

Can Computers Create Humor?

Graeme Ritchie

■ *Despite the fact that AI has always been adventurous in trying to elucidate complex aspects of human behavior, only recently has there been research into computational modeling of humor. One obstacle to progress is the lack of a precise and detailed theory of how humor operates. Nevertheless, since the early 1990s, there have been a number of small programs that create simple verbal humor, and more recently there have been studies of the automatic classification of the humorous status of texts. In addition, there are a number of advocates of the practical uses of computational humor: in user interfaces, in education, and in advertising. Computer-generated humor is still quite basic, but it could be viewed as a form of exploratory creativity. For computational humor to improve, some hard problems in AI will have to be addressed.*

Although the terms *creative* and *humor* are both very broad in what they can cover, there is little doubt that the construction of humor is generally regarded as creative, with the most successful creators of humor sometimes being hailed as “comic geniuses.” This suggests that the task of getting a computer to produce humor falls within the area of computational creativity, and any general theory of creativity should have something to say about humor.

This article reviews some of the work in computational humor, makes some observations about the more general issues that these projects raise, and considers this work from the viewpoint of creativity, concluding with an outline of some of the challenges ahead.

The Place of Humor within AI

The use of humor is a major factor in social and cultural life throughout the world, influencing human behavior in various ways (as well as giving rise to the whole industry of comedy). Despite this, the cognitive and emotional processes involved in creating or responding to humor are still unexplained. Thus humor constitutes a complex but baffling human behavior, intimately related to matters such as world knowledge, reasoning, and perception; on the face of it, this should be an obvious target for AI research. After all, the remit of artificial intelligence, since its very beginning, has been to make computers behave in the ways that humans do. The proposal for the historic Dartmouth Conference in 1956, often taken as the founding event in AI, argued:

... that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be

made to simulate it. (Charniak and McDermott 1985, 11)

Despite this, “intelligence” was rather narrowly understood as being primarily about reasoning and about acquiring concepts (learning). Emotional aspects of human behavior were not deemed a suitable matter for AI research until about 1990, and mainstream work in AI completely ignored humor for the first 35 years or so of the field’s existence. Two major encyclopedias in AI and cognitive science (Shapiro 1992, Wilson and Kidd 1999) have no sections or index entries for “humo(u)r,” and there is no sign of this topic in the table of contents of a more recent reference work (Dopico, Dorado, and Pazos 2008). Of course, it could be argued that the creation of humor is not truly “intelligence,” but it is certainly a complex and subtle form of human behavior, with both cognitive and emotional aspects, the mechanisms for which are far from obvious. AI no longer avoids the topic of humor completely, but computational modeling of humor is still at an early stage, with few established precepts or results. In assessing the state of the field, it is important to realize that this subarea of AI has received few resources and relatively little effort. Hardly any computational humor projects have received external funding from mainstream grant-awarding bodies (probably the largest examples are those described by Stock and Strapparava [2003] and by Manurung et al. [2008]).

Where’s the Theory?

The centuries preceding the birth of AI witnessed discussions of humor by such famous names as Cicero, Kant, Schopenhauer, and Freud (see Morreall [1987] and Attardo [1994] for historical reviews) so it might seem that the starting point for computational work should be the current theory of humor. Why should AI researchers try to reinvent what these great minds have provided? Unfortunately, it is not as simple as that. Although there are many books and articles on humor from the perspective of philosophy and literature (and—more recently—of psychology), none of that has resulted in agreement about a theory of humor. Nor is it the case that there are several competing theories to choose from, if “theory” means a detailed, precise, coherent, empirically grounded set of principles that explain the known phenomena. Although the published works on humor provide stimulating ideas about what factors may be involved in humor, they do not provide the level of detail, or the clear definition of concepts, that would be required to formulate a computer model. In view of this lack of a directly usable framework, researchers have applied conventional AI methodologies to the problem of humor, developing mod-

els that are very limited in their scope, but that can at least be implemented. Such programs typically make little or no use of theories of humor. Occasionally, computational humorists will acknowledge some theoretical framework, but it can be hard to see how the actual details of the implementations follow from the theoretical ideas cited. An exception to this trend is the program of Tinhoft and Nijholt (2007), which uses a mechanism derived from the Raskin (1985) criterion for a text being a joke; this is discussed in more detail later on.

Humor-generating programs are typically designed by examining some extremely small area of humor, abstracting some structural patterns from existing examples (for example, jokes found in joke books), formulating computationally manageable rules that describe these patterns, and implementing an algorithm to handle these rules.

Does this mean that computational projects cannot contribute to wider theories of humor? Opinions are divided on this. The implementors of the existing programs might argue that even very small studies can throw up interesting insights into the nature of humor, gradually chipping away at the mystery and slowly extending the scope of the investigation to wider and more general areas; this position is espoused by Ritchie (2004). Sceptics would argue that these tiny implementations are overly narrow and based on extremely ad hoc mechanisms, which means that they tell us nothing about any other forms of humor. The latter position has been vigorously expressed by Raskin (1996, 2002), who argues that only by adopting linguistically general and theoretically based concepts can computational humor progress.

Humor in Language

Humor out in the everyday world appears in many forms: visual gags, physical slapstick, funny sound effects, unexpected twists in situation comedies, and so on. However, it is very common for humor to be conveyed in language: funny stories, witty sayings, puns, and so on. A great deal of academic research into humor has focused on humor expressed in language, and this is not surprising, as there are some very good reasons for concentrating on this genre. Even without the added complications of vision, physical movement, and so on, verbally expressed humor still displays a very rich and interesting range of phenomena to study, and the surrounding culture provides a wide variety of established genres for textual humor, with lavish quantities of data available for study (for example, joke books, collections of aphorisms). This means not only that there is a plentiful supply of data, but also that the examples are *attested*, in the sense that the researcher knows that the particular texts are

recognized by society as actually being instances of humor. Data (that is, examples of humor) are much easier to record and pass around in the form of texts. In contrast, complete depictions of acts of slapstick (particularly if not performed for the camera) are harder to set down. Moreover, there are relatively good formal models of the more obvious aspects of language, because linguists have developed a wealth of terminology and concepts for describing the structure of chunks of text (spoken or written). This means it is much easier to provide a detailed account of the contents of a humorous passage.

These advantages apply even more strongly when it comes to the computational treatment of humor, as it is important to be able to represent the data in machine-readable form, and it is helpful to have some formal model of the structure of the data items. All the working humor programs discussed here handle textual humor only; there are almost no computer programs dealing with non-textual humor (but see Nack [1996] and Nack and Parkes [1995] for another approach).

Even within verbally expressed humor, there is a further subclassification routinely made by humor scholars: humor created by the language form itself (often called *verbal* humor) and humor merely conveyed by language (often called *conceptual* or *referential* humor). Attardo (1994) cites many authors, as far back as Cicero, who have made this distinction. Verbal humor depends directly on the linguistic features of the words and phrases used: puns, use of inappropriate phrasings, ambiguities, and so on. Such humor relies on the particular characteristics of one language, and so may be difficult or (usually) impossible to translate. In contrast, conceptual or referential humor involves all sorts of events and situations and may be conveyed in any medium: cartoons, video, text, speech, and so on. Where a textual joke depends on referential humor, it can generally be translated into another language without distorting or losing its humorous properties.

Verbal Humor

Traditional symbolic AI can be seen as an attempt to reduce some complex behavior (game playing, reasoning, language understanding, and so on) to the manipulation of formal relationships between abstract entities. Computational humor programs are no exception. They are all based on the assumption that the humorous items under consideration can be characterized by a statement of various formal relations between parts of those items or some underlying abstract entities. The interesting question is: what sorts of entities and relations are involved? The earliest humor-generating programs (see Ritchie [2004, chapter 10] for a review) were founded on the observation that

some very simple types of wordplay jokes (puns) rely on linguistic properties that are structural and relatively close to the “surface” (that is, actual words and grammatical units) rather than being “deeper” (for example, abstract concepts of incongruity or of social appropriateness).

Probably the earliest actual joke generator is that of Lessard and Levison (1992), which built very simple puns such as this one, in which a quoted utterance puns with an adverb:

“This matter is opaque,” said Tom obscurely.

The next stage of development was *punning riddles*, which began emerging from computer programs in 1993. A punning riddle is a short joke that consists of a question and a one-line answer, with some sort of wordplay involving the answer. Another very small text generator by Lessard and Levison (1993) produced examples such as the following:

What kind of animal rides a catamaran?

A cat.

Around the same time, the joke analysis and production engine (JAPE) program (Binsted 1996; Binsted and Ritchie 1994, 1997) was constructed. It could spew out hundreds of punning riddles, one of the better ones being:

What is the difference between leaves and a car?

One you brush and rake, the other you rush and brake

JAPE was a conventional rule-driven program, implemented in Prolog, using a relatively small number of pattern-matching rules to compose the riddle structures and using a conventional dictionary based on WordNet (Miller et al. 1990; Fellbaum 1998) to supply words and two-word phrases. The large size of the dictionary (more than 100,000 entries) meant that JAPE could generate a vast number of texts, although only a few of them were good jokes, and many were incomprehensible (Binsted, Pain, and Ritchie 1997).

A few years later, Venour (1999) built a pun generator that could construct simple two-word phrases punning on a preceding setup sentence, as in the following:

The surgeon digs a garden. A doc yard.

Another small-scale pun generator was that of McKay (2002), whose program could build simple one-line puns like the following one:

The performing lumberjack took a bough.

All the early pun generators were programmed without reference to any general plan of how a punning program should be constructed (since there was no precedent), and there was no obvious commonality in their components. The architecture adopted by the JAPE program, however, has a relatively clean modular structure (and formed the basis for a later, larger system, System to Augment

Nonspeakers Dialogue Using Puns (STANDUP) [Manurung et al. 2008]). With hindsight, this architecture could be used quite naturally to produce the output of the other simple pun builders.

Although most computer-generated verbal humor consists of puns, there has also been a non-punning system that may lie on the boundary of verbal and referential humor: the HAHAcronym program (Stock and Strapparava 2003, 2005). This system can perform two tasks: given an existing acronym (for example, *DMSO*, for *defense modeling and simulation office*), it computes alternative words with these initials to form a whimsical phrase (for example, *defense meat eating and salivation office*). Alternatively, given two words that indicate some concept, it finds an English word pertinent to the concept whose letters are the initials of a phrase also (humorously) relevant to the concept. For example, *{fast, processor}* yields *TORPID*, for *Traitorously Outstandingly Rusty Processor for Inadvertent Data processing*.

Referential Humor

On the whole, computational humor has concentrated, so far, on verbal humor, although there have been a few small forays into referential humor. Bergen and Binsted (2003) examine a subclass of joke based on exaggerated statements on some scale, such as the following, and Binsted, Bergen, and McKay (2003) briefly allude to implementing a generator of such jokes.

It was so cold, I saw a lawyer with his hands in his own pockets.

Stark, Binsted, and Bergen (2005) describe a program that can select (from a short list of options) the most suitable punchline for a joke of the type illustrated as follows:

I asked the bartender for something cold and filled with rum, so he recommended his wife.

However, this appears to be a very small implementation that works on one or two examples using a very specific knowledge base.

Tinholt and Nijholt (2007) describe a small prototype system for generating jokes in the form of deliberately misunderstood pronoun references (see also Nijholt [2007]). Such humor is exemplified by the following exchange within a Scott Adams cartoon:¹

“Our lawyers put your money in little bags, then we have trained dogs bury them around town.” “Do they bury the bags or the lawyers?”

(The actual cartoon has a further punchline: “We’ve tried it both ways.”) Tinholt and Nijholt’s process works as follows. If a pronoun occurs within a text, a general antecedent-finding program (Lappin and Leass 1994) fetches all candidate antecedents, including the proposed correct one. The correct antecedent is compared with each of

the others to find an incorrect candidate that is “opposed” to the correct one. Two potential referents are deemed to be opposed if they have most of their properties in common according to ConceptNet (Liu and Singh 2004) but also have at least one property with opposite (antonymous) values. If these criteria are met, the program generates a pseudoclarificatory question asking which of these two potential antecedents is involved in the relation mentioned in the text. Tinholt and Nijholt tested this method on chatterbot transcripts and story texts, comparing its identification of joke opportunities with a manual annotation of the texts. They state that it was “still quite susceptible to errors” so that it was “unfeasible to use the system in existing chat applications,” but opine that this method “does provide future prospects in generating conversational humor.”

Could a Computer Learn to Make Jokes?

A dominant paradigm in AI at present is that of machine learning (in various forms). This raises the question: could a learning program develop joke-making skills? There are certainly plenty of positive instances (jokes) to use for training.

Mihalcea and Strapparava (2006) carried out a number of studies in which various machine-learning and text-classification methods were applied to large quantities of textual data, both humorous and nonhumorous. The aim was to see whether it was possible to evolve classifiers (of various kinds) that distinguish humorous texts from nonhumorous. They used a collection of one-liner jokes found on the web and four nonhumorous datasets: Reuters news titles, proverbs, ordinary sentences from a large corpus of English (the British National Corpus [Leech, Rayson, and Wilson 2001]), and “commonsense” statements from a knowledge base (Singh 2002). Using certain stylistic features (alliteration, antonymy, adult slang), their program automatically constructed a decision tree for each pair of humorous data and nonhumorous data sets (that is, there were four different comparisons). The accuracy of the classification learned varied from 54 percent (humor versus proverbs) to 77 percent (humor versus news titles). They also tried using aspects of the content of the text, with two other classifier techniques (naïve Bayes, support vector machine), obtaining broadly similar results in the two cases: accuracy of 73 percent for humor versus ordinary sentences, and 96 percent for humor versus news titles. Combining the stylistic and content features hardly improved performance at all. Mihalcea and Pulman (2007) discuss these results, concluding that certain features show up as more salient in the computed classifications: human-centered vocabulary, negation,

negative orientation, professional communities, and human weakness.

Mihalcea and Strapparava's experiments suggest that humorous texts have certain gross textual properties that distinguish them from other texts, to the extent that it might well be possible to have a program guess, with much better accuracy than chance, whether a text was a joke or not. While this is interesting, there is some way to go before such findings can give rise to a humor-creation system (and this is not what Mihalcea and Strapparava were proposing). It is possible to imagine some form of "generate-and-test" architecture, or an evolutionary algorithm, into which a text classifier might be fitted. Even if such an arrangement could be made to work, it would be quite hard to argue that this demonstrates creativity of any depth.

The main problem is that the automatic classifiers detect correlations (loosely speaking) between certain features and humor but do not determine causality from features to humor. A text could be replete with the types of feature listed by Mihalcea and Pulman without being funny.

The Methodology of Humor Generation

Although it is exciting to think of humorous programs as being models of creative or witty humans, perhaps even displaying a "sense of humor," there is a less glamorous, but potentially very helpful, perspective on the relationship between computation and humor research. Most of the generating programs built so far have not been embedded in some larger cognitive model, do not function in an interpersonal context, and perform no specific application task (although Tinholt and Nijholt's pronoun misunderstander attempts to move outside these limitations). Instead, they simply follow symbolic rules to produce output. In this way, they function as testbeds for whatever formal models of joke structure are embodied in their implementation. Just as there was once a vogue for "grammar-testing" programs that checked the consequences of a linguist's proposed formal grammar (Friedman 1972), a joke generator tests the implications of a formal model of joke mechanisms. Although the programs do not yet connect to general theories of humor, this testbed role would be very valuable where the implementation was based on a proposed theory.

When testing a formal model, the program can be run under laboratory conditions, since it is not intended as a simulation of a human joke teller in a real context. Under these circumstances, certain methodological principles follow (some of which are more generally applicable to creative programs).

The first issue is the status of humorous *input* to the program. If the system is some sort of learning

program, which takes in a collection of existing humorous items and, by some process of generalization, manages to construct further humorous items that are noticeably distinct from the original input, then that is a significant achievement. (It is also one of the types of situation that stimulates interesting debates about ascribing creativity to programs [Ritchie 2007]).

On the other hand, if the program merely recycles the input items, perhaps with some superficial rearrangement or packaging, then that is not very interesting, as it casts no light on how humor can be created. Hence, programs that do not aspire to the kind of learning model just sketched should not accept humorous items within their input. This is because a program can embody a computational model of humor creation only if it constructs the humor from nonhumorous material. The whole point of developing models of humor, in the general tradition of reductionist science, is to decompose the nature of humor creation into some configuration of other (nonhumorous) processes. Furthermore, there should not be some module within the system whose function is to create humor and that is a "black box," in the sense that its inner workings are not known, or not documented, or not regarded as part of the proposed model. If the reason that the program manages to produce humor is because it has a box inside it that (mysteriously) "creates humor," that would not illuminate the problem of humor creation. Instead, this would introduce a form of regress, in which there is now a new problem: how does that black box achieve its results?

This is subtly different from allowing the program to have knowledge that is about how to create humor, perhaps localized to one module. That is completely acceptable: somewhere in the program there must be a capability to form humorous items. If the model is to cast light on the nature of humor, it is essential to know what that knowledge is and how it is used. This means that the builder of the humor generator should be quite clear about which parts of his system, or which of the knowledge sources upon which the program relies, or which interactions between modules are regarded as central to creating the humor and which are merely supporting facilities.

This is illustrated in the early pun generators. These systems crucially had to find words that were in a well-defined relationship, such as sounding alike. These items are the core of the pun, but other text must be created around them to set up the pun. Within certain constraints on the internal coherence of the overall text, there could be many such setups for the same central pun. For example, given two core lexical items in which one sounds similar to the start of the other, such as *pylon* and *pa*, the answer for a punning riddle can be con-

structed, “*a pa-lon*,” and the question for the riddle can be framed by asking about some entity that has properties of both the named concepts. This can be done using other words that are semantically related to these two words, such as hypernyms or synonyms, but the question can be formed in various ways, such as “*what kind of tower is a dad?*” “*what do you get when you cross a tower with a father?*” “*what do you call a tower with a family?*” and so on.

Hence the pun generators can be viewed as having central modules that select suitable words and set out the minimum needs for the setup, but there are also basic text-building modules that supply the surrounding context to meet that requirement.

Any purely supporting facilities should not only be nonhumorous, they should, as far as possible, be general purpose. That is, they should not be too finely tuned for performing the supporting role in the program, since there is a risk that such tuning might stray accidentally into covert injection of humorous content in a way that would obscure the real contributions of the parts of the model. Although the text-generation capabilities of past pun generators are usually not wholly general text-generation programs, they are not fine-tuned *for puns*, in that there are conventional language-generation programs, for real practical applications, that use equally crude and narrowly targeted rules (Reiter and Dale 2000). Lessard and Levison, and Venour, made use of a general text generator, Vinci (Levison and Lessard 1992), and the the lexicon used by Binsted was a wholly general-purpose dictionary (Miller et al. 1990) containing the kind of information that linguists would normally associate with words and phrases (grammatical categories, relations such as synonymy, and so on). It might be that *all* the knowledge resources are “supporting,” in the sense that the humor is created entirely from the interaction of various types of nonhumorous knowledge; in that case, the “recipe for humor” would be in the interaction (as defined by the program designer), and this should be made completely explicit and open to scrutiny.

The testing of humor-creating programs is also a matter worth reflecting upon, if such evaluations are to illuminate the phenomenon of humor. The pun generators summarized earlier do not have any real notion of an “input parameter”—they simply run with all their knowledge bases every time. In such cases, there is no question of selecting values for the input (beyond the general hygiene principles outlined above, concerning the nature of the knowledge bases). However, if the program can be run on specific data items—as in the case of the HAHAcronym system—then any trials of the system must have a rational and documented way of selecting the values used. For example, the input values might be randomly selected

from the set of all possible values, or the data items might be selected according to some principles, such as featuring in certain linguistic sources in some particular way (for example, frequency of occurrence in a large collection of naturally occurring texts). What should not be done, in the interests of objective testing, is for the parameters to be selected in an ad hoc manner by hand, as there is the risk of subjective estimates of “suitable values” influencing the choice.

If a system produces only a small number of items, then there is no problem about deciding which of these to test: all of them can be tested, giving an exhaustive assessment of the program’s output. If, on the other hand, the system can produce large amounts of data then the next question is: which output items are to be subjected to testing? To avoid bias creeping in, some form of random selection is probably necessary. If the testing mechanism is constrained in some way (for example, only certain other texts are available as control items), it may be necessary to restrict the choice of output data for compatibility with the available control items, but the selection should be as objective as possible.

The most obvious, and still the best, way to test humor output is with human judges. The exact details of how to elicit judgments about the texts from appropriate humans (for example, children for child-oriented humor) need to be planned, but this area is not mysterious, as this type of study is routine within psychology. There should be control items alongside the program output items, the order of presentation of items should be varied, the mix of types of items should be balanced, the questions to the judges should be carefully phrased, there should be a sufficiently large number of subjects, and so on. There is thus no obstacle in principle to assessing the quality of computer-generated humor, but most implementations have been on such a small and exploratory scale (for example, as student projects) that there have been few thorough evaluations. Probably the fullest and most systematic testing was the evaluation of the JAPE program (Binsted, Pain, and Ritchie 1997).

Practical Applications

Science is not the only motivation for computational modeling of humor: there is also engineering. From an early stage, there has been the idea that computer-generated humor might be of practical use. It is clear that AI is a long way from the automated writing of comedy scripts or writing gags for comedians, but there have been some proposals for the use of humor for real-world tasks. Strapparava, Valitutti, and Stock (2007) describe some explorations of the possibility of using computational pun making in the “creative” develop-

ment of an advertising campaign, but as yet that area is unexploited. The two main types of application that have been advocated and at least tentatively explored are friendlier user interfaces and educational software.

Binsted (1995) considered whether a natural language user interface for a computer system would be more congenial if the system could make use of humor to lessen the impact of inadequacies in its performance (for example, lack of factual knowledge, poor language understanding, slow response). She suggested a number of types of humor that might work: self-deprecating humor (the system mocking its own failings), wry observations (for example about the system's lack of speed), whimsical referring expressions. Stock (1996) developed this theme and, in passing, mentioned the possible application areas of advertising and "edutainment."

These two essays were essentially speculative and preceded the experimental results of Morkes, Kernal, and Nass (1999), who found that users reacted more positively to a computer in which the user interface contained examples of humor (not computer generated, sadly). Nevertheless, Binsted and Stock raise some valid general points. The humor will probably be more effective if it is attributed to some sort of quasi-human agent that is not synonymous with the entire computer system. In that way, frivolous wisecracks can be seen as separate from the main task, and any failings of the computer system are not associated with the purveyor of humor, and vice versa—poor attempts at humor are not seen as symptoms of a useless overall system. Conversely, if there are independent reasons for having a lifelike artificial agent as part of the interaction, then imbuing it with humorous skills will help to make it more lifelike and likeable (Nijholt 2002). Humorous remarks that are of poor quality, or are contextually inappropriate, will be counterproductive, being likely to irritate rather than engage the user. This is particularly pertinent to the argument for implementing humorous interfaces at the moment, because the current state of computational humor makes it extremely hard to generate automatically humor that is of good quality and moderate relevance. This means that the arguments in favor of having humor-capable interfaces are in some contrast to the abilities of the field to provide humor generation of a sufficient quality to make implementation a reality.

It is often argued that the use of humor within education can have beneficial effects, although the empirical evidence is mixed (Martin 2007, chapter 11). The idea that computational humor might be used in interactive teaching software was mentioned in passing by Binsted (1996), and Stock (1996) alludes to the scope for edutainment. This motivation is also invoked by McKay (2002) and

by Binsted, Bergen, and McKay (2003), although neither provides a design for a real tutoring system. Since existing computational humor is mostly concerned with language, it is relatively plausible to argue that a program that manipulates textual items for humorous effect may well help a student to learn about language. For example, a simple pun is usually based on similarity of sound, often uses ambiguity, and may rely on the fact that two different words mean the same. Thus a schoolchild who encounters such a pun may absorb some understanding of the workings of the language, while being motivated by the "fun" aspect of the experience. It is also possible to automate the explanation of why a particular text qualifies as a pun; for example, McKay (2002) shows how his pun generator could outline the links behind the jokes. Such explicit pedagogy, sweetened by the amusing material, might be quite effective.

One educationally motivated joke-generation system is STANDUP (Manurung et al. 2008), in which the ideas of Binsted's JAPE were reimplemented as the core of an interactive, child-friendly, robust, and efficient riddle generator.

The aim was to create a "language playground" for children with *complex communication needs* (that is, impairments of motor or cognitive ability that impede normal language use). The system was evaluated by a number of children with cerebral palsy who had limited literacy and conversational skills. Overall, the children were greatly engaged by being able to interact with the software and build simple puns and appeared to become more lively and communicative (Black et al. 2007; Waller et al. 2009). However, that study was small and so it is hard to draw firm conclusions. The STANDUP system is capable of producing literally millions of different jokes, but the overall quality of its output has not been tested.

As with user interfaces, the arguments in favor of joking software tutors are plausible, but the state of the art does not make it easy to build working systems, let alone perform rigorous trials.

How Creative Are These Systems?

A central question remains: to what extent are computational humor programs *creative*? By adopting a rather trivial notion of creativity, which demands only the production of novel items (compare Chomsky's arguments that routine human language use is inherently creative), then the verdict is simple. But creativity means more than simply producing hitherto unseen material. It is not possible to do justice here to the long-standing debate about when a computer program can be seen as creative (Boden 1998, 2003; Ritchie 2007; Colton 2008), but a few humor-specific remarks are in order.

For humans, simply appreciating humor—of which everyone is capable—is not usually regarded as creative. Hence, even where a computer program manages a first step toward appreciation (namely, recognizing that an item is humorous), it is hard to argue that the software is being creative, no matter how effectively it performs this step.

Humans who produce novel humor (particularly humor of a high standard) are usually deemed to be creative, so perhaps humor-generating programs should have similar status. For programs, the results so far show some promise in the area of *novelty*, an important component of creativity. Boden (2003) makes the useful distinction between artifacts that are novel for the creator (for example, where an inexperienced mathematician proves a result, not realizing that it is already known in the field), as opposed to results that are novel for the whole culture (such as the development of cubist art). The former type of achievement can lead to *P-creativity* (“personal”), whereas the latter are candidates for *H-creativity* (“historical”). Where do joke-generating programs stand in this respect? Some of the programs definitely produce jokes that have not been seen before, so they have at least the minimum qualification for an accolade of H-creativity. Unfortunately, these programs score very poorly on the scale of *quality*, another factor that is regarded as highly important for an overall verdict of being creative. The created texts generally consist of relatively feeble humor.

Whereas “novelty” could simply mean that the specific created entity has not been encountered before, there is a more demanding criterion, which concerns the extent to which the created entity is radically different from previous artefacts of this kind; this could be labelled “originality.” Boden has proposed that there are two qualitatively different forms of creativity, determined by the extent to which the act of creation overturns previous norms. The first, *exploratory* creativity, involves the navigation of a structured space of possibilities, forming new artifacts according to existing guidelines (perhaps abstract, unwritten, unacknowledged). More striking results will come from *transformational* creativity, in which the existing structured space is significantly altered. From this perspective, the joke machines can at best aspire to exploratory creativity: they rely on a small set of hand-crafted rules, and the joke texts are the results of applying these rules in different combinations or with different data values. However, much human-created humor can also be viewed in this way—it is rare for humor to cross established boundaries or restructure the space of humorous possibilities. Exploratory creativity, in this sense, is not trivial, and can even reach high levels of quality. This means that there are still demanding standards for computer generation of humor to aim

for, even if the programs are exploring an already structured space. In the current state of the art, there is plenty of room for improvement within this type of humorous creativity, without yet considering what it would mean to be radically original (“transformational”). Humor generators should show that they are proficient at walking before they attempt to run.

The Challenges Ahead

Twenty years ago, computational humor was unheard of, and—as noted at the start of this article—little attention was being paid to humor within AI. In such a setting, it was an interesting step to set down the simple structural properties of certain kinds of basic puns, and hence automatically construct texts that had the gross attributes of jokes. This showed that not all humor was beyond the reach of AI and that there was an “abstract syntax” to certain classes of joke. However, it is difficult to argue now that building another pun generator would, in itself, be a significant step forward, unless it introduced some nontrivial step forward in the direction of wider generality.

Even in the case of just verbal humor (wordplay, and so on), there are two ways in which computer-generated jokes such as puns could be more interesting.

First, there is the issue of contextual relevance. Although attempts at joke generation have implicitly been tackling the issue of “quality”—since the aim is to create jokes that are as good as possible—the joke generators are usually designed to create single texts in isolation. That is, these are “self-contained” puns, in which the central words or phrases are directly related to other words *in the same text*. In none of these cases is an utterance created that somehow links to a realistically complex context of use. But such contextually related wisecracks are part of the essence of punning in everyday life (at least in those cultures where punning occurs freely). The existing pun generators are competing with “canned” puns, such as those found in some children’s comics, or—in Britain—Christmas crackers, but have not begun to imitate contextual wisecracking. Loehr (1996) presented a preliminary study in using keyword matching to select JAPE-generated jokes for insertion in a dialogue, but that was a very rough and basic trial that did not really grapple with the complexities of contextual relevance.

There might be some ways to widen the range of computational punning (see Ritchie [2005] for some suggestions), but at the moment the state of the art is still limited. This has consequences for the short-term prospects of using current software in applications where impromptu and natural joke making might provide an appearance of lifelike

behavior, as discussed earlier. Relevance of content is just one of two criteria that Nijholt (2007) lists as central to the contextually appropriate use of humor; there is also the larger question of whether it is acceptable to use humor at a particular stage of the conversation. This latter issue is extremely difficult, requiring high-level cultural and social knowledge, and goes far beyond the very low-level mechanical approach currently used in computational humor. It is still difficult for designers of interactive computer systems to ensure that human-computer interactions flow naturally when all that is required is nonhumorous discourse. If the interface is to allow spontaneous humor, which may violate the norms of informative communication, the task becomes even more complicated.

The second possible improvement would be in the quality of the humor. Although existing programs produce structurally well-formed puns, most of the examples are not at all funny, and some barely qualify as jokes (as opposed to unmotivated ad hoc wordplay). If some abstract model could be devised that resulted in a program consistently producing genuinely funny output, that would be a step forward not only for possible applications but also for humor theory. That is, research must go beyond mere structural well-formedness in order to produce funny output consistently. What might lead to such improvements in funniness? It has often been argued that *incongruity* is a vital factor in humor, possibly combined with some *resolution* of that incongruity (Attardo 1994, Ritchie 2004), but there is still no formal and precise definition of what constitutes incongruity (or resolution). A typical dictionary explanation of incongruity is “want of accordance with what is reasonable or fitting; unsuitableness, inappropriateness, absurdity; want of harmony of parts or elements; incoherence” (Little et al. 1969). The knowledge representation techniques employed in AI should be a fruitful framework for stating a more precise and formal definition of incongruity (and perhaps also “resolution”). If such a notion of incongruity resolution could be shown empirically to result in humor (by using it in a joke generator), that would be a major advance.

One widely proposed account of humorous incongruity has its origin in the AI of the 1970s, when *frames* (Minsky 1975) and *scripts* (Schank and Abelson 1977) were proposed as knowledge structures for stereotypical situations and events. The application of these ideas to humor comes in the *semantic script theory of humor*, or SSTH (Raskin 1985), which in its more recent versions (Raskin 1996, 2002) continues to connect to this strand of AI work through *ontological semantics* (Nirenburg and Raskin 2004). The central idea of the SSTH is that a text is a joke if it is associated with two

scripts (complex knowledge structures) that, although having a large portion in common (an overlap), also differ crucially in certain aspects (are *opposite*). This could come about, for example, by the text being ambiguous. The SSTH then provides a list of what counts as “opposite.” The pronoun misunderstander of Nijholt and Nijholt, discussed earlier, is heavily influenced by the SSTH, in its method for comparing linguistic structures for humorous potential.

Any solution to the incongruity question, whether from the SSTH or elsewhere, should help to widen the focus of computational humor to other forms of humor. As noted earlier, computer generation of humor has largely concentrated on “verbal” humor, because the crucial relationships in the textual structures are primarily lexical, and hence can be computed. However, referential humor constitutes a very large proportion of the humor that appears in our culture.

Even if there was a good definition of incongruity (and its resolution), referential humor is still vastly more challenging than simple punning. Stock (1996) refers to fully developed humor processing as being “AI-complete”—that is, a full computational account of humor will not be possible without a solution to the central problems of AI. This view is echoed by Nijholt (2007). The problems become clear with even a superficial inspection of quite simple jokes, which typically involve cultural and real-world knowledge, sometimes in considerable quantities. Most referential humor consists of short stories (ending in a punchline), but AI has yet to produce a high-quality, fully automatic story generator, let alone one that can perform subtle tricks of narrative. It is common to observe that many jokes rely on ambiguity, but current text generators have nothing like the sophistication that would be needed to create exactly the right ambiguities for humor. Subtle manipulation of the audience’s expectations, indirect allusions, and carefully timed revelations, all of which are central to many jokes, require inference mechanisms of some sophistication. For example, the following, consisting of just one short sentence, requires considerable hypothetical reasoning.

A skeleton walked into a bar and said: “I’ll have a Budweiser and a mop, please.” (Tibballs 2000, No. 297, 49)

An even more vast and challenging aspect of humor is still to be tackled by AI. Science fiction might tell us that one crucial distinction between computers and humans is that machines (or software) can never have a sense of humor. It would be hard to collect firm evidence for or against such a view, given the lack of a single accepted definition of “sense of humor” (Ruch 1996, 2007), but computational humor is still far from addressing these

deeper issues. Research still focuses narrowly on the structural properties of the humorous items (jokes, and so on) without considering any of the human cognitive or emotional processes. There is therefore an exciting but unexplored area: the modeling of the effects of humor on the thoughts and emotions of the recipient.

Acknowledgements

The writing of this article was partly supported by grants EP/E011764/01 and EP/G020280/1 from the Engineering and Physical Sciences Research Council in the UK.

Note

1. www.dilbert.com.

References

- Attardo, S. 1994. *Linguistic Theories of Humour*. Number 1 in *Humor Research*. Berlin: Mouton de Gruyter.
- Bergen, B., and Binsted, K. 2003. The Cognitive Linguistics of Scalar Humor. In *Language, Culture and Mind*, ed. M. Achard and S. Kemmer. Stanford, California: CSLI Publications.
- Binsted, K. 1996. Machine Humour: An Implemented Model Of Puns. Ph.D. dissertation, University of Edinburgh, Edinburgh, Scotland.
- Binsted, K. 1995. Using Humour to Make Natural Language Interfaces More Friendly. Paper presented at the International Joint Conference on Artificial Intelligence Workshop on AI and Entertainment, Montreal, Quebec, 21 August.
- Binsted, K., and Ritchie, G. 1997. Computational Rules for Generating Punning Riddles. *Humor: International Journal of Humor Research* 10(1):25–76.
- Binsted, K., and Ritchie, G. 1994. An Implemented Model of Punning Riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. Menlo Park, CA: AAAI Press.
- Binsted, K.; Bergen, B.; and McKay, J. 2003. Pun and Non-pun Humour in Second-Language Learning. Paper presented at the CHI 2003 Workshop on Humor Modeling in the Interface. Fort Lauderdale, April 6.
- Binsted, K.; Pain, H.; and Ritchie, G. 1997. Children's Evaluation of Computer-Generated Punning Riddles. *Pragmatics and Cognition* 5(2): 305–354.
- Black, R.; Waller, A.; Ritchie, G.; Pain, H.; and Manurung, R. 2007. Evaluation of Joke-Creation Software with Children with Complex Communication Needs. *Communication Matters* 21(1): 23–27.
- Boden, M. A. 2003. *The Creative Mind*, 2nd ed. London: Routledge.
- Boden, M. A. 1998. Creativity and Artificial Intelligence. *Artificial Intelligence* 103(1–2): 347–356.
- Charniak, E., and McDermott, D. 1985. *Introduction to Artificial Intelligence*. Reading, MA: Addison-Wesley.
- Colton, S. 2008. Creativity Versus the Perception of Creativity in Computational Systems. In *Creative Intelligent Systems: Papers from the AAAI Spring Symposium*, ed. D. Ventura, M. L. Maher, and S. Colton. AAAI Press Technical Report SS-08-03. Menlo Park, CA: AAAI Press.
- Dopico, J. R. R.; Dorado, J.; and Pazos, A., eds. 2008. *Encyclopedia of Artificial Intelligence*. Hershey, PA: IGI Global.
- Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: The MIT Press.
- Friedman, J. 1972. Mathematical and Computational Models of Transformational Grammar. In *Machine Intelligence 7*, ed. B. Meltzer and D. Michie, 293–306. Edinburgh: Edinburgh University Press.
- Lappin, S., and Leass, H. J. 1994. An Algorithm for Pronominal Anaphora Resolution. *Computational Linguistics* 20(4): 535–561.
- Leech, G.; Rayson, P.; and Wilson, A. 2001. *Word Frequencies in Written and Spoken English: Based on the British National Corpus*. London: Longman.
- Lessard, G., and Levison, M. 1993. Computational Modelling of Riddle Strategies. Paper presented at the Association for Literary and Linguistic Computing (ALLC) and the Association for Computers and the Humanities (ACH) Joint Annual Conference, Georgetown University, Washington, DC, 15–19 June.
- Lessard, G., and Levison, M. 1992. Computational Modelling of Linguistic Humour: Tom Swifities. In *Research in Humanities Computing 4: Selected Papers from the 1992 Association for Literary and Linguistic Computing (ALLC) and the Association for Computers and the Humanities (ACH) Joint Annual Conference*, 175–178. Oxford, UK: Oxford University Press.
- Levison, M., and Lessard, G. 1992. A System for Natural Language Generation. *Computers and the Humanities* 26(1): 43–58.
- Little, W.; Fowler, H. W.; Coulson, J.; and Onions, C. T., eds. 1969. *The Shorter Oxford English Dictionary on Historical Principles*, revised 3rd ed. London: Oxford University Press.
- Liu, H., and Singh, P. 2004. Commonsense Reasoning in and over Natural Language. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2004)*, volume 3213 of *Lecture Notes in Computer Science*, ed. M. Negoita, R. Howlett, and L. Jain, 293–306. Heidelberg: Springer.
- Loehr, D. 1996. An Integration of a Pun Generator with a Natural Language Robot. In *Proceedings of the International Workshop on Computational Humor*, 161–172. Enschede, Netherlands: University of Twente.
- Manurung, R.; Ritchie, G.; Pain, H.; Waller, A.; O'Mara, D.; and Black, R. 2008. The Construction of a Pun Generator for Language Skills Development. *Applied Artificial Intelligence* 22(9): 841–869.
- Martin, R. A. 2007. *The Psychology of Humor: An Integrative Approach*. London: Elsevier Academic Press.
- McKay, J. 2002. Generation of Idiom-Based Witticisms to Aid Second Language Learning. In *The April Fools' Day Workshop on Computational Humor*, 77–87. Enschede, Netherlands: University of Twente.
- Mihalcea, R., and Pulman, S. 2007. Characterizing Humour: An Exploration of Features in Humorous Texts. In *Proceedings of the Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*, *Lecture Notes in Computer Science* 4394, ed. A. F. Gelbukh, 337–347. Berlin: Springer.
- Mihalcea, R., and Strapparava, C. 2006. Learning to Laugh (Automatically): Computational Models for

- Humor Recognition. *Computational Intelligence* 22(2): 126–142.
- Miller, G. A.; Beckwith, R.; Fellbaum, C.; Gross, D.; and Miller, K. 1990. Five papers on WordNet. *International Journal of Lexicography* 3(4). Revised March 1993.
- Minsky, M. 1975. A Framework for Representing Knowledge. In *the Psychology of Computer Vision*, ed. P. H. Winston, 211–277. New York: McGraw-Hill.
- Morkes, J.; Kernal, H. K.; and Nass, C. 1999. Effects of Humor in Task-Oriented Human-Computer Interaction and Computer-Mediated Communication: A Direct Test of SRCT Theory. *Human-Computer Interaction* 14(4): 395–435.
- Morreall, J., ed. 1987. *The Philosophy of Laughter and Humor*. Albany, NY: SUNY Press.
- Nack, F. 1996. AUTEUR: The Application of Video Semantics and Theme Representation for Automated Film Editing. Ph.D. dissertation, Lancaster University, Bailrigg, Lancaster, UK.
- Nack, F., and Parkes, A. 1995. Auteur: The Creation of Humorous Scenes Using Automated Video Editing. Paper presented at the International Joint Conference on Artificial Intelligence Workshop on AI and Entertainment, Montreal, Quebec, 21 August.
- Nijholt, A. 2007. Conversational Agents and the Construction of Humorous Acts. In *Conversational Informatics: An Engineering Approach*, ed. T. Nishida, chapter 2, 21–47. London: John Wiley and Sons.
- Nijholt, A. 2002. Embodied Agents: A New Impetus to Humor Research. In *The April Fools' Day Workshop on Computational Humor*, 101–111. Enschede, Netherlands: University of Twente.
- Nirenburg, S., and Raskin, V. 2004. *Ontological Semantics*. Cambridge, MA: The MIT Press.
- Raskin, V. 2002. Quo Vadis Computational Humor? In *The April Fools' Day Workshop on Computational Humor*, 31–46. Enschede, Netherlands: University of Twente.
- Raskin, V. 1996. Computer Implementation of the General Theory of Verbal Humor. In *Proceedings of the International Workshop on Computational Humor*, 9–20. Enschede, Netherlands: University of Twente.
- Raskin, V. 1985. *Semantic Mechanisms of Humor*. Dordrecht: Reidel.
- Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems*. Cambridge, UK: Cambridge University Press.
- Ritchie, G. 2007. Some Empirical Criteria for Attributing Creativity to a Computer Program. *Minds and Machines* 17(1): 67–99.
- Ritchie, G. 2004. *The Linguistic Analysis of Jokes*. London: Routledge.
- Ritchie, G. 2005. Computational Mechanisms for Pun Generation. Paper presented at the Tenth European Workshop on Natural Language Generation, Aberdeen, Scotland. 8–10 August.
- Ruch, W., ed. 2007. *The Sense of Humor: Explorations of a Personality Characteristic*. Mouton Select. Berlin: Mouton de Gruyter.
- Ruch, W., ed. 1996. *Humor: International Journal of Humor Research* 9(3/4).
- Schank, R., and Abelson, R. 1977. *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum.
- Shapiro, S. C., ed. 1992. *Encyclopedia of Artificial Intelligence*, 2nd ed. New York: John Wiley.
- Singh, P. 2002. The Public Acquisition of Commonsense Knowledge. In *Acquiring (and Using) Linguistic (and World) Knowledge for Information Access: Papers from the AAAI Spring Symposium*. AAAI Technical Report SS-02-09. Menlo Park, CA: AAAI Press.
- Stark, J.; Binsted, K.; and Bergen, B. 2005. Disjunctive Selection for One-Line Jokes. In *Proceedings of First International Conference on Intelligent Technologies for Interactive Entertainment*, volume 3814 of Lecture Notes in Computer Science, ed. M. T. Maybury, O. Stock, and W. Wahlster, 174–182. Berlin: Springer.
- Stock, O. 1996. “Password Swordfish”: Verbal Humor in the Interface. In *Proceedings of the International Workshop on Computational Humor*, 1–8. Enschede, Netherlands: University of Twente.
- Stock, O., and Strapparava, C. 2005. The Act of Creating Humorous Acronyms. *Applied Artificial Intelligence* 19(2): 137–151.
- Stock, O., and Strapparava, C. 2003. HAHAAcronym: Humorous Agents for Humorous Acronyms. *Humor: International Journal of Humor Research* 16(3): 297–314.
- Strapparava, C.; Valitutti, A.; and Stock, O. 2007. Automating Two Creative Functions for Advertising. Paper presented at the 4th International Joint Workshop on Computational Creativity, Goldsmiths, University of London, London, UK. 17–19 June.
- Tibballs, G., ed. 2000. *The Mammoth Book of Jokes*. London: Constable and Robinson.
- Tinholt, H. W., and Nijholt, A. 2007. Computational Humour: Utilizing Cross-Reference Ambiguity for Conversational Jokes. In *7th International Workshop on Fuzzy Logic and Applications (WILF 2007), Camogli (Genova), Italy*, volume 4578 of Lecture Notes in Artificial Intelligence, ed. F. Masulli, S. Mitra, and G. Pasi, 477–483. Berlin: Springer Verlag.
- Venour, C. 1999. The Computational Generation of a Class of Puns. Master's thesis, Queen's University, Kingston, Ontario.
- Waller, A.; Black, R.; O'Mara, D.; Pain, H.; Ritchie, G.; and Manurung, R. 2009. Evaluating the STANDUP Pun Generating Software with Children with Cerebral Palsy. *ACM Transactions on Accessible Computing (TACCESS)* 1(3). Article 16. New York: Association for Computing Machinery.
- Wilson, R. C., and Kidd, F. C., eds. 1999. *The MIT Encyclopedia of the Cognitive Sciences*. Cambridge, MA: The MIT Press.

Graeme Ritchie has degrees in pure mathematics, theoretical linguistics, and computer science and has published in various areas of AI and computational linguistics over the past four decades. He has worked on models of verbally expressed humor since 1993 and at present is a senior research fellow in the Natural Language Generation group at the University of Aberdeen.