have some large sampling of n dialogs, $x = x_1x_2...x_n$ generated by L, the system that we wish to model. If the dialogs are independent, $L(x_ix_j) = L(x_i)L(x_j)$ for all $1 \leq i, j \leq n$, then we can estimate the cross entropy as the number of bits per character in an optimal encoding of x using language model M,

$$H_M(L) \approx \frac{1}{|x|}\log_2\frac{1}{M(x)} = \frac{1}{|x|}\sum_{i=1}^{n}\log_2\frac{1}{M(x_i)}$$

because each x_i occurs with probability $L(x_i)$.

The sample dialogs should contain examples of the behavior we wish to simulate. If we wish to simulate human behavior, then the dialogs should consist of natural language text. For instance,

2+2=/4/
"white house" in Spanish is/casa blanca/
What is the largest state?/Alaska/What's its capitol?/Juneau/

Large quantities of text for testing are now available on the Internet, though not exactly in this form. Nevertheless, a model that does well at predicting successive characters in the dialog above must have solved the same types of problems needed to do well on strings such as

The largest state in the U.S. is ______

Arguments for the Compression Test

We present three arguments for compression. First, compression is more stringent than the Turing test. Second, prediction requires observable knowledge, as evidenced by its use by people. Third, we find that compression, like AI, is unsolved.

1. Equivalence to the Turing Test

We proposed using text prediction to test for artificial intelligence, with compression (using optimal encoding) as a measure. We now show that compression is a more stringent test than the Turing test. An optimal solution to modeling a human language L under compression is an exact model M = L. This model will be indistinguishable from human, so should pass the Turing test by being misjudged 50% of the time. However, there are two conditions under which a machine will be misjudged more than 50% of the time. One is when the human confederate and judge use language models that are more disparate than the judge and machine. The second is when the machine, rather than giving a human distribution of responses y to input x, $P(y) = L(xy)/L(x)$, it favors the most likely response, $\max_y L(xy)/L(x)$. In other words, just because people sometimes give the wrong answer doesn't mean that a machine must make the same mistakes.

The Turing test depends on the human ability to recognize intelligent behavior. It assumes that, if given two dialogs x and y, then people can estimate, say, $L(x) > L(y)$, and conclude that x is more likely to be generated by a human than y. Converting generation to prediction in a Markov process is a simple transformation (Abramson 1963), but it doesn't mean that people can do it. Nevertheless, there is strong evidence from psycholinguistics (Hörmann 1979) that people do indeed use the same distribution L to generate and recognize (and store) messages:

- People make fewer speech recognition errors in a noisy environment when the speech consists of meaningful sentences rather than nonsense syllables, or when the number of possible words is reduced (Miller, Heise, Litchen 1951, cited by Hörmann p. 86).
- People perform better in short-term recall tests on random sequences of words when the order of approximation to English increases (Miller and Selfridge 1950, cited by Hörmann p. 104-105).
- People perform worse in short-term recall tests if syntactic constraints are removed by randomly reordering words, or if lexical constraints are removed by using nonsense words (Epstein 1962, cited by Hörmann p. 205). The following example used in the study obeys syntactic but not lexical constraints: *A haky deebs reciled the dison tofently um flutest pav.* The following obeys lexical and syntactic but not semantic constraints: *Wavy books worked singing clouds to empty slow lamps.*

Thus, recall and recognition improve as the entropy of the generator decreases, evidence that the same language model is used in all three processes. (The short-term recall experiments, which are not directly relevant to the Turing test, suggest a fixed storage capacity of 50 to 100 bits after compression).

The Turing test involves a machine, a human confederate, and a judge who must decide which is which based on the two dialogs between the judge and each agent. Suppose that the judge prepares a list of n questions, $j_1, j_2, ..., j_n$ for the two agents, generating two dialogs, $jc = j_1c_1j_2c_2...j_nc_n$ with the confederate and $jm = j_1m_1j_2m_2...j_nm_n$ with the machine. The three languages used to generate the dialogs are L_J for the judge, L_C for the confederate, and L_M for the machine. The judge calculates the probabilities P(jc is human) and P(jm is human), knowing only that one is human and the other is not. In a test with many agents, the probabilities would be used to rank them. In a two-agent test, we would say that the machine has artificial intelligence if odds(jm is human) ≥ 1, where odds(x) ≡ P(x)/(1 – P(x)).

The judge does not know L_C or L_M, so uses L_J as an estimate of the human language L_C, and makes no assumptions about L_M. Let $L_J(c)$ and $L_J(m)$ denote the probabilities that L_J would generate the responses $c = c_1, c_2, ..., c_n$ and $m = m_1, m_2, ..., m_n$ respectively, if given inputs $j = j_1, j_2, ..., j_n$.

$$L_J(c) \equiv \prod_{i=1}^{n} \frac{L_J(j_1c_1 \cdots j_ic_i)}{L_J(j_1c_1 \cdots j_i)}$$

and similarly for m. Then the odds that jm is human, given what the judge knows, is

$$odds(human(jm)|L_J, jm, jc) = \frac{L_J(m)}{L_J(c)}$$

This can be shown as follows (implicitly assuming L_J):

$$\begin{aligned} &P(human(jm)|\, jm, jc) \\ &= \frac{P(jm, jc|human(jm))\,P(human(jm))}{P(jm, jc)} \\ &= \frac{L_C(m)L_M(c)\frac{1}{2}}{\frac{1}{2}(L_C(m)L_M(c) + L_M(m)L_C(c))} \\ &= \frac{L_J(m)}{L_J(m) + L_J(c)} \end{aligned}$$

$$odds(human(jm)|\, jm, jc, L_J) = \frac{\frac{L_J(m)}{L_J(m)+L_J(c)}}{\frac{L_J(c)}{L_J(m)+L_J(c)}} = \frac{L_J(m)}{L_J(c)}$$

In the first step, we apply Bayes law. In the second step, we use the independence of jm and jc to conclude P(jm,jc|human(jm)) = P(jm|human(jm))P(jc|human(jm)). The last step of the first equation applies because the judge uses L_J as an estimate of L_C, and knows nothing about L_M. Thus, as far as the judge knows, $L_M(m) = L_M(c)$, and the $L_M(c)/2$ can be factored out. In practice, a judge may decide that one response is more "machine-like" than another, but a fair test requires that there be no communication between judge and agents prior to the test, so no assumptions about machine behavior are justified.

Next, we ask what conditions maximize the probability of fooling the judge. Given the languages L_J, L_C, L_M, and the knowledge that jm is not human, we find the expected value of the log of the odds of fooling the judge over all possible dialogs. Note that log(odds(x)) is a strictly increasing function of P(x).

$$\begin{aligned} &E_{jm,jc}\left[\begin{array}{l}\log_2(odds(human(jm)|\, jm, jc, L_J)) \\ |\, L_J, L_C, L_M, \neg human(jm)\end{array}\right] \\ &= \sum_m \sum_c L_M(m)L_C(c)\log_2 \frac{L_J(m)}{L_J(c)} \\ &= \sum_m \sum_c L_M(m)L_C(c)(\log_2 L_J(m) - \log_2 L_J(c)) \\ &= \sum_m L_M(m)\log_2 L_J(m) - \sum_c L_C(c)\log_2 L_J(c) \\ &= -H_J(L_M) + H_J(L_C) \end{aligned}$$

The first step takes the weighted average over all possible dialogs, which are distributed according to L_M and L_C. The third step uses the fact that $\Sigma_x L_C(x) = \Sigma_x L_M(x) = 1$.

We pass the Turing test when the difference of the cross entropies is nonnegative. This happens when the odds of fooling the judge are at least 1 (probability 1/2). This condition holds when the model is exact, $L_M = L_C$. However, this solution is not optimal. First, any error in the judge's approximation of the confederate's language will increase the cross entropy, $H_J(L_C)$. Second, since we are varying L_M rather than L_J, we can minimize $H_J(L_M)$ by favoring the most common responses, setting $L_M(m) = 1$ for the largest value of $L_J(m)$ and $L_M(m)$ to 0 for all other m.

2. Knowledge as a Property of Language

It is apparent that people use both low and high level knowledge to predict successive characters in natural language text. For instance, to complete the string *roses are r_*, we combine knowledge at several levels. From lowest to highest:

- Lexical – A leading *r* is usually followed by a vowel.
- Syntactic – An adjective is likely after *are*.
- Semantic – Roses have physical properties such as color.
- Real-world – Roses are red (except in Texas).

By using a probabilistic language model, we implicitly assume that any knowledge that is relevant to a text-based AI system is a statistical property of the language in which it is expressed. As evidence, we simply observe that the most common statements in any natural language are those that are lexically, syntactically, semantically, and factually correct, such as *roses are red*.

People use *expressible* knowledge to predict text. Only expressible knowledge is relevant to a text-based AI system, because only expressible knowledge is observable. Examples of inexpressible knowledge might be a description of a person's face, the taste of a banana, or the skills needed to ride a bicycle. It is difficult to imagine questions that could test for any knowledge that couldn't also be taught through words.

Some knowledge can be learned both verbally or nonverbally; a blind person can know the color of roses even if the information is meaningless. Such knowledge is still expressible, and therefore testable in an AI system and relevant to text prediction.

3. Human vs. Machine Prediction

We now compare text prediction in humans and machines, with the reassuring result that humans are superior. Shannon bounded the entropy of written English by having people solve character-completion puzzles (Shannon 1951). Text samples were taken from a variety of sources, such as classic literature or technical manuals. The text was reduced to a 27 character alphabet of monocase letters and spaces. Subjects guessed at each letter until correct, and were allowed to use references such as dictionaries and character frequency tables. Shannon estimated that the entropy or uncertainty of written English is between 0.6 and 1.3 bits per character.

The uncertainty in Shannon's measurement results because many different rankings can result in the same probability distribution. Cover and King (1978) had subjects assign probabilities directly in a betting game, and obtained an upper bound of 1.3 bits per character. Tan (1981) used this technique to obtain 1.3 bits per character for Malay text (which has the same alphabet as English).

These entropy measurements were compared with the compression ratios obtained using the best known data compression programs. Three English text files were reduced to 27 characters (lower case letters and spaces) to reproduce the conditions under which the human tests were performed. In theory, a compression ratio of 1.3 bits per character should be possible. In reality, the best program averaged 1.87 bits per character on the three test files. Results are shown in table 1, along with compression results for the Calgary Corpus (1993), a widely-used benchmark of 14 text and binary files in a variety of formats totaling 3,141,622 bytes. The test files were:

- *alice* – Alice in Wonderland (Carroll 1865) from the Gutenberg press. The legal header was removed. Upper case characters were replaced with lower case, and then all nonalphabetic character sequences were replaced with a single space. The resulting file was 135,059 characters.
- *hardy* – from the file *book1* in the Calgary corpus, "Far from the Madding Crowd", by Thomas Hardy. SGML tags (enclosed in <angle brackets>) were removed, and the file processed as above. The file was reduced from 768,771 to 729,966 characters.
- *witten* – from the file *book2* in the Calgary corpus, "Principles of Computer Speech" by Ian Witten, in UNIX *troff* format. All lines beginning with a period, or that contained characters other than letters, spaces or the punctuation characters . , ? ! ; ' and " were removed. This removed most *troff* codes, tables, and mathematical formulas, leaving mostly readable English text. The file was then reduced to 27 characters as in *alice*, reducing it from 610,856 to 315,749 characters.

The compression programs tested are of two types, Ziv-Limpel (LZ) and prediction by partial match (PPM), both described in (Bell, Witten, Cleary 1989). LZ compressors are the most popular, due to their high rate of decompression, but PPM achieves a better compression ratio. UNIX *compress*, *pkzip* (PKZIP 1993), and *gzip* (Gailly 1993) are all LZ compressors.

All compression algorithms exploit the lexical redundancy found in most files, including text files; the tendency of some character strings to occur more often than others. In an LZ compressor, the second and subsequent occurrences of a substring are replaced with pointers to a previous occurrence whenever the pointer can be encoded using fewer bits.

A predictive arithmetic encoder (a class that includes PPM) has two parts, a *predictor*, and an *encoder*. The predictor assigns a probability $P(x_i)$ to each successive character x_i, given the previous text, $x_1x_2...x_{i-1}$. The encoder assigns a code of length $\log_2 1/P(x_i)$ bits. Using arithmetic encoding, it is possible to effectively assign fractional code lengths by assigning a single code of length $\lceil \log_2 1/P(x) \rceil$ bits to the entire file, x. An order-n PPM encoder uses only the context of the last n characters, $x_{i-n}...x_{i-1}$ to determine $P(x_i)$, based on statistics from previous occurrences of the same context. Going beyond order-4 or 5 rarely helps. It has been shown that LZ is a special case of predictive encoding (Bell, Witten, Cleary 1989).

The following compression programs were tested. Options shown were selected for maximum compression when possible. For archivers, which compress multiple files into a single file, the overhead of storing the filename, date, checksum, etc., is not included in table 1 for individual files.

- *pkzip* version 2.04e. A popular LZ archiver (PKZIP 1993).
- *gzip386 -9* version 1.2.4. An LZ compressor equivalent to *gzip* (but faster) for DOS on x386 and higher processors (Gailly 1993).
- *ha a2* version 0.98, an order-5 PPM archiver (Hirvola 1993), the best compression on the Calgary corpus (apparently as of 1993) according to (Data compression FAQ 1998), and exceeding the best reported by (Bell, Witten, Cleary 1989).
- *ppmz* version 9.1, a PPM compressor, reported to be the best known, improving on *ha* (Bloom 1998). Decompression failed, so results could not be verified,

although sizes are consistent with those reported by the author for files from the Calgary corpus.

- *boa -m15* (maximum memory option, 15MB), version 0.58 beta, an archiver, probably PPM, although the documentation gives no details (Sutton 1998). Best compression known as of Sept. 1998 on both the Calgary corpus and on text, according to the Archive Comparison Test (Gilchrist 1998), and improving on *ha* and *ppmz*.

	pkzip	**gzip**	**ha**	**ppmz**	**boa**
alice	2.616	2.301	1.989	1.877	1.871
hardy	2.955	2.922	2.028	failed	1.957
witten	2.511	2.482	1.828	1.688	1.682
calgary	2.629	2.591	2.155	1.935	1.913

Table 1. Data compression results, bits per character.

The PPM encoders, led by *boa*, give the best compression, 1.86 bits per character over the three text files.

Conclusion

We described the technique of using data compression on a corpus of text to measure the intelligence of a language model of an AI system. Like the Turing test, we use a strictly behavioral definition of intelligence. The Turing test says that a machine is intelligent if people believe it to be human. A data compression test says that a machine is intelligent if it predicts text as well as a human. We showed that the compression test is more stringent than the Turing test. A compression test is also objective, quantitative, and easier to implement.

In our proof, we reduced observable human knowledge to a statistical property of language, devoid of meaning. This reduction is supported both by psycholinguistic evidence and by the probabilistic language model as a generalization of formal computation.

One objection to considering compression as an AI task is that unlike other problems, people aren't good at it. That is because decompression requires an exact copy of the prediction model used during compression, and we cannot copy the human brain.

The equivalence of compression and AI means that advances in either field could be applied to the other. For AI, it suggests applying statistical approaches to knowledge, learning, and reasoning. For compression, it suggests that better results could be obtained by adding syntactic and semantic rules.

As a final check, we compared human and machine models in text prediction. It is somewhat comforting to find that compression, like AI, is still unsolved.

References

Abramson N., 1963. *Information Theory and Coding*, New York: McGraw-Hill.

Bell, T., Witten I. H., Cleary J. G., 1989. Modeling for Text Compression. ACM Computing Surveys 21:557-591.

Bloom, C., 1998. Solving the Problems of Context Modeling. http://www.cco.caltech.edu/~bloom/papers/ppmz.zip

Calgary Corpus 1993. http://www.kiarchive.ru/pub/msdos/compress/calgarycorpus.zip

Carroll, L., 1865. *Alice in Wonderland*. Gutenberg Press, ftp://sunsite.unc.edu/pub/docs/books/gutenberg/etext97/alice30h.zip

Cover, T. M., and King, R. C., 1978. A Convergent Gambling Estimate of the Entropy of English. IEEE Transactions on Information Theory 24:413-421.

Data Compression FAQ, 1998 (July 25). http://www.cs.ruu.nl/wais/html/na-dir/compression-faq.html

Flinders University, 1998. The Flinders University of South Australia presents the 1999 Loebner Prize Competition. http://www.cs.flinders.edu.au/Research/AI/LoebnerPrize/

Gailly, J. 1993, gzip 1.2.4, http://www.kiarchive.ru/pub/msdos/compress/gzip124.exe

Gilchrist, J. 1998. Archive Comparison Test, http://www.geocities.com/SiliconValley/Park/4264/act-mcal.html

Hirvola, H., 1993. ha 0.98.

Hörmann, H., 1979. *Psycholinguistics*, 2nd Ed., New York: Springer-Verlag.

Hutchens, J., and Alder, M., 1997. Language Acquisition and Data Compression. 1997 Australasian Natural Language Processing Summer Workshop Notes. 39-49.

Hutchens, J, and Alder, M. 1998a. Finding Structure via Compression. Proceedings of the International Conference on Computational Natural Language Learning. 79-82.

Hutchens, J., and Alder, M, 1998b. Introducing MegaHAL. Proceedings of the Human-Computer Communication Workshop. 271-274.

PKZIP 1993, version 2.04e, PKWARE Inc.

Raymond, E. 1997. Jargon Dictionary, http://www.netmeg.net/jargon/terms/a/ai-complete.html

Rich, E, and Knight, K., 1991. *Artificial Intelligence*, 2nd Ed., New York: McGraw-Hill.

Shannon, C., and Weaver W., 1949. *The Mathematical Theory of Communication*. Urbana: University of Illinois Press.

Shannon, C. 1951. Prediction and Entropy in Printed English. Bell Sys. Tech. J 3:50-64.

Shieber, S. M., 1994. Lessons from a Restricted Turing Test. http://xxx.lanl.gov/abs/cmp-lg/9404002

Sutton, I., 1998. boa 0.58 beta, http://webhome.idirect.com/~isutton/

Tan, C. P., 1981. On the Entropy of the Malay Language. IEEE Transactions on Information Theory 27:383-384.