Machine humour:
An implemented model of puns

Kim Binsted

Ph.D.
University of Edinburgh
1996

"Judging from their laughter, the children at school found my remarks humorous. So without understanding humor, I have somehow mastered it."

– Lal, in Star Trek, "The Offspring"

Abstract

This thesis describes a formal model of a subtype of humour, and the implementation of that model in a program that generates jokes of that subtype.

Although there is a great deal of literature on humour in general, very little formal work has been done on puns, and none has been implemented. All current linguistic theories of humour are over-general and not falsifiable. Our model, which is specific, formal, implemented and evaluated, makes a significant contribution to the field.

Punning riddles are our chosen subtype of verbal humour, for several reasons. They are very common, they exhibit certain regular structures and mechanisms, and they have been studied previously by linguists. Our model is based on our extensive analysis of large numbers of punning riddles, taken from children's joke books.

The implementation of the model, JAPE (Joke Analysis and Production Engine), generates punning riddles, from a humour independent lexicon. Pun generation requires much less world knowledge than pun comprehension, making it feasible for implementation.

To support our claim that all of JAPE's output is punning riddles, we conducted an evaluatory experiment. We took JAPE texts, human-generated texts, nonsense non-jokes and sensible non-jokes, and asked joke experts to evaluate them. For joke experts, we used 8–11 year old children, since psychological research suggests that this age group enjoys, and can recognize, punning riddles better than other age groups. The results showed that JAPE's output texts are, in fact, recognizably jokes.

The evaluation showed that our model adequately describes a significant subtype of verbal humour. We believe that this model can now be expanded to cover puns in general, as well as other types of linguistic humour.

Acknowledgements

I would first of all like to thank my supervisors, Drs Graeme Ritchie and Helen Pain, for their careful guidance during the course of the research reported here, and for smiling patiently at all the bad jokes – mine and JAPE's.

Sev Davison has been the most wonderful assistant that anyone could ask for — thanks for helping with the experiments, reading through this tome, and being generally supportive. Thanks too to Fiona Pollard and Ben Hambidge for their devoted efforts to convince small children that experiments are fun, and to Paul Bailey for proofreading the thesis.

I would also like to thank all the volunteers who helped me test the prototype program, in particular: David Asher, Jim Broughton, Don Casadonte, William Chesters, Myles Chippendale, Mark Dalgarno, Richard Evans, Enrique Filloy-Garcia, Chris Gathercole, Richard Henson, Tudor Wyatt Johnston, Matthias Klaes, Stewart Long, Steve McCoy, Marc Nantel, Simon Perkins, Tim Pizey, Sheila Rock, Sarah Rose, and Chris Seah. Your comments were invaluable.

The final evaluation was made possible by a great deal of help from the staff of Craiglockhart Primary School, and the staff of the Edinburgh International Science Festival. Thanks also to all the children who took part.

Thanks to Garry Dobson for his voice.

The final two years of my PhD were funded by the Natural Science and Engineering Council of Canada. In my final year, I was helped by a Special Opportunity Grant. Special thanks to Jean-Pierre Lalande and Lyn Pharand, for being much too helpful to be real bureaucrats, and to Graeme Hirst, for helping to convince NSERC to fund me.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Kim Binsted
Edinburgh
November 6, 1996

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis describes a model of punning riddles, based on an analysis of puns generated by and for humans. This model was implemented in a program, JAPE (Joke Analysis and Production Engine), which generates simple puns. Its output was evaluated by children 'joke judges', and judged to be of a similar quality to human-generated jokes.

We have taken a scientific approach to the problem of understanding verbal humour. Our methodology is based on generative linguistics and exploratory programming, which require that models of linguistic phenomena be formal and falsifiable. Few, if any, current linguistic theories of humour are implementable; our model, which is formal, implemented and evaluated, makes a significant contribution to the field.

In this chapter, we present the problem, and motivate our approach to solving it. We also discuss some methodological issues, and suggest some reasons why the artificial intelligence (AI) research community should be interested in our results.

1.1 Understanding verbal humour

Verbal humour — humour which is transmitted through language — has traditionally been the domain of literary scholars and some theoretical linguists. Recently, however, computational linguists have made considerable progress in modelling linguistic forms related to humour, such as metaphor and analogy. By building formal, testable models, they have made concrete progress towards understanding both how these phenomena work and the role they play in language as a whole. We believe it is time to do the same for humour.

Verbal humour as a whole is too large a domain to tackle all at once. For this reason, this thesis looks at linguistic humour, which is humour based on the language itself (this distinction is discussed in section 3.2). More specifically, we are interested in *punning riddles*, of the kind commonly found in children's joke books. For example:

> What do you get when you cross a sheep and a kangaroo? *A woolly jumper.* [Webb, 1978]

Most competent language users would recognise the above as a pun; moreover, they would also agree, for the most part, on whether any given text was a pun or not.

The question, then, is whether or not a model can be developed which captures the key features of simple puns. We do not expect this model to describe *all* linguistic humour, or even all puns; however, we do expect all texts described by this model to be puns, recognisable as such by human judges, and representative of the genre.

We believe that the development of a model of a sub-type of linguistic humour is a necessary first step to developing a more general computational model of verbal humour.

1.2 Methodological issues

There has been very little work done on the formal linguistics of puns, or indeed, of humour in general. What little has been done is described in chapter 2. Here we aim to justify our approach. In particular, we discuss why the computational approach is useful in the study of humour, why humour should be of interest to the AI research community, and why the particular task of generating punning riddles was chosen.

1.2.1 The computational approach

The work described in this thesis relies more on generative linguistics and AI exploratory programming than on literary or psychological studies. This approach has certain advantages. Both generative linguistics and exploratory programming are well understood, so the results of our research can be interpreted consistently within their methodological framework. Moreover, these approaches explicitly address issues of formality and falsifiability.

Falsifiable, formal and implementable models

Although many scholars have said many things about the linguistics of humour over the years (see [Attardo, 1994] for a good survey), very little of what has been said is *falsifiable*.

For a hypothesis to be scientific, it must be falsifiable. That is, there must be some possible experiment or discovery which would prove the hypothesis false. For example, a theory which predicts the result of an experiment is falsifiable since, if the experiment does not turn out as predicted, the theory has been proved false.

To be falsifiable, a model must make specific predictions. A *formal* model — that is, a model given in precise, unambiguous terms — is necessarily falsifiable, as a mismatch between the model and the phenomenon it is intended to describe would prove the model wrong.

Unfortunately, it is sometimes difficult to tease out all the various ramifications of a formal model, especially as they can be quite complex. It is therefore useful if the model is *implementable* — that is, if it is feasible to put the model into program form on a computer. Once implemented, the program can be run, and its performance compared with the phenomenon the model was intended to describe. The running of an implemented program can be seen as an experiment which may or may not falsify some theory.

The model described in this thesis is a falsifiable, formal and implemented model of a subtype of humour. Our methodology follows those of generative linguistics and exploratory programming, described below.

Generative linguistics

Generative linguistics[1] is a methodological framework for the study of language [Chomsky, 1957, Chomsky, 1965]. Most of the rules and guidelines of generative linguistics are implicit in the methodology of modern computational linguistics.

According to generative linguistics, the goal of language research is to define precise and detailed symbolic rules and structures which characterise what constitutes a sentence of a language and what does not. Such descriptions are falsifiable, in that there is no doubt about what they predict, and their predictions can be compared directly with sentences known to exist in the language in question. In principle, the rules and structures could be implemented on a computer, and used to generate or parse sentences.

The symbolic descriptions should capture regularities in the language. If two sentences in the language are similar syntactically, semantically or otherwise, the descriptions of those sentences should also be similar, so that the key features of the language are explicitly represented.

We have to a large extent adopted these attitudes in our study of riddles. We have attempted to devise abstract symbolic accounts of the detailed mechanisms underlying our chosen set of phenomena (certain types of punning riddle), we have defined these rules precisely (as shown by the computer implementation), and we believe that they show regularities in exactly the way that linguists expect grammars to display generalisations about sentences.

Exploratory programming

Within AI, the research paradigm sometimes known as *exploratory programming* is common. In this, a computer program is used to explore and test ideas. The exploratory programming approach is as follows:

1. Explore **ideas**.

[1] This section, and the following, are paraphrased from [Binsted and Ritchie, 1997].

2. Develop a detailed abstract **model**.

3. Devise a computational **task** central to testing the model.

4. Implement, debug and test a **program** to carry out the task.

5. Analyse the behaviour of the program and draw theoretical **conclusions**.

6. Use the conclusions to refine the model, and repeat.

This methodology owes a lot to the notion of an "experiment" in traditional science. However, exploratory programming is more often used as a way for researchers to refine and modify their ideas, than as a final test of a model (although that can also be the case). Having to construct a computer program which embodies your ideas forces a degree of detail and precision, and observing the behaviour of a running program often leads to insights into the phenomenon being modelled. (See [Buchanan, 1988, Newell and Simon, 1976, Ritchie, 1994] for further discussion of this approach).

The work described here can be seen as exemplifying this approach. The important product of the work was *not* the design, implementation and testing of the program itself; rather, it was the set of ideas that we developed in the course of the work.

1.2.2 Why humour?

We have established that AI methodologies might help us come to understand how humour works. However, AI, as a field, has its own agenda. Most AI researchers would agree with Minsky's claim that a suitable goal for AI research is to get a computer to do "... a task which, if done by a human, requires intelligence to perform," [Minsky, 1963]. If so, then most of human experience is open for investigation. Why research humour generation? How would an understanding of humour contribute to an understanding of intelligence in general?

Humour generation is falsifiable

If AI is to be science, then the hypotheses developed by AI researchers must be falsifiable — that is, it must be possible to devise an experiment which could disprove any claims of success (see section 1.2.1). This requirement makes the artistic side of human nature hard to investigate because, although artistic creativity is within the domain claimed by AI, it is difficult (if not impossible) to disprove the claim "this is art," no matter who or what produced the work in question. Scientific research into poetry, painting, and music suffers from this problem.

A second factor to take into account is whether or not an implementable model of the task can be constructed. AI research is often an exploratory attempt to develop a precise and detailed theory of the processes involved in performing some task; however, it helps if the task is already well defined and reasonably well understood. For a model of a phenomenon to be implementable, it is necessary (but not sufficient) that a formal description of the task has been, or can be, developed.

These two constraints, falsifiability and implementability, reduce the domain of AI research considerably, particularly when looking at creative intelligent behaviour. However, there is at least one area of human creativity that is both falsifiable and implementable, at least in part — humour generation.

Humour generation is falsifiable in that, unlike for most (if not all) other arts, there is a simple test of its success: whether the audience laughs or not. Although the reaction to potential humour varies from person to person, it is possible to state with confidence whether or not a particular person found a given 'joke', funny. The experimenter can therefore choose a reasonable statistical goal (e.g. a potential joke is successful if some percentage of its audience find it funny), then test it in a rigorous way.

Although no implementable model of humour as a whole has yet been developed, humour has been studied extensively in psychology, sociology, literature, anthropology, linguistics and other like fields (see chapter 2 for a review of the relevant literature). The study of humour has reached a state such that a precise model of at least a subset of humour can be developed, and implemented on computer.

Linguistic ambiguity

Competence in the use of natural language is a human trait that has been studied extensively by AI researchers. Most natural language research to date has seen ambiguity in language as an obstacle to comprehension. Most systems for comprehending natural language, for example, attempt to reduce the number of possible interpretations of the input to one, and a failure to do so is seen as a weakness in the system.

The potential for ambiguity, however, can be seen as a positive feature of a natural language. Metaphors, idioms, poetic language and humour all use the multiple senses of texts to suggest connections between otherwise dissociated concepts which cannot, or should not, be stated explicitly.

Fluent users of a natural language are able to both use and interpret the ambiguities inherent in that language. Linguistic humour (puns in particular) is one of the most regular uses of linguistic ambiguity. Any insights into how we use linguistic ambiguity to suggest humorous connections will further natural language research as a whole.

"Computers won't really be intelligent until..."

AI research is often motivated by lay benchmarks. For example, before the advent of capable chess–playing machines, it was assumed by many that general (human) intelligence was required to play chess well, and that computers would not *really* be intelligent until they could play chess too. Such benchmarks are changeable: now that computers regularly beat chess masters, few people believe that chess-playing ability is a good indicator of general intelligence. Instead, a sense of humour is often held up to be the mysterious key element that artificial intelligences will never have.

For example, Lieutenant Data, on the television show "Star Trek: The Next Generation", is an android, able to walk, see, speak and understand several languages, reason, and do many other tasks generally acknowledged to require intelligence. He cannot, however, tell or understand jokes — even simple punning riddles — although his attempts to do so are often used to comedic effect.

One of the goals of this research is to show that humour, like chess, is not so mysterious. Although there are many technical and theoretical obstacles to giving a machine a full human sense of humour, we show here that some simple subtypes of humour can be analysed, modelled, and then generated by a program.

Practical applications

Linguistic fluency requires the ability to use and understand non-literal language, such as metaphors, humour, exaggeration, etc. If we want to be able to talk easily to computers (and have them talk back), they must be able to use and understand humour.

Humour is used by humans in a work environment to entertain, release tension, increase bonding, disguise ignorance, veil criticism, and elicit co-operation [Barsoux, 1994]. It can be argued [Binsted, 1995] that humour could be used by a computer to similar ends. However, early research [Loehr, 1996] suggests that the use of humour by a machine must not be clumsy or inappropriate, lest it more irritate than amuse the human user. This issue is discussed in more detail in section 2.4.5.

1.2.3 Why generate punning riddles?

Humour itself is a broad subject. There are many different kinds of humour, expressed in a variety of forms and media. We have chosen to look at punning riddles, and to implement our model in a program which *generates* them.

Punning riddles are the type of humour chosen for investigation for several reasons. The linguistics of riddles has been investigated in at least one previous work [Pepicello and Green, 1984], although the model developed there is not entirely satisfactory (see section 2.2 for further discussion). Also, there is a large corpus of riddles to examine: books such as the Crack-a-Joke Book [Webb, 1978] record them by the thousand. Finally, riddles exhibit regular structures and mechanisms, which could be modelled and used to generate new riddles. The genre of punning riddles is described in more detail, and its key features discussed, in section 3.2.

The computational task (see section 1.2.1) chosen to test our model is the generation of punning riddles. In theory, the model could also have been implemented in a program which *recognises* punning riddles. A joke-understanding program, however, is not feasible for two reasons: implementability and falsifiability.

Joke comprehension requires a wide range of world knowledge, which is not generally available in computational form; if any key information is not in the system's knowledge base, the joke will not be recognised or understood. A joke generating system, however, can generate jokes from whatever information it *does* have, however limited.

Moreover, joke comprehension is not readily testable. Senses of humour differ. If a model of punning riddles were implemented in a system which was purported to understand jokes, and it failed to recognise a common pun, what conclusions could be drawn? It could be argued that the joke did not appeal to the system's sense of humour. Joke generation, however, is testable, in that its output can be given to human 'joke judges' — if a significant proportion of them agree that its output is humorous, the system can be deemed a success. Please see chapter 5 for a discussion of our evaluation methodology.

1.3 Research questions

The research described in this thesis is the modelling and generation of punning riddles, using a computational linguistics approach. The questions this research intends to answer are:

- Is there a subtype of humour which exhibits structures and mechanisms regular enough to be captured in a formal model? What features would such a model have?
- Would such a model be implementable? Can the kind of knowledge used to generate jokes be put into computational form?
- Would the texts generated by such a program be jokes? If so, would they be of a similar quality to those generated by humans?
- Would the behaviour of a joke-generating program say anything about how humans generate, recognise and use verbal humour?

1.4 Research goals

Motivated by the above, the goals of this research are:

- To develop a falsifiable, formal and implementable model of punning riddles, in such a way that their key features and mechanisms are captured;
- To implement that model in a program which generates punning riddles, and *only* punning riddles;
- To evaluate the performance of that program, by comparing the reaction of human joke judges to both the program's output and human-generated jokes; and,
- To draw conclusions, based on the performance of the program, about the nature of this subtype of humour.

If this research achieves these goals, then it will have made a significant step towards the understanding of linguistic humour and humour in general.

1.5 Overview

This thesis describes our efforts to answer the research questions and reach the research goals described above.

Following the methodology of exploratory programming (see section 1.2), we examined a large number of punning riddles, analysing regularities in their structures and mechanisms. Based on this analysis, we developed a formal model of punning riddles.

Our model is not based on any particular theory of humour, as all theories of humour to date are over-general and unimplementable. There have, however, been some useful linguistic studies of punning riddles, and their observations were incorporated into the model.

The model was first implemented in JAPE-1, which used a small hand–built lexicon to generate simple punning riddles. The results of JAPE-1's informal evaluation was used to inform improvements to the model and to the program. JAPE-2 has a much wider scope than JAPE-1, and uses a large humour–independent lexicon to generate puns.

To experimentally evaluate JAPE-2's behaviour, we took JAPE texts, human-generated texts, nonsense non-jokes and sensible non-jokes, and asked joke experts to rate them.

For joke experts, we used 8–11 year old children, since psychological research suggests that this age group enjoys, and can recognize, punning riddles better than other age groups. The results showed that JAPE-2 output texts are recognisably jokes, and suggested several ways in which the model and the lexicon could be improved.

The evaluation showed that our model adequately describes a significant subtype of verbal humour. We believe that this model can now be expanded to cover puns in general, as well as other types of linguistic humour.

1.6 Chapter outline

The following chapters describe a formal model of simple punning riddles, its implementation in a system which generates such riddles, and the evaluation of that implementation.

Chapter 2 reviews the literature relevant to this research. Although there is some work available on humour in general, current linguistic theories of humour tend to be overgeneral. Few, if any, previous models of puns are formal enough to be implemented.

Chapter 3 gives our model of punning riddles, which is based on an analysis of the structures and mechanisms found in human-generated punning riddles.

Chapter 4 describes the implementation of the model given in chapter 3, and the linguistic resources required for the system, JAPE (Joke Analysis and Production Engine), to be able to generate punning riddles.

Chapter 5 describes both the exploratory and confirmatory evaluations of JAPE, which show that JAPE does indeed generate jokes, although they are neither as joke-like nor as funny as human-generated jokes.

Chapter 6 discusses issues raised during the implementation of the model and the evaluation of JAPE. Possible fixes and further work are discussed.

Chapter 7 assesses the overall success of the research.

Chapter 2

Literature review

2.1 Introduction

Although a great deal of work has been done on humour in psychology, literature and sociology, less has been done in linguistics, and little in AI or computational linguistics. Models of humour abound; computationally tractable models do not.

The field of humour studies is wide, and has been well summarised elsewhere (e.g. in [Chapman and Foot, 1976] and [Attardo, 1994]). This chapter provides an overview of the works that have influenced this research — in particular, Pepicello and Green's analysis of the language of riddles [Pepicello and Green, 1984], and Attardo and Raskin's General Theory of Verbal Humour [Attardo and Raskin, 1991]. We also discuss other relevant papers in linguistics, especially those that adopt a computational approach.

We then *briefly* discuss some more theoretical or philosophical approaches to humour research. Work of this type is valuable, and could inform a general theory of humour; however, it is much too general to give much concrete guidance to our work. For similar reasons, mathematical [Paulos, 1980] and neural net [Katz, 1994] analyses of humour perception are also only briefly discussed.

2.2 Riddle theory

There are numerous collections and analyses of riddles, from the viewpoints of anthropology, sociology, literature and related fields. Aside from providing an overview of humour (and possibly some goal jokes to replicate), however, these works are not particularly relevant to this research. There has been much less *linguistic* research on riddles as such (see [Attardo, 1994] for a survey). In fact, the only linguistic work that explores the genre of punning riddles in detail sufficient for our purposes is [Pepicello and Green, 1984].

2.2.1 Ambiguity and wit

In their book, Pepicello and Green describe the various grammatical, written and visual strategies incorporated in riddles[1]. They hold the common view that humour is closely related to ambiguity. This ambiguity could be in the language of the riddle itself (such as the phonological ambiguity in a punning riddle), or in the situation the riddle describes. Moreover, they claim that humour depends on that ambiguity being 'unsolvable' by the listener, at least until the punchline resolves it in some unexpected way.

Pepicello and Green divide linguistic ambiguity into three kinds: phonological, morphological, and syntactic ambiguity. For example[2], the sentence "John lives near the bank" is phonologically ambiguous, since the noun "bank" can refer to either a building where money is stored, or the shore of a river[3]. The sentences "The book is read," and "The book is red", however, are morphologically ambiguous, since "read" is only phonetically identical with "red" in its past participle form. Finally, the sentence "John looked over the car" is syntactically ambiguous, since it has two distinct parse trees.

Each kind of ambiguity, or a combination, can be used in riddles. For example:

1. **Phonological:** What bird is lowest in spirits? *A bluebird.*
2. **Morphological:** Why is coffee like soil? *It is ground.*
3. **Syntactic:** Would you rather have an elephant charge you or a gorilla? *I'd rather have the elephant charge the gorilla.*

[1] This section is essentially a precis of chapters two and three of [Pepicello and Green, 1984].

[2] These examples are taken from [Pepicello and Green, 1984].

[3] Most linguists would call this *word sense* or *lexical* ambiguity, rather than phonological ambiguity.

As can be seen from these examples, the ambiguity can occur either in the question (3) or the punchline (1 and 2) part of the riddle.

Pepicello and Green go on to describe many different strategies used in riddles to produce and manipulate these linguistic ambiguities. However, what all these strategies have in common is that they ask the 'riddlee' to accept a similarity on a phonological, morphological, or syntactic level as a point of *semantic* comparison, and thus get fooled. For example, the riddle:

> Why is a river lazy? *Because it seldom gets out of its bed.* [Webb, 1978]

uses the phonological ambiguity in the word "bed" to imply that a river bed is semantically identical with a sleeping bed, and therefore that not getting out of a river bed is a sign of laziness.

So, Pepicello and Green's main point is that riddles use ambiguity to confuse the riddlee, and that a common technique is to use phonological, morphological and syntactic ambiguities to suggest false semantic connections.

2.2.2 Relevance of Pepicello and Green's work to this research

Pepicello and Green analyse a great number of punning riddles, and describe some basic linguistic features of the genre. They have identified several types of linguistic ambiguity common in punning riddles, and several mechanisms the riddles use to exploit that ambiguity to humorous effect. We also subdivide the genre of punning riddles according to type of ambiguity and mechanism, and although we make slightly different categorisations, the influence of Pepicello and Green is strong.

Please see section 3.2 for our short taxonomy of riddles, and a discussion of which of these are computationally tractable.

2.3 Linguistics of humour

Although several studies have been done on the language of humour (e.g. [Chiaro, 1992] and [Booth, 1974]; see [Attardo, 1994] for a good review of the literature), few have attempted to develop detailed linguistic models of humour. This is not to denigrate the work that has been done in the field; however, for a humour theory to be falsifiable, a formal linguistic model of (at least a subset of) humour is required at some point.

The prevailing linguistic theory of humour is Salvatore Attardo and Victor Raskin's General Theory of Verbal Humour (GTVH) as described and developed in [Attardo and Raskin, 1991], [Attardo and Raskin, 1994] and [Ruch et al., 1993]. More computational models of subtypes of humour are discussed in section 2.4.

2.3.1 The General Theory of Verbal Humour

The GTVH is an attempt by Attardo and Raskin [Attardo and Raskin, 1991] to build a linguistically sound model of verbal humour[4]. By analysing the similarities and differences of a set of variants on a light-bulb joke, Attardo and Raskin find six joke parameters, or *knowledge resources* (KR), which between them determine the final text form of the joke. These KRs are organised into a hierarchy, as shown in figure 2.1.

Script Opposition

The *script opposition* KR is based on Raskin's earlier script-based semantic theory of humour, which he summarises as follows:

> "A chunk of structured semantic information, the script can be understood for the purposes of this article as an interpretation of the text of a joke. The main claim of [the script-based semantic theory of humour] is that the text of a joke is always fully or in part compatible with two distinct scripts and that the two scripts are opposed to each other in a special way. In other words, the text of a joke is deliberately ambiguous, at least up to a point, if not to the very end. The punchline triggers the switch from the one script to the other by making the hearer backtrack and realize that a different interpretation was possible from the very beginning." [Attardo and Raskin, 1991]

[4] This section is essentially a precis of [Attardo and Raskin, 1991]. Their examples are used.

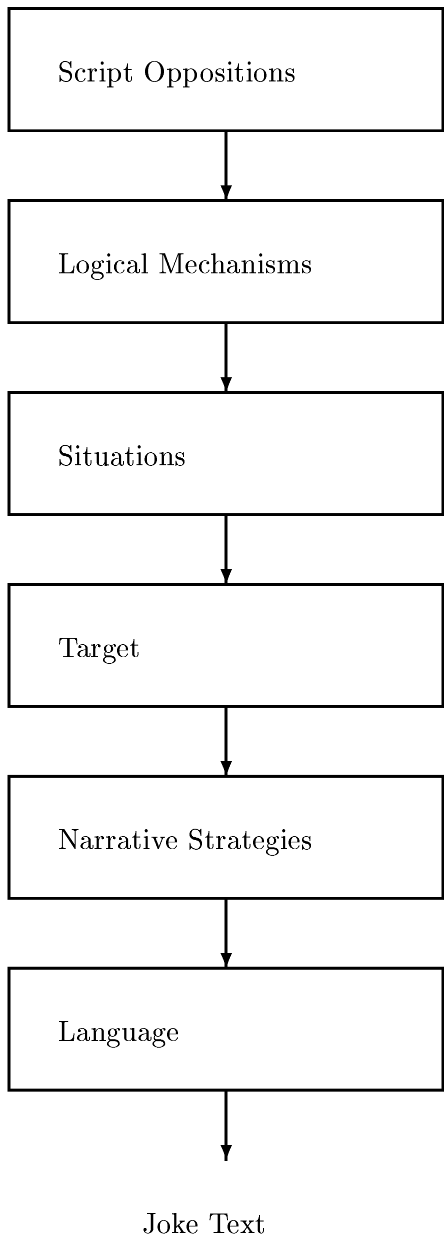


Figure 2.1: A complete hierarchy of the six Knowledge Resources of the General Theory of Verbal Humour.

The 'special ways' in which scripts can be opposed are at various levels of abstraction. Raskin's examples of types of opposition include: real vs unreal, good vs bad, high status vs low status, and nondumb vs dumb. For example, the joke:

> JOKE 1: How many Poles does it take to screw in a light bulb? Five. One to hold the bulb and four to turn the table he's standing on.[5] [Freedman and Hofman, 1980]

uses the nondumb vs dumb opposition, since it is about applying a stupid method to a simple task which most people deal with in a simple, intelligent fashion.

Although incongruity is a feature of humour that has long been noted, it is not clear what Attardo and Raskin mean by the 'opposition' of scripts. Puns, for example, usually favour one interpretation of the punning word, then suddenly force the listener to accept the other, but the two meanings are not necessarily in opposition. For example, in the punning riddle:

> What do you get when you cross a sheep and a kangaroo? *A woolly jumper.* [Webb, 1978]

It is not clear what the two opposed scripts might be. Two senses of the word "jumper" appear, certainly; however, they do not seem to be *opposed*, as such. It seems that "script opposition" is scarcely more specific than "incongruity" is, as a precondition for humour.

Logical Mechanism

This parameter determines the mechanism used to oppose the scripts. For example, joke 1 apparently uses figure-ground reversal. When screwing in a light bulb, the room, the ladder, and the person screwing in the bulb usually stay still, while the light bulb moves; joke 1 reverses that situation.

[5] My apologies for the Pole jokes. Raskin's main examples are targetted (usually at an ethnic minority) jokes of this type.

Holding the other parameters of joke 1 constant, but changing the logical mechanism to ‘false analogy’, Attardo and Raskin get:

> How many Poles does it take to screw in a light bulb? Five. One to hold the light bulb and four to look for the right screwdriver.

Other mechanisms cited by Attardo and Raskin include simple reversal, false priming, simple juxtaposition, and “the juxtaposition of two different situations determined by the ambiguity or homonymy in a pun” [Attardo and Raskin, 1991, p. 306]. Although intuitively appealing, these mechanisms are given only vague definitions; no criteria are given for determining which mechanism or mechanisms are used in any given joke. See section 3.2.5 for the mechanisms we propose.

Attardo and Raskin do note that, in the “joke telling mode of communication” [Attardo and Raskin, 1991, p. 306], the truth of statements and their consistency become less important. The pseudo-logic of the joke, therefore, need not be valid, just vaguely persuasive — persuasive enough that the listener will go along with the joke.

Situation

The situation of a joke is the set of details (e.g. time, place, objects, or activity) which specify the joke. A given script opposition and logical mechanism can be applied to a number of different situations. For example,

> How many Poles does it take to wash a car? Two. One to hold the sponge and one to move the car back and forth.

differs from joke 1 only in situation.

Target

The target of a joke, the person or stereotype the joke is aimed at, is the only optional parameter of the six. Many jokes have no identifiable target. The target of joke 1 is, clearly, Poles, but it could be changed to almost any other group which is stereotyped as ‘stupid’.

Narrative Strategy

This KR determines the form the joke will take, i.e. riddle, conundrum, expository text, etc. The more standard strategies, Attardo and Raskin suggest, have the advantage that the punchline automatically falls in the right place. Also, the choice of logical mechanism apparently limits the range of narrative strategies available.

Joke 1 as expository text, rather than conundrum, might look like this:

> It takes five Poles to screw in a light bulb: one to hold the light bulb and four to turn the table he's standing on.

Language

This parameter specifies which paraphrasing of the joke is used (i.e. what the surface form of the joke is). It is constrained by all the other parameters. For example, although the language parameter determines the exact phrasing and placement of the punchline, all the other parameters (particularly narrative strategy and logical mechanism) have a lot of input into it as well.

2.3.2 Relevance of GTVH to this research

By providing a parameterised model of verbal humour in general, Attardo and Raskin have provided a rough structure which could, in part, guide the design of a humour-generating program.

In particular, they note that holding some of their parameters constant produces a joke 'template'[6]. If one or two of the parameters are kept variable, and the rest held constant, we have a constrained model of (certain types of) humour which could, in theory at least, be used as the basis for a program design.

Unfortunately, their theory has several important flaws. It is neither detailed nor formal enough to be implemented as it stands, even in a constrained, 'template' form. Some

[6] We later (sections 3.7 and 4.5) use "template" to refer to our mechanism for putting descriptions into question-answer form.

of their 'knowledge resources', in particular the script-opposition and logical mechanism KRs, require a near-complete understanding of the world (including the rules of physics, the operations of human society, and common-sense reasoning) in order to operate. Even their language KR "includes all the choices at the phonetic, phonologic, morphophonemic, morphologic, lexic, syntactic, semantic, and pragmatic levels of language structure that the speaker is still free to make" as well as "a few specifically humorous elements and relations" [Attardo and Raskin, 1991, p. 298]. Moreover, the logical mechanism KR, which is discussed only briefly, seems to contain the essential humour–creating knowledge: how to bring together two incongruous concepts in a humorous way.

In order for computer implementation of this model to be feasible, it must be severely constrained, perhaps so much as to be unrecognisable. Similarities between the GTVH and the more restricted model developed here are noted in section 6.4.2.

2.4 Humour computation

There are not many researchers currently investigating humour computation. Ephratt [Ephratt, 1990] has modified Schubert's [Schubert, 1986] preference parsing algorithm to detect some limited types of humorous ambiguity. Weiner and De Palma [Weiner and de Palma, 1993] have developed a model of simple riddles, but have not implemented it. Takizawa [Takizawa, 1993] has implemented a simple system that can detect some puns in Japanese. Attardo and Raskin [Attardo and Raskin, 1994] are developing a computational model of humour based on the GTVH (see section 2.3.1). Finally, Dan Loehr [Loehr, 1996] has integrated our system, JAPE, into his natural language using agent. These are all discussed below.

Aside from Loehr, only Attardo and Raskin have implemented any kind of joke generating system.

2.4.1 The Light Bulb Joke Generator

Attardo and Raskin have put together a simple joke generating system, LIBJOG (LIght Bulb JOke Generator) [Attardo and Raskin, 1994], mainly to show how poorly simple cut–and–paste methods work. The first version combines an entry for a commonly–stereotyped group, for example:

```
(i)(Poles ((activity1 hold the light bulb)
          (numberX 1)
          (activity2 turn the table he is standing on)
          (numberY 4)))
```

with an outline of a light bulb joke:

> How many (group name) does it take to screw in a light bulb? (NumberX). One to (activity1) and (numberY) to (activity2). [Condition: $X = 1 + Y$.]

to make, not surprisingly:

> How many Poles does it take to screw in a light bulb? Five. One to hold the light bulb and four to turn the table he's standing on.

Clearly, this is cut-and-paste generation of the very simplest kind.

Although Attardo and Raskin claim that later versions of LIBJOG "introduced more templates, more fields, and looser (and richer) relations among them," [Attardo and Raskin, 1994, p. 26] they give no evidence of a significantly improved method. The joke generating mechanism seems to remain the same: substitute the (humour–related) values in an entry for a stereotyped group, directly into a light bulb joke template like the one above.

In this research, we have implemented a more interesting joke generating system, JAPE, that differs from LIBJOG in several significant ways:

- JAPE's lexicon is humour-independent; that is, it contains only common-knowledge (rather than joke–oriented) semantic and syntactic information, as one might find in a lexicon designed for other applications.
- JAPE produces a wider range of riddles, which do not have the fixed surface format of light-bulb jokes.

- JAPE is an implementation of a model of humour, albeit a very simple one, rather than a program that can produce jokes in an uninteresting way. Although JAPE uses a lexicon and templates, there is more to its method than simply pasting the two together.

2.4.2 Ephratt

Ephratt's work [Ephratt, 1990] was one of the first to look at punning riddles from a computational linguistics standpoint. She shows how linguistic riddles (i.e. riddles based on lexical, structural, or idiomatic reading vs literal reading ambiguities) could be parsed with a modified preference parser.

Schubert's trade-off preference parsing algorithm [Schubert, 1986], which Ephratt modifies, assigns equal weights to various linguistic criteria when choosing a parse of an ambiguous sentence. The preferred reading of a node in a (non-joke) parse tree is the reading with the lowest cost. In other words, there are linguistic heuristics which assign a cost to certain parsing choices, and the parser chooses the lowest-cost parse.

According to Ephratt, jokes can be parsed with only one modification to this algorithm. Rather than selecting the lowest-cost parse, the parser should:

> - "locate the one multiple parsing node with the largest numerical gap between its highest cost and its lowest cost
> - define this node as the punch node and its highest cost as its punch parsing
> - combine this punch parsing (of the node) with the lowest cost reading of the rest of the tree." [Ephratt, 1990, p. 47]

That is, in a joke parse, one node has both a low-cost and a high-cost reading. The low-cost reading would normally be preferred; however, *because* the text is a joke, the high-cost reading is chosen instead. The rest of the tree is parsed as for a normal text, with the low-cost readings preferred.

The worked example Ephratt gives is:

A gold-miner is a person that has strong hands and boxes.

Using Schubert's algorithm, only one ambiguous node is found (the "boxes" node), and the interpretation with the overall lowest score is:

A gold-miner is a person that has strong hands and strong boxes.

However, if the *highest* cost parse at the ambiguous node is chosen, the resulting interpretation is:

A gold-miner is a person that has strong hands and he boxes.

This, according to Ephratt, is the joke reading. Unfortunately, this 'joke' given above is a good example of differing senses of humour; we must admit that we just do not get it. That the text is ambiguous is clear; that it is a joke is not.

Although the elegance of this model of pun parsing appeals, all of the examples given are unconvincing. It is not obvious that the above reading is the 'correct' joke reading, although it is clear how the parser reaches this result. In some punning riddles, it seems that one parse is not selected over another, but that, instead, both parses are retained. For example, in:

What do you get when you cross a sheep and a kangaroo? *A woolly jumper.*

it seems that neither reading for the punchline ("woolly jumping thing" or "woolly sweater") is the 'correct' joke reading — instead, both readings are entertained simultaneously, producing the humorous effect. Ephratt's system, however, would prefer one reading to the other, thus perhaps missing the point.

2.4.3 Weiner and De Palma

Weiner and De Palma [Weiner and de Palma, 1993] model simple riddles as texts which have at least one lexical (word sense) ambiguity remaining after all syntactic and semantic processing has finished, and leave the listener preferring the 'incorrect' reading. They hold that it is often parallelism [Prince, 1981], "the tendency to expect syntactic, semantic and pragmatic consistency" [Weiner and de Palma, 1993, p. 186], which keeps the listener entertaining the incorrect reading even after another reading becomes apparent.

Weiner and De Palma's chosen subtype of riddle is very similar to our own. They analyse question-answer riddles, based on word-sense ambiguity only, in which the question contains attributes which could describe more than one noun phrase: the given answer to the question (i.e. the punchline) and the 'straight' answer. The riddle question is constructed such that the listener expects the 'straight' answer.

For example, in the riddle:

> What has a mouth but cannot eat? *A river.*

the ambiguous word is "mouth", which has two senses: the oral cavity, and the point where a river enters a larger body of water. According to category theory ([Barsalou, 1982] and [Lakoff, 1986]), the listener selects the most context-independent reading first (i.e. "mouth" as the oral cavity), and expects the answer to correspond to that interpretation (e.g. "nothing", or perhaps "a baby with its lips sewn together"[7]); however, the given answer instead corresponds to the context-dependent reading (i.e. the "mouth" of a river). Note that the punchline is a valid answer to the question asked — that is, the 'straight' answer is preferred, not because the alternative is nonsensical, but because the alternative interpretation is more context-dependent.

It is also interesting to note that the riddles of the type that Weiner and De Palma describe and use as examples are not very *funny*. They are more conundrums or brainteasers than jokes. This may be due to the fact that they only examine riddles which have the ambiguous word in the *question* (rather than the punchline). These are very rare in the humorous riddle corpora (e.g. [Webb, 1978]) we have examined.

In their 1993 paper [Weiner and de Palma, 1993], Weiner and De Palma anticipate building a system, based on their findings, which would generate riddles. To our knowledge, no such system has been implemented to date.

---

[7] My example.

2.4.4 Takizawa

Takizawa [Takizawa, 1993] has developed a simple pun-understanding system, DUJAL [Takizawa, 1991], which can recognise some simple puns in pre-processed, spoken Japanese. His definition of a pun is:

> "...a language expression in which a phoneme sequence is replaced with a similar phoneme sequence ...resulting in additional or hidden meanings besides the usual interpretation." [Takizawa, 1993, p. 171]

This description is similar to the *substitution mechanism* described in section 3.2.5. Although Takizawa recognises that a text segment can be replaced with a phonologically identical (as opposed to "similar") segment to form a pun, he does not consider these, as they cannot be detected by his system. He calls the original phoneme sequence the *base* word, and the similar phoneme sequence the *skewed* word.

Takizawa collected 325 Japanese puns from volunteers. He found that there were two main types: those in which both the base word and the skewed word appear, and those in which only the skewed word appear. The first type subsumes what we call *juxtaposition* and *comparison* puns, and the second type is closely related to our *substitution* puns (see section 3.2.5).

Based on his collected puns, Takizawa makes several observations, some of which also apply to English-language puns:

- When both the base word and the skewed word appear, both retain their normal pronunciation. However, when only the skewed word appears, its pronunciation dominates. For example, all the words in the spoonerism "mute kitten and cute mitten" are pronounced normally; however, in the pun "spooktacles" ("What does a near-sighted ghost wear?"), the initial syllable is pronounced as "spook" rather than "spec". This observation applies to the spelling as well as the pronunciation of the texts.

- Skewed words are usually single words, rather than compounds.

- The syntactic category of the skewed word is usually different from that of the base word. This is *not* true of English-language puns. Takizawa says that:

  > "Puns with the same word classes of both the base word and the skewed word are too commonplace to be regarded as puns. This is possibly because the Japanese language has many homonyms and similar sounding word pairs ..." [Takizawa, 1993, p. 176]

  Japanese also has a very regular pronunciation and spelling system, unlike English.

- The base word is usually part of a familiar idiomatic phrase. This is also true in English.

DUJAL is a system which detects puns in speech. A pun is considered to be detected when DUJAL correctly identifies the base word and the skewed word. Its input is preprocessed Japanese speech, in the form of a phoneme lattice, with confidence scores on the identified phonemes. DUJAL's output is the suspected base and skewed words, also with confidence scores. It works as follows:

1. The system tries to match a sequence of contextually appropriate words (SC) to its input. If this sequence matches the input perfectly, including intonation, there is no pun. Otherwise,

2. the system looks for a sequence of words (SP), not necessarily contextually appropriate, which match the phoneme input more closely. If it finds one,

3. the system looks for:

   - a semantic relationship between SP and SC, or
   - self-consistency within SP, or
   - conformity between SP and its knowledge base (which contains limited 'real world' knowledge)

   If it finds any of these, it has detected a pun, with SC as the base words, and SP as the skewed words.

Although DUJAL is a toy system, it is quite successful at identifying a small number of Japanese puns. Because the types of Japanese pun identified by Takizawa are very similar to those we have found in English, we suspect that DUJAL could be modified to recognise English language puns.

2.4.5 Loehr

Loehr [Loehr, 1996] has integrated a prototype of our system, JAPE, into his system, Elmo. Elmo is a natural language using agent, which roams a text-based virtual environment designed for English language students. Elmo is now able to use humour in one of four ways. He can generate a question-answer riddle when asked, for example:

> KIM: Elmo, tell me a joke.
> ELMO: OK, but it won't be a very good one, Kim.
> What do you get when you cross a sheep and a kangaroo?
> A woolly jumper!

Elmo can also search the lexicon for key words, based on the user's input, and attempt to generate a joke based on those key words, for example:

> KIM: Will the dog bite me?
> ELMO: Hmmm ...
> What do you call computer affection?
> A love byte!

In this case, it found that "byte" is a homonym of "bite", which was found in the user's input, and found that a riddle could be generated containing the word "byte". Despite the use of the key word, Elmo's response is barely relevant.

The third type of humour that Elmo produces is much more appropriate. An example is:

> KIM: No more help for me, Elmo?
> ELMO: I don't know what else to say, Kim.

> What do you call a useless assistant?
>
> A lemon aide.

In this case, the response is quite relevant, as the joke is about a useless assistant, which is exactly what Elmo is in the example situation. Unfortunately, this type of humorous response was hard-wired into Elmo, with appropriate JAPE riddles chosen by hand to suit particular situations. The final type of humour Elmo produces was also hard-wired:

> KIM: That's all the help you have?
>
> ELMO: Sorry I'm not more helpful. I guess all I can offer is 'lemon aid'.

Unsurprisingly, an informal survey found that users preferred the last kind of humour, in which the pun is smoothly integrated into the dialogue. They least preferred the second kind, in which an unsolicited, barely relevant riddle is produced in the middle of the conversation.

Although it would have been preferable to generate all four kinds of conversational humour automatically, the integration of JAPE with Elmo does show that humour can be integrated into textual dialogues by a computer. It also emphasizes the importance of smooth integration — unsolicited, irrelevant humour is usually more annoying than entertaining.

Loehr is now attempting to integrate the final version of JAPE, which is larger and more sophisticated, into Elmo.

2.5 The pragmatics of humour

Several researchers have proposed models of humour based on theories of pragmatics. Although these models are not formal enough to be implemented, they do make concrete suggestions about how humans process (linguistic) humour. In this section, we look at two of the more formal researchers in this area, Curcó [Curcó, 1996] and Giora [Giora, 1991].

2.5.1 Curcó

Curcó [Curcó, 1996] attempts to use relevance theory [Sperber and Wilson, 1986] to explain how hearers arrive at humorous interpretations of text. She sees irony, as treated under relevance theory [Wilson and Sperber, 1992], as a particular case of verbal wit, and extends Wilson and Sperber's treatment accordingly.

Relevance theory is a theory of pragmatics, which describes the processing of an utterance in terms of the relevance of that utterance to the hearer. It attempts to account for pragmatic effects in communication with a single principle. According to relevance theory, the hearer attempts to find an interpretation of the utterance that is of optimal relevance, in that it gives maximum cognitive effects for minimum cognitive effort. A cognitive effect is produced when an existing assumption is strengthened, contradicted, eliminated, or combined with new information to produce a new assumption.

Curcó identifies three mechanisms that produce humorous effects:

- **"The entertainment of contradictory propositional content."** [Curcó, 1996, p. 2] That is, the utterance either contains, implies, or leads the hearer to assume two contradictory ideas.
- **"The treatment of foreground assumptions as if they were in the background."** [Curcó, 1996, p. 2] That is, there is an assumption which is important and relevant, but is treated as if it were irrelevant and not worth mentioning.
- **"A clash between the expectations of the way in which upcoming material will achieve relevance and the way in which it actually does."** [Curcó, 1996, p. 3] As an utterance is uttered, the hearer searches for relevance in what is being said, often anticipating the kind of information the remainder of the utterance will have to provide to attain relevance. A humorous utterance confounds this anticipation, by becoming relevant in an unexpected way.

Curcó goes on to suggest that a speaker uses the above mechanisms to distance herself from the certain assumptions behind the utterance, while expressing a particular attitude towards them. It is this combination of distancing and attitude expression that make for humorous effects, not the mechanisms themselves.

Curcó's work offers some insight into how humour processing might happen. Her mechanisms of humour are a detailed view of what might be meant by "incongruity" in humour — a term few humour researchers bother to define. However, it is difficult to disentangle Curcó's views on humour from the relevance theoretic framework around it, which is unfortunate for researchers who want to use her results without necessarily buying into relevance theory.

2.5.2 Giora

Giora [Giora, 1991] attempts to give well-formedness conditions for semantic jokes. What exactly is meant by "semantic jokes" is unclear, as Giora claims to exclude jokes which manipulate linguistic expectations, yet includes several simple puns which use word-sense ambiguity. Nonetheless, her comments are relevant to most kinds of verbal humour.

Giora's work is based around the *Graded Informative Requirement*, which is the requirement that a text begin with the least informative (i.e. most prototypical) element, and get *gradually* more informative as it goes on. She claims that "a joke is well formed if and only if it:

1. obeys the Relevance Requirement [Grice, 1975]
   and
2. violates the Graded Informativeness Requirement [Giora, 1988] ... in that it ends in a markedly informative message ...
   and
3. causes the reader to perform a linear shift: the reader is made to cancel the first interpretation upon processing the second marked interpretation." [Giora, 1991, p. 470]

In other words, although the information given in a joke is relevant, it is not given in the order of gradually increasing informativeness, as we would expect from a non-joke text. Instead, the final message of the joke is extremely informative, in that it forces the reader to switch from the unmarked interpretation to a much less accessible marked interpretation. This approach is similar to that of Weiner [Weiner and de Palma, 1993], although the range of humour considered is wider and the theoretical background of the authors is different.

Giora gives several examples of how a joke can be reduced to a non-joke by violating one of her conditions of well-formedness. Raskin's favourite example joke:

> "Is the doctor at home?" the patient asked in his bronchial whisper. "No," the doctor's young and pretty wife whispered in reply. "Come right in." [Raskin, 1985]

becomes a non-joke when the Graded Informative Requirement is satisfied — that is, when the information to support the marked interpretation is given gradually:

> "Is the doctor at home?" the patient asked in his bronchial whisper. "No," the doctor's young and pretty wife whispered in reply. "But you are such a handsome man, I guess I can be of some help. Don't you fancy me? Come right in." [Giora, 1991, p. 475]

Giora's third joke well-formedness requirement, that the second interpretation should completely replace the first interpretation, is common to many analyses of humour, including [Weiner and de Palma, 1993]. However, most simple schoolyard puns do not seem to resolve to a single interpretation. For example, when hearing the joke:

> Where do you weigh a whale? *At a whale-weigh station.* [Webb, 1978]

the listener begins by imagining a place to weigh whales; then, when the punchline is heard, imagines a railway station. The railway station interpretation does not replace the one about weighing whales; instead, they seem to be combined into a somewhat nonsensical vision of a place where trains come and go *and* the weight of whales is determined.

Giora finesses this problem by categorising puns as witty *non-jokes*. As such, they need only fulfil requirements 1 and 2 for joke well-formedness. Other examples of witty texts, according to Giora, include metaphors, similes, and advertising slogans. However, it is difficult to accept the categorisation of puns as non-jokes, given the large number of puns in joke books such as [Webb, 1978] and [Metcalf, 1994].

2.6 Other literature

Here we merely touch on some of the other work related to computational humour.

Two AI theorists, Minsky [Minsky, 1980] and Hofstadter [Hofstadter et al., 1989] have expressed an interest in humour, giving it a key place in their philosophical framework for cognition. These works, although interesting, are too general to guide this research.

Minsky's influential paper on humour cognition [Minsky, 1980] is loosely based on Freud's [Freud, 1976] theory of humour, which claims that jokes fool the social censors in our mind by hiding a 'naughty' message behind an 'innocent' story. Minsky expands this idea to include cognitive censors, which prevent our minds from getting into logical loops and other dangerous or unproductive thought processes. He argues that "...humour is involved with how our censors learn ..." [Minsky, 1986, p. 279].

Hofstadter took part in an early workshop on humour and cognition [Hofstadter et al., 1989], at which he pointed out a close relationship between good jokes and bad analogies, in that both involve the blending of frames (clusters of concepts about a particular topic). Hofstadter's 'frame-blend' view of humour is very similar to Raskin's script opposition theory [Raskin, 1985] (see section 2.3.1), but is even more general, and has not been developed further since the workshop. Despite Hofstadter's interest in humour cognition, he is skeptical about the computer generation of humour:

> "We have a long way to go in the scientific study of how human minds work before we will come up with a computer that can produce even a single good joke, let alone a novel."[Hofstadter, 1995, p. 161]

We hope to prove him wrong on this point.

On a different tack, Paulos [Paulos, 1980] discusses in some detail the interesting similarities between mathematics and humour. Axiom systems, like texts, can have several different interpretations, and these interpretations are not necessarily consistent with each other. He explores these parallels at length, discussing how certain features of humour (incongruity, iteration, etc.) might have analogues in mathematics. Paulos, too, holds a low opinion of computers:

> "I should emphasize here that to get (i.e. understand) a joke, either situational or canned, one must ascend, so to speak, to the metalevel at which both interpretations, the familiar and the incongruous, can be imagined and compared (or, if there is only one interpretation, at which its oddness can be appreciated)... These complex operations are all metalevel (or meta-metalevel) activities and are beyond the capabilities of computers and people who want to be computers."[Paulos, 1980, p. 27]

Paulos' book attempts to both express some ideas about humour, and educate the reader on some aspects of mathematics. Unfortunately, the ideas about humour are at too high a level to directly influence our work.

Finally, Katz [Katz, 1993] has attempted to build a neural model of humour processing. It is very simple — a 'concept' is a node in a four node network, and 'incongruity' is a negative connection between two concept nodes — yet the behaviour of this network corresponds to what Katz expects from a brain processing humour. However, almost nothing is known about what a brain does do when it processes humour, so it is very difficult to comment on his claims. More importantly for our purposes, Katz's model is never related to actual humorous texts, either as input or output.

2.7 Conclusion

Although there has been very little work in the area of computer generation of humour, there has been enough reasonably formal work done on the linguistics of humour, particularly riddles, that this research could proceed in an informed manner. However, the relevant previous work did not lay out an obvious path for investigation; instead, it guided decisions along the way.

The work by Pepicello and Green [Pepicello and Green, 1984] suggests that riddles that use what they call phonological ambiguity (rather than morphological or syntactic) might be easier to generate, since phonological similarities between words are well documented (e.g. in [Townsend and Antworth, 1993]).

Pepicello and Green also give the guidelines for a taxonomy based on the strategies the riddles use, as opposed to their surface structure. Such a taxonomy helps determine which subset of riddles using phonological ambiguity are suitable for computer generation (see section 3.2 for more details).

Attardo and Raskin [Attardo and Raskin, 1991] give a general parameterised model for humour, which can be constrained in such a way that it is an informal template for a subtype of humour. JAPE's model (see chapter 3) could be seen as an extremely constrained version of the GTVH; however, several aspects of the GTVH that its creators consider important are not used in JAPE's model (e.g. script opposition).

Chapter 3

The model

3.1 Introduction

This chapter describes a model of some common types of simple punning riddle. First, we examine the regular structures and mechanisms that punning riddles exhibit, and develop a simple taxonomy. Then, having chosen subtypes to model, we examine the kinds of knowledge and mechanisms that would be required to generate riddles of those types. Some of these parts are humour-specific, some are not. We then bring them together into a model of simple punning riddles, and demonstrate that the model adequately describes our chosen domain.

3.2 An exploration of riddle types

3.2.1 Motivation for exploration

The category of question-answer riddles is broad and ill–defined. Our approach to analysing and modelling this domain has been to model a small but representative subset of riddles, implement that model, then expand to other types once good riddles in that subset have been generated. To support this approach, we have developed a simple taxonomy of punning riddles, so that the subtypes to model can be selected in a principled way.

Riddles can be divided into groups in a variety of ways — by subject matter, narrative strategy, source etc. The types of jokes JAPE attempts to generate, however, must be homogeneous on a fairly deep level. After [Pepicello and Green, 1984], we group the riddles twice: first according to the level of ambiguity they use; then, having chosen a subset to investigate further, according to the mechanisms they use to exploit that ambiguity. From these subgroups, we then select several that are well–defined, large enough to include several example riddles, and representative of question-answer riddles in general.

3.2.2 Scope and source of riddles

Most of the riddles discussed below come from "The Crack-a-Joke Book" [Webb, 1978], a collection of jokes chosen by British children, from "Super Silly Animal Riddles" [Ertner, 1993], or from "Super–Duper Jokes" [Young, 1993]. Such books are the ideal source for several reasons. The riddles are simple, requiring only basic English to understand; their humour generally arises from their riddle nature, rather than their subject matter (children don't seem eager to joke about God, politics or taxes, and the books were undoubtedly edited for sex and toilet humour); and there are a huge number of riddles to choose from.

The books include some non-riddle humour, as well: limericks, dialogues, and punning book titles, for example. None of these types will be discussed here, as they are outside our chosen domain. If a given riddle comes from a book, it is cited; if there is no citation, then the riddle was remembered from casual conversation.

3.2.3 Grouping by level of ambiguity

Language is both written and spoken. Pieces of text can be similar in one form, but different in another — for example, the words "cereal" and "serial" are similar phonologically, but look very different on paper. Many riddles take advantage of this fact, by asking the reader (implicitly or explicitly) to accept a falsehood: that pieces of text that are similar in one form are therefore just as similar in another.

Words have a written form (e.g. "carrot"), a spoken or phonological form (e.g. [k,ae,r,ax,t][1]),

[1] Strictly speaking, this is a textual representation of the spoken form, as the spoken form is, well, spoken, and thus difficult to get down on paper. To avoid confusion, we will call it the phonological form. See appendix A for details of the ARPAbet representation.

and a semantic form, or sense. The word "word" is here used to refer to the combination of a written form, a phonological form and a sense. Two words can share senses, phonological forms or written forms. If two words have one form in common, then *that form* of the words is ambiguous. For example, the words with the written forms "carrot" and "karat" share the phonological form [k,ae,r,ax,t]. Thus, the phonological form of "carrot" (and "karat") is ambiguous. Similarly, the words with the phonological forms [w,uw,n,d] and [w,aw,n,d] share a written form, "wound", so that form is ambiguous. Different words can also share a semantic form, or sense, although this is usually called synonymy, rather than ambiguity. Sometimes both the written and phonological forms are shared, while the sense differs. For example, the written form "fair" and the phonological form [f,ea,r] are shared by words with the senses 'carnival' and 'impartial'.

Here we will use "homonyms" to refer to two words that sound the same but are spelled differently, and "homophones" to describe two text segments that sound the same and may or may not have the same spelling.

Groups of words, or *texts*[2], also have two forms and an interpretation, and so can be ambiguous. For example, the texts written "Please announce her" and "Please, an ounce sir" have the same phonological form, and so that form is ambiguous. A text containing an ambiguous part will also be ambiguous, although the context may disambiguate it.

Texts, however, have more complex meanings than single words, and these meanings are built up through several levels of interpretation. Two (or more) interpretations can become apparent at any level. It is common practice (cf. [Pepicello and Green, 1984]) to taxonomize ambiguity according to the level of processing at which it becomes apparent that the text has two meanings. However, different researchers have different models of text processing, so different taxonomies of ambiguity arise.

Here we attempt to justify our labelling of types of ambiguity, and discuss how they are commonly used in jokes.

[2] Here, "text" is shorthand for "group of words", and is not meant to suggest that the group of words is in written form.

Types of ambiguity found in the Crack-a-Joke book

As discussed in chapter 2, Pepicello and Green [Pepicello and Green, 1984] describe riddles as exploiting linguistic ambiguity at the phonological, morphological or syntactic level. Although there are riddles that use two or more types of ambiguity, it is relatively straightforward to divide the bulk of question-answer riddles according to the primary level of ambiguity they use. For example:

1. **Phonologically ambiguous:** What do you call a cat that sucks acid drops? *A sour puss.* [Webb, 1978]
2. **Morphologically ambiguous:** Why is coffee like soil? *It is ground.* [Pepicello & Green 84]
3. **Syntactically ambiguous:** What grows up while it grows down? *A baby duckling.* [Webb, 1978]

As [Pepicello and Green, 1984] describe them, these three types of ambiguity are all *low–level* ambiguities. Low–level ambiguities are those which allow several different senses for a word or words in a spoken or written text. In texts that have *high–level* ambiguity, the senses of the individual words are not in question, only the interpretation of the whole text. We divide low–level ambiguities slightly differently from [Pepicello and Green, 1984]. The three low–level ambiguities that we have identified are:

spelling ambiguity: This is where one phonological form corresponds to two (or more) written forms and senses. For example, the phoneme sequence [s,ia,r,ia,l] could be written either as "cereal" or as "serial". The ambiguity lies in the fact that there are several possible phoneme to text mappings. This joke uses spelling ambiguity:

> What do you get when you cross a rabbit with a lawn sprinkler? *Hare spray.* [Young, 1993]

pronunciation ambiguity: This is when one written form corresponds to two (or more) phonological forms and senses. For example, the written form "wind" corresponds to the phonological forms [w,ih,n,d] (n. moving air) and [w,ay,n,d] (v. twist). For example:

What do you call a blizzard forecast on April 1st? *A wind–up.*

word sense ambiguity: This is when one phonological form *and* one written form correspond to two or more senses. When the word is spoken *or* read, the listener/reader cannot tell which sense of the word is meant. For example, "bank" is word sense ambiguous, in that it has two senses: the side of a river, and a financial institution. Here is a riddle that uses word sense ambiguity:

Where do snowmen keep their money? *In snow banks.* [Young, 1993]

There are other low–level ambiguities involved in phoneme segmentation and identification (in the case of speech) and letter identification (in the case of writing). However, in order to keep the discussion relatively focussed, we will only consider processing which occurs after the phonemes (speech) or letters (writing) have been accurately identified.

There are also some jokes in our corpus which contain high–level ambiguities. For example:

1. Which would you rather, an elephant charge you or a gorilla? *I'd rather the elephant charge the gorilla.* [Pepicello and Green, 1984]
2. Why do birds fly south in winter? *Because it's too far to walk.* [Webb, 1978]
3. How can you tell if an elephant has been in your fridge? *Footprints in the butter.* [Webb, 1978]

The question part of joke 1 can be interpreted as either "Which would you rather, an elephant charge you or a gorilla charge you?" or "Which would you rather, an elephant charge you or an elephant charge a gorilla?". Note that none of the individual written words are ambiguous — the sense of each of the words is clear. It is only the interpretation of the whole sentence which is in doubt. In this case, some material was elided from the two unambiguous sentences above, to produce the ambiguous text.

The question part of joke 2, on the other hand, is ambiguous because the focus of the sentence cannot be determined without more information. The "why" in the question could be questioning any part of the statement "Birds fly south in winter". In speech, such questions are disambiguated by intonation: "Why do birds **fly** south in winter?" is the question answered by the punchline in joke 2; whereas "Why do birds fly **south in winter**?" is the more usual reading, and could be answered with "Because winter in the north is too cold for them." The two versions could be paraphrased as "Given that birds go south in winter, why do they fly?" and "Given that birds fly, why do they go south in winter?". Again, this is a high–level ambiguity because none of the senses of the words are in doubt.

Joke 3 uses what Pepicello and Green call *contextual* ambiguity, "ambiguity produced through a conscious manipulation of social decorum that results in disorientation or confusion of the riddlee" [Pepicello and Green, 1984]. For example, one could argue that joke 3 is contextually ambiguous because it asks the riddlee about an elephant, then describes an entity whose properties are inconsistent with the essential features of being an elephant (i.e. being very large). The riddlee is then left unable to decide if the subject of conversation is elephants or not. We, however, would argue that there is no linguistic ambiguity as such in joke 3, because neither the question nor the punchline have more than one interpretation.

Other jokes use what could be called *pragmatic* ambiguity. In these, the illocutionary or perlocutionary force of the text is deliberately confused. One classic example of this (admittedly, not a riddle) is:

> Waiter, there's a fly in my soup!
> *Don't shout so loud, sir, or everyone will want one.*

because it confuses the usual reason for shouting about flies in soup (to complain), with an alternative explanation (excitement about good fortune).

In fact, it seems that a joke can be constructed around any linguistic ambiguity. Some types, however, are more common than others. Although jokes of other types do appear, the bulk of the riddles in our chosen corpus use low–level ambiguity. Pronunciation ambiguity, however, is much less common in jokes than spelling ambiguity and word–sense ambiguity, perhaps because it is less common in the language in general.

Suitability for this research

Of the types described in this section, riddles containing low–level ambiguities are most appropriate for this research, for several reasons. Information about the pronunciation of words is available, whether a phonological description of a word (such as one might find in a dictionary), or a list of phonologically similar words (homonyms or near–homonyms). High–level ambiguities, on the other hand, are related to the whole structure of the sentence, by definition, which would probably make them more difficult to generate, if not to detect.

More importantly, the strategies employed by riddles containing low–level ambiguities are simple and general (see section 3.2.5 below), whereas the strategies employed by the other kinds of riddle are often specific to the exact ambiguity used in that particular joke.

We have, however, chosen not to model jokes which contain pronunciation ambiguities, for several reasons. Because pronunciation ambiguities are relatively rare in English, and because our corpus of joke books contained jokes in written form, very few jokes with pronunciation ambiguities appeared. This left us few jokes of this type to analyse. Moreover, jokes containing pronunciation ambiguities usually have to be pronounced to be understood, which would complicate the evaluation of this type of riddle.

The kinds of high–level ambiguity mentioned, such as focus, contextual or pragmatic ambiguity, are also not common enough in our corpus to allow us to generalise about their structures and strategies. Word sense and spelling ambiguities, however, are used in over half the riddles in our corpus, which simplifies the task of identifying common structures and strategies.

For these reasons, for the remainder of this thesis, we concentrate on question-answer riddles that use spelling or word sense ambiguity. From now on, riddles of this type will be called *puns*[3] or *punning riddles.*

3.2.4 Confusability

It is useful in discussing riddles to introduce the idea of 'confusability'.

If an utterance[4] is spelling ambiguous, then it has more than one written form. These written forms are *completely confusable*, in that it is impossible to tell which written form was intended by the original utterance. For example, the phoneme sequence [s,ia,r,ia,l] is spelling ambiguous, and it has two written forms: "cereal" and "serial". These two written forms are completely confusable. Likewise, if a written text is pronunciation ambiguous, then it has two phonological forms. These phonological forms are completely confusable.

Often, segments of text which are only *similar* in one form, rather than identical, are confused with each other in a joke. For example:

> What does a near-sighted ghost wear? *Spooktacles.* [Webb, 1978]

The word segment "spec" and the word "spook" are not spelled or pronounced the same, yet one is substituted for the other in this joke. They are *partially confusable.* Two texts which contain confusable *or identical* segments are considered to be partially confusable. In this case, "spec" and "spook" each start with the phoneme sequence [s,p], and end with [k]. There are regular ways of being partially confusable, such as rhyming or alliterating. For example, rhyming words (such as "cat" and "hat") are partially confusable, because both end with the same phoneme sequence (i.e. [ae,t]).

Metathesis is a special way of producing partially confusable pairs of texts, whereby confusable text segments are exchanged. Metathesis is used, for example, in spoonerisms like "cat and parrots" and "pat and carrots". Two identical pairs of segments (the "at" of "cat" and "pat", and the "arrots" of "parrots" and "carrots") are exchanged, resulting in two partially confusable phrases. Another type of metathesis involves the

[3] Note that other works may use the word "pun" differently.

[4] Here "utterance" is shorthand for "spoken text". Other works may use this term differently.

exchange of whole words, rather than parts of words. For example, the words "bear" and "arm" both have two meanings, and the pair of words can be arranged in two different ways: "bear arms" and "arm bears". The resulting phrases are also partially confusable.

Any type of segment (syllable, word, phrase) can be confused with any other type of segment, for example:

- a word with a word ("lynx" with "links")
- a part of a word (syllable) with a word (the word "fan" with the first syllable of the word "fantastic")
- a syllable with a syllable (the joke word "a VCRdvark", assuming letters in an acronym are syllables, confuses the last syllable of "VCR" with the first syllable of "aardvark")
- a word with a phrase ("notwithstanding" with "not with standing")
- a phrase with a phrase ("melancholy baby", "melon collie baby")

Unfortunately, as defined above, almost any pair of texts could be considered partially confusable, as long as they have any segment in common. How similar two confusable texts have to be to be used in a joke is an open question. Here, we concentrate on texts that are partially confusable in a regular way, such as rhyming, alliterating or spoonerizing.

3.2.5 Strategies used in punning riddles

There are three main strategies used in puns to exploit spelling or word sense ambiguity: *juxtaposition*, *substitution*, and *comparison*. This is not to say that other strategies do not exist; however, none were found among the large number of punning jokes examined.

Juxtaposition

Juxtaposition is the most simple mechanism, simply placing the confusable segments near each other and treating them as a normal construction. For example:

> What do you call a weird market? *A bizarre bazaar.* [JAPE]
>
> What do you get if you cross a dog and a kangaroo? *A pooch with a pouch.* [Ertner, 1993]

Jokes which use juxtaposition alone are really very weak, as they do little more than demonstrate an unexpected phonological similarity.

Because jokes which use juxtaposition alone are very weak, juxtaposition is often used in combination with other techniques. For example, the image brought to mind is often funny in and of itself - thus combining a linguistic joke with a visual joke. However, jokes which rely on the imagery they produce are not strictly linguistic humour - they are complex jokes which combine linguistic and visual humour. Jokes which use juxtaposition alone are still jokes, albeit weak, without the visual element.

Substitution

This mechanism works by substituting one confusable segment for another, as part of a larger text, and using the resulting text as if it were a sensible construction. That is, the constructed text is used normally, but with a new interpretation that is a combination of the interpretations of the elements which make it up.

For example, the word "purr" is confusable with the first syllable of "purgatory". If we substitute "purr" for "pur", we get the constructed text "purrgatory". We must also construct a plausible interpretation for the construction — "cat afterlife", for example. If the constructed text and its new interpretation are used as if they were normal, we get a joke. For example:

> Where do cats go when they die? *Purrgatory.* [Ertner, 1993]

Sometimes, although rarely, the substitution is not actually carried out, but is implied. That is, the written form of the text containing the confusable segment is not altered, but its interpretation is changed as if the substitution had taken place. For example:

> What's an octopus? *A cat with eight legs.* [Webb, 1978]

If the substitution had been completed, "octopus" would have been spelled "octopuss" — which would have given away the joke, since "octopus" appears (unusually) in the question part of the riddle. Instead, the spelling of "octopus" was left unchanged, but its interpretation was changed as if the substitution had taken place, to "a cat with eight legs".

With substitution jokes, it should be noted that the interpretation of the larger text into which a confusable piece is substituted also plays a role. Often, the larger text (usually the punchline) is a familiar phrase or saying. This tends to make the joke stronger, since the 'usual' interpretation of a familiar phrase is more strongly evoked than that of a more neutral piece of text, which makes the constructed interpretation more incongruous. If the larger piece of text is not already familiar to the listener, then the joke will have to 'work harder' to suggest both interpretations.

Comparison

This mechanism explicitly compares two confusable texts, usually by asking for similarities or differences in the question part of the riddle. Positive comparison riddles ask for similarities ("How is A like B?"), and negative comparison riddles ask for differences ("What is the difference between A and B?"), although the phrasing of the question can vary quite a bit.

Negative comparison riddles most often contain pairs of texts constructed with metathesis. Metathesis pairs are similar in a regular way (they have exchanged sounds), and are confusable. Spoonerisms are a kind of metathesis, in which the initial sounds of a phrase are exchanged to form a pair of confusable texts (e.g. "cute mitten" and "mute kitten") The comparison mechanism asks the listener to compare the (very different) interpretations of the confusable texts. For example:

> What's the difference between a short witch and a deer running from the hunters? *One's a stunted hag and the other's a hunted stag.* [Webb, 1978]

The phrases "stunted hag" and "hunted stag" form a partially confusable metathesis pair. The normal interpretations of these phrases have little, if anything, in common, yet the question asks what the difference between them is, implying that they are similar. The joke answer then suggests that the only difference between them is the exchange of initial sounds.

Negative comparison riddles using metathesis are so common that they are often left uncompleted, for the listener to figure out. This has the advantage of giving the listener the added pleasure of solving a puzzle, and often disguises a taboo word. For example:

> What's the difference between a rooster and a lawyer? *One clucks defiance...*

However, not all comparison riddles use metathesis, and not all riddles containing metathesis pairs are comparison riddles. For example, this riddle:

> How is a red–haired loonie like a biscuit? *They're both ginger nuts.* [Webb, 1978]

is a positive comparison riddle, and it does not contain a metathesis pair; whereas this riddle:

> What do you get if you cross a bear, a bee and a rabbit? A honey bear with bunny hair.

contains metathesis but is not a comparison riddle. Riddles such as these, however, are quite rare.

3.2.6 The chosen domain

The model which is presented in the remainder of this chapter describes several significant subtypes of punning riddle. A punning riddle, for our purposes, is a question-answer riddle based on a word sense or spelling ambiguity. These use three types of mechanism: juxtaposition, substitution, and comparison. Our model describes riddles using all three mechanisms.

3.3 Overview of model

Now that we have restricted the goal set of jokes to simple punning question-answer riddles, it is necessary to describe the common features of their structure in such a way that they could be constructed by a computer.

These features must be specified well enough that any piece of text that has them is a joke. It is *not* adequate to say that there are some jokes among the pieces of text that share these features, since the implemented model does not have a sense of humour with which to sift its output. Although heuristics for rating the jokes can be added, the texts that fit the model should be recognisably jokes (if not always good ones). On the other hand, not all jokes (nor even all question-answer punning riddles) need have these features — to completely specify everything that could make text funny is too ambitious a task for this research.

The mechanisms we have described so far — juxtaposition, substitution, and comparison — take advantage of this confusion of levels in different ways. However, they all work by constructing a non-lexicalized (i.e. not already in the lexicon) segment of text. The riddle then uses the resulting constructed word or phrase as if it were a semantically sensible construction. The effective meaning of this construction is a combination of the meanings of the pieces of text used to build it. For example, here is a joke which uses substitution:

> What do you give an elephant that's exhausted? *Trunkquillizers.* [Webb, 1978]

In this joke, the word "trunk", which is confusable with the syllable "tranq", is substituted into the valid English word "tranquillizer". The constructed word "trunkquillizer" is given a meaning, referred to in the question part of the riddle, which is some combination of the meanings of "trunk" and "tranquillizer":

trunkquillizer: A tranquillizer for elephants.

Note that this is not the only meaning for "trunkquillizer" that could produce valid jokes. For example:

> What kind of medicine gives you a long nose? *Trunkquillizers.*

is a joke (if not a good one) based on a different definition for "trunkquillizer" — namely, 'a medicine that gives you a trunk'. The constructed meaning combines notable semantic features of both of the valid words/phrases used to construct it, so that the riddle question is a reasonable description of the constructed concept.

In non-humorous communication, we can use the meaning of a real word/phrase to build a question that has the word/phrase as an answer:

> What do you call someone who douses flames? *A firefighter.*

Similarly, riddles like the 'trunkquillizer' example use the constructed meaning of a constructed word or phrase to build a question that *would* have that word or phrase as its answer, if it really existed. This question becomes the first part of the riddle, and the constructed word/phrase becomes the punchline.

At this point, it is important to distinguish between the task of building the *meaning* of the constructed segment, and the task of using that meaning to build a question with that segment as an answer. For example, the following questions use the same meaning for 'trunkquillizer', but refer to that meaning in different ways:

- What do you use to sedate an elephant?
- What do you call elephant sedatives?
- What kind of medicine do you give to a stressed-out elephant?

On the other hand, *these* questions are all put together in the same way, but from different constructed meanings:

- What do you use to sedate an elephant?

- What do you use to sedate a piece of luggage?
- What do you use to medicate a nose?

So, there are several different jobs to be done in constructing a joke of this type:

1. A non-lexicalized word/phrase must be constructed.
2. A plausible meaning for that segment must be built.
3. A question-answer pair must be constructed, using the word/phrase and its meaning.

Of these, only the first is humour-specific, in that it uses juxtaposition, substitution or comparison to build the word/phrase. The second and third are general linguistic tasks. Given a sensible word/phrase, step two would construct a sensible meaning for that phrase; likewise, step three would construct a reasonable question-answer pair. In this model, step one is done by the schemata (section 3.5); step two is done by the small adequate description generator (section 3.6); and step three is done by the templates (section 3.7). Briefly,

1. **Schemata** constrain and assert relations between lexical items and constructed items (including descriptions).
2. The **small adequate description generator**, given a sequence of lexemes, constructs a plausible description of the concept referred to by that sequence.
3. **Templates** translate relations (between items and descriptions) into question–answer pairs.

Each of these steps requires knowledge in order to be successful. Some of this knowledge is phonological, semantic and syntactic information about the words used to construct the riddle (section 3.4); some is information about what constitutes a plausible, adequate description of something (section 3.6); and some is knowledge about what forms a question-answer pair can take (section 3.7).

The remainder of this chapter describes these mechanisms and knowledge sources in some detail.

3.4 Lexicon

In order to construct jokes based on linguistic ambiguity, the system must have access to large amounts of linguistic information. In particular, it requires:

phonological information, so that the phonological similarities or differences between two texts can be determined;

semantic information, which can be used to evoke particular concepts or words in the listener;

syntactic information, so that the words can be used grammatically in the final joke; and

surface form information, so that similarities in the appearance of two texts (i.e. their spelling) can be determined.

The store of this information is the lexicon. The lexicon required for our needs is not humour-specific; that is, it contains no information put there for the purpose of creating jokes. The more entries the lexicon has, and the more information in each entry, the more puns can be created from that information. The information contained in the lexicon need not be all from one source; it could be several specialist lexicons (e.g. one containing only phonological information, one containing only syntactic, etc) joined together.

In this model, we assume that the lexicon contains a number of entries, each attached to a *lexeme*, a unique identifying symbol for the entry. The entry contains phonological, semantic, syntactic and psycholinguistic information about a word or phrase, as well as that word or phrase's written form. The phonological information consists of a sequence of phonemes, which is the pronunciation of the word or phrase. For the sake of this discussion, we will represent the phonemes using the ARPAbet system[5]. For example,

[5] ARPAbet is the system used in the British English Example Pronunciation (BEEP) dictionary [Robinson, 1996].

the entry for the word "lemon" would contain the sequence of phonemes [l,eh,m,ax,n]. The semantic information is a list of semantic relations between this lexeme and other lexemes in the lexicon. For example, the entry for "lemon" might include the set of relations: {class(lemon, citrus_fruit), specifier(lemon, yellow), adjective(lemon, oval), has(lemon, pip)}. Not much syntactic information is required for this model; however, the syntactic category (i.e. noun, verb, adjective, compound nominal etc) of the word or phrase is important.

Which semantic relations are available in the lexicon affects other parts of the system, in particular, the rules used to generate small adequate descriptions (see section 3.6.1). These rules can be modified to suit the semantic relations available. However, for the sake of discussion, we will assume that the lexicon can contain the following semantic relations:

class: relates the lexeme to its supertype (e.g. class(lemon, citrus_fruit)).

specifier: relates the lexeme to the feature that distinguishes it from most of the other items in its class (e.g. specifier(lemon, yellow)).

adjective: relates the lexeme to an adjective that describes it (e.g. adjective(lemon, oval)).

has: relates the lexeme to some part of itself (e.g. has(lemon, pip)).

act_verb: relates the lexeme to some action it typically performs (e.g. act_verb(plant, grow)).

inact_verb: relates the lexeme to some action that is typically performed on it (e.g. inact_verb(plant, harvest)).

Any lexicon containing the above information about its entries should work with this model, assuming a suitable interface can be constructed. However, it should be noted that most of the schemata (see section 3.5) are heavily constrained, so that a lexicon would need to be quite large before jokes based on the information contained in it could be found.

3.5 Schemata

A schema specifies the relationships between lexical items (which are explicitly represented in the lexicon) and 'constructed' concepts. It also relates lexemes to small adequate descriptions (SADs; see section 3.6). The schema itself does not *build* such descriptions — in our system, this is done by the SAD generator — but it does *constrain* them.

A schema consists of a set of variables, constraints which must hold between those variables, and assertions of relations between the variables. All variables are existentially qualified, and are instantiated as the schema is 'filled in'. There are two kinds of constraints on the variables: *lexical preconditions*, which are lexical relationships which must hold between the instantiated variables; and *SAD constraints*, which, when instantiated, are relations between sequences of lexemes and SADs.

A schema has three slots:

Lexical preconditions: This slot contains the lexical preconditions that must be true (i.e. found in the lexicon) for the schema to be applicable. These can be constraints on lexemes, phonemes, or surface text.

SAD constraints: This slot contains a set of **described_by** relations. When instantiated, the **described_by** relation relates a sequence of lexemes to a SAD which plausibly describes the concept to which the sequence of relations refers. Our implementation (see chapter 4) checks the SAD constraints by sending them to the SAD generator; if an appropriate SAD can be constructed, the constraint is satisfied. However, *any* SADs which satisfy the constraints in this slot can be used in a pun, no matter how they are generated. See section 3.6 for a description of how SADs are represented and generated.

Relationships: This slot contains a set of asserted relations. These relations are asserted for the purposes of constructing a pun; they need not be consistent with information in the lexicon. The only relations that are permissible in this slot are:

describes/2, a relation between a SAD and the text it describes.

describes_same/2, a relation between two SADs that describe the same concept.

describes_diff/2, a relation between two SADs that describe *different* concepts.

For example, here is a schema which substitutes a word into a noun phrase:

```
{Lexical preconditions:
        noun_phrase(NPLex)
        component_lexemes(NPLex, LexA, LexB)
        written_form(LexA, WordA)
        homophone(WordA, HomWord)
        written_form([HomLex], HomWord)
        written_form([HomLex, LexB], NPWF)
 SAD constraints:
        described_by([HomLex, NPLex], Desc)
 Relationships:
        describes(NPWF, Desc)}
```

In this notation, a schema begins with a left brace, then the text "Lexical preconditions:", followed by any number of lexical constraints. Then, the text "SAD constraints:" is followed by any number of **described_by** relations. Then, the text "Relationships:", is followed by any number of asserted relations (of the types described above), which are followed by a right brace. Here we adopt standard predicate logic notation, as well as the Prolog convention that a term beginning with an uppercase letter is a variable.

For the above schema to be applied, the lexical preconditions require that there be a compound nominal lexeme (NPLex) which can be divided into two component lexemes (LexA and LexB). The written form of LexA (WordA) must have a homophone (HomWord) which can be found in the lexicon as a lexeme (HomLex). It must be possible to construct a grammatically correct compound nominal (NPWF) from HomLex and

LexB. The SAD constraints require that there must be some SAD which plausibly describes the concept referred to by the lexeme sequence [HomLex, NPLex]. If all the constraints can be satisfied, it is asserted that this SAD describes the constructed compound nominal, NPWF.

Seen procedurally, this schema constructs a new compound nominal by substituting a homophone for the first word in a lexicalized compound nominal, and selects the entries for the original compound nominal and the homophone as those from which the constructed compound nominal's SADs will be built. Henceforth, we will describe schemata procedurally, for conciseness; however, please keep in mind that, in our model, schemata consist of constraints and assertions.

For example, this schema could be instantiated as follows:

```
{Lexical preconditions:
        noun_phrase(serial_killer)
        component_lexemes(serial_killer, [serial, killer])
        written_form(serial, ``serial'')
        homophone(``serial'', ``cereal'')
        written_form([cereal], ``cereal'')
        written_form([cereal, killer], ``cereal killer'')
 SAD constraints:
        described_by([cereal, serial_killer], {class(Lex, murderer), has(Lex,
        fibre)})
 Relationships:
        describes(``cereal killer'', {class(Lex, murderer), has(Lex,
        fibre)})}
```

The above instantiated schema, with an appropriate template to produce the surface form of the question-answer pair, gives the riddle:

What do you call a murderer with fibre? *A cereal killer.* [JAPE]

The schema described above uses substitution to exploit the word-sense or spelling ambiguity of two homophones. Other schemata use different mechanisms to exploit different kinds of ambiguity. For example, this schema juxtaposes two homophones, to construct a compound nominal:

```
{Lexical preconditions:
  homophone(WordA, WordB)
  written_form(LexA, WordA)
  written_form(LexB, WordB)
  noun(LexB)
  written_form([LexA,LexB], WFAB)
 SAD constraints:
  described_by([LexA,LexB], SAD)
 Relationships:
  describes(WFAB, SAD)}
```

This might be instantiated as follows:

```
{Lexical preconditions:
  homophone(``bizarre'', ``bazaar'')
  written_form(bizarre, ``bizarre'')
  written_form(bazaar, ``bazaar'')
  noun(bazaar)
  written_form([bizarre,bazaar], ``bizarre bazaar'')
 SAD constraints:
  described_by([bizarre,bazaar],{class(Lex, market),
                                 specifier(Lex, strange)})
 Relationships:
  describes(``bizarre bazaar'',{class(Lex, market),
            specifier(Lex, strange)})}
```

This might be used to construct the riddle:

> What do you call a strange market? *A bizarre bazaar.* [JAPE]

There are also schemata which use the comparison mechanism. For example, this schema compares two confusable phrases, constructed with metathesis:

```
{Lexical preconditions:
  rhyme(WordA, WordB)
  rhyme(WordC, WordD)
  alliterate(WordA, WordC)
  alliterate(WordB, WordD)
  written_form(WordC, LexC)
  written_form(WordD, LexD)
  noun(LexC)
  noun(LexD)
  written_form(LexA, WordA)
  written_form(LexB, WordB)
  written_form([LexA, LexD], WFAD)
  written_form([LexB, LexC], WFBC)
SAD constraints:
  described_by([LexA, LexD], SAD1)
  described_by([LexB, LexC], SAD2)
Relations:
  describes(WFAD, SAD1)
  describes(WFBC, SAD2)}
```

This schema can only be applied if a pair of grammatical spoonerisms can be constructed. Here is how it could be instantiated:

```
 {Lexical preconditions:
  rhyme(``cute'', ``mute'')
  rhyme(``kitten'', ``mitten'')
  alliterate(``cute'', ``kitten'')
  alliterate(``mute'', ``mitten'')
  written_form(kitten, ``kitten'')
  written_form(mitten, ``mitten'')
  noun(kitten)
  noun(mitten)
  written_form(cute, ``cute'')
  written_form(mute, ``mute'')
  written_form([cute, mitten], ``cute mitten'')
  written_form([mute, kitten], ``mute kitten'')
SAD constraints:
  described_by([cute, mitten],
              {class(Lex1, glove), specifier(Lex1, pretty)})
  described_by([mute, kitten],
              {class(Lex2, cat), specifier(Lex2, silent)})
Relations:
  describes(``cute mitten'', {class(Lex1, glove), specifier(Lex1,pretty)})
  describes(``mute kitten'', {class(Lex2, cat), specifier(Lex2, silent)})}
```

With an appropriate template, this would form the joke:

> What's the difference between a pretty glove and a silent cat? *One's a cute mitten, the other's a mute kitten.* [Ertner, 1993]

In all, there are thirteen schemata. See appendix J for a full listing of the schemata, and examples of the jokes they could produce. There could, of course, be more schemata, as there are many different specific ways of exploiting low–level linguistic ambiguities for humorous effect, but these thirteen are very common, and are representative of punning mechanisms in general.

The schemata were given names, of no particular significance, for convenience. Here they are qualitatively described, with examples from joke books (see appendix J for examples from JAPE):

phonsub: This schema substitutes a word for a confusable segment of another word. The SAD is constructed from the entries for both words. For example:

> What kind of ears do engines have? *Engine-ears.* [Girling, 1988a]

coatshed: This schema negatively compares two confusable verb-object phrases, constructed by reversing the order of two pairs of homophones. For example:

> What's the difference between a hairy dog and a painter? *One sheds his coat and one coats his shed.*[Ertner, 1993]

hopchew: Similar to **coatshed**, this schema negatively compares two confusable verb phrases, constructed with metathesis. For example:

> What's the difference between a hungry kangaroo and a lumberjack? *One hops and chews, the other chops and hews.* [Ertner, 1993]

negcomp: This schema, discussed above, negatively compares two confusable compound nominals, constructed with metathesis. For example:

> What's the difference between a pretty glove and a silent cat? *One's a cute mitten, the other's a mute kitten.* [Ertner, 1993]

poscomp: This schema constructs a word-sense ambiguous phrase, and positively compares its two senses. For example:

> How's a red–haired loonie like a biscuit? *They're both ginger nuts.* [Webb, 1978]

lotus: This schema, discussed above, substitutes a homophone for the first word in a lexicalized compound nominal. The SAD is constructed from the entries for the homophone and the noun phrase. For example:

> What do you call a hairy beast in a river? *A weir–wolf.* [Forrester, 1994][6]

rhyming_lotus: Similar to **lotus**, this schema substitutes a rhyming word for the first word in a lexicalized compound nominal. The SAD is constructed from the entries for the rhyming word and the compound nominal. For example:

> What do you call a police dog? *A copper spaniel.* [Young, 1993]

elan: This schema substitutes a homophone for the last word in a lexicalized compound nominal. The SAD is constructed from the entries for the homophone and the compound nominal. For example:

> How wide is every cemetery? *A grave yard.* [Young, 1993]

jumper: This schema substitutes a homophone for the second word in a lexicalized compound nominal. The SAD is constructed from the entries for the homophone and the first word in the compound nominal. For example:

> Where do hamburgers like to dance? *At a meat ball.* [Young, 1993]

woolly: This schema substitutes a homophone for the first word in a lexicalized compound nominal. The SAD is constructed from the entries for the homophone and the second word in the compound nominal. For example:

> What do you call a cat that likes to go bowling? *An alley cat.*[Young, 1993]

double_pun: This schema substitutes homophones for both the words in a lexicalized compound nominal. The SAD is constructed from the the entries for the homophones. For example:

[6] The Vampire Joke Book [Forrester, 1994] seems to have a fondness for this form, as it also has jokes based on "wear–wolf" ("What do you call a hairy beast with clothes on?"), "were–wolf" ("What do you call a hairy beast that no longer exists?") and "where–wolf" ("What do you call a hairy beast that's lost?").

What is an astronaut's favorite pub? The space bar. [Young, 1993]

bazaar: This schema, discussed above, juxtaposes two homophones in a compound nominal. The SAD is generated from the entries for the homophones. For example:

> How does a whale cry? *Blubber blubber.* [Young, 1993]

vn: This schema is similar to the **coatshed** schema, described above, but with fewer constraints. It too negatively compares two confusable verb-object phrases, constructed by reversing the order of two pairs of homophones — but instead of comparing what the items are able to do, it compares what one is able to do with what the other is *unable* to do.

> What's the difference between an elephant and a flea? *An elephant can have fleas, but a flea can't have elephants.* [Young, 1993]

The above schemata are the only humour-specific element of this model of punning riddles. They each use either substitution, juxtaposition, or comparison to take advantage of some kind of low-level linguistic ambiguity. Between them, they describe the humour-producing mechanisms of a large proportion of punning riddles.

3.6 Small adequate descriptions

The lexicon entry for any lexeme is a set of relationships to other lexemes. All of these contribute to the 'meaning' of the lexeme. However, not all of these are necessary to describe the entity in question. For example, the entry for a lemon may include the relations class(lemon, citrus_fruit), adjective(lemon, sour), specifier(lemon, yellow), has(lemon, peel), has(lemon, seeds), etc. However, in order to adequately describe a lemon to someone, it is only necessary to say "a yellow citrus fruit". A lemon is not the only thing that fits this description, but is one of the first two or three answers someone will come up with if asked to name a yellow citrus fruit. "A yellow citrus fruit" would be a reasonable crossword clue, if the answer was meant to be "lemon", whereas "something yellow" would not, as there are too many things that fit that description.

Certain sets of relationships, taken from a lexeme's entry, combine to make an adequate description of that lexeme. For example, a lexeme's synonym is always an adequate description of it (e.g. {syn(tedious,boring)} is an adequate description of the lexeme 'tedious'). For a noun lexeme, its specifier and its class are an adequate description (e.g. {class(boar, pig), specifier(boar, wild)} is an adequate description of the lexeme 'boar').

The whole entry for the lexeme is also an adequate description. However, what is needed for our purposes is a minimal adequate description; that is, a set of relationships which, if any one were removed, would no longer constitute an adequate description. However, since what constitutes a minimal adequate description varies from person to person, we will instead use a small adequate description (SAD) — adequate, but somewhat smaller than the whole entry. A SAD will be represented as a set of instantiated relationships. For example, a SAD for 'lemon' would be:

```
{class(lemon,citrus_fruit), specifier(lemon,yellow)}.
```

A lexeme can have several SADs. We will represent this as the lexeme being described, separated with a colon from the SADs that describe it, enclosed in angle brackets. For example:

```
<lemon: {class(lemon,citrus_fruit), specifier(lemon,yellow)},
{class(lemon,citrus_fruit),has(lemon,seeds)}>
```

For any particular slot-based lexicon, it should be possible to list sets of relationship names such that, for any lexeme, those relationships from the lexeme's entry (if they exist) constitute an adequate description for that lexeme. For example, as discussed above, {synonym} is such a set, since a lexeme's synonym will always constitute an adequate description of it. It should be possible to determine a set of such sets from the definitions of the slots for any lexicon. These sets of uninstantiated relationships will be called USADs (Uninstantiated SADs). For example, one version of our system uses a hand–built lexicon (see section 4.3). The set of USADs for that lexicon is:

```
{{synonym}, {specifier, class}, {cross-between}, {adj, inact-verb},
```

```
{inact-verb, location}, {has, adj}, {class, inact-verb}}.
```

The set of USADs will vary from lexicon to lexicon, depending on the slots available in the lexicon. If the slots, and the kind of information that can fill them, are clearly defined, then constructing a set of USADs for any lexicon should be possible.

Given the USADs for the lexicon, a genuine lexeme, and its entry, it is straightforward to generate one or more SADs for that lexeme. For example, if the set of USADs for our lexicon is as given above, and the entry for 'lemon' includes the relationships {specifier(lemon, yellow), class(lemon, citrus_fruit), adjective(lemon, sour), inact_verb(lemon, eat), has(lemon, seeds)}, then we would get the SADs:

```
<lemon:
{specifier(lemon, yellow), class(lemon, citrus_fruit)},
{adjective(lemon, sour), inact-verb(lemon, eat)},
{has(lemon, seeds), adjective(lemon, sour)},
{class(lemon, citrus_fruit), inact-verb(lemon,eat)}>
```

Note that there is not a SAD for every USAD in this case, since the relationships synonym, cross-between, and location are not used in the entry for 'lemon'.

3.6.1 The small adequate description generator

Generating a SAD for a lexicalized item is simply a matter of picking the USAD relationships out of the item's lexical entry. Generating a SAD for a non-lexicalized item, such as the words and phrases constructed by the schemata discussed in section 3.5, is more difficult, as they do not *have* lexicon entries from which to select USAD relationships. Instead, the entry(ies) from which the USAD relations are to be derived must be explicitly specified. In our model, such entries are specified by a *sequence of lexemes*, and the *SAD generator* derives the USAD relations from the lexical entries corresponding to the lexemes in the sequence.

In the case of non-lexicalized items constructed by a schema, the instantiated schema specifies a suitable sequence of lexemes in the 'SAD constraints' slot, using the **described_by/2** relation, in which the first argument is a sequence of lexemes, and the second argument is instantiated to a SAD generated by the SAD generator. However, the SAD generator should be able to construct plausible SADs for *any* sequence of lexemes, under two assumptions: that the last lexeme in the sequence is the *head* of the phrase being described, and that the other lexemes *modify* the head. See section 3.6.2 for a discussion of heads and modifiers.

So, the SAD generator takes a sequence of two or more lexemes, and generates a set of SADs, which should all be plausible descriptions of the concept referred to by the sequence. In our model of puns, the sequence of lexemes is from **described_by** relations in the 'SAD constraints' slot in an instantiated schema. For example, an instantiated jumper schema, as described in section 3.5, could give the SAD generator the lexeme sequence [lemon, aide], to build a SAD for the constructed phrase "lemon aide".

If the lexicon entry for 'lemon' is:

```
{specifier(lemon, yellow), class(lemon,citrus_fruit),adjective(lemon,sour),
inact_verb(lemon,eat), has(seeds)}
```

and the lexicon entry for 'aide' is:

```
{synonym(aide, assistant), act_verb(aide, help), adjective(aide, helpful)}
```

then the SAD generator might generate the following set of SADs:

```
<lemon_aide: {class(lemon_aide, assistant), specifier(lemon_aide, sour)},
 {class(lemon_aide, assistant), inact_verb(eat)}>
```

Note that if the SAD generator had been given a different sequence of lexemes from which to construct the SADs (e.g. from a different schema) it would have constructed a different set of SADs. For example, given the sequence of lexemes [aide, lemon_ade] with which to construct the SADs, the SAD generator might come up with the following set of SADs for 'lemon_aide':

```
<lemon_aide: {class(lemon_aide, drink), specifier(lemon_aide,helpful)},
  {class(lemon_aide, drink), act_verb(lemon_aide, help)}>
```

Each of these adequate descriptions contains the same relations as a 'normal' set of adequate descriptions (for this lexicon) would; that is, the USADs for the lexicon constrain the well–formedness of the output of the generator. So, the task of the SAD generator is to derive appropriate USAD relations from the lexical entries for the lexemes in its input lexeme sequence. It does this using *translation rules.*

A translation rule requires certain lexical relationships to be in the entries for the lexemes in the sequence it is given, and it specifies how the arguments in those relationships can instantiate the relationships in a USAD. So, a translation rule contains two types of information: which relationships to look for in the entries of the lexemes it is given, and how the arguments in those relationships are to be used to 'fill in' a USAD.

For example, one such rule is:

```
[Described lexeme: NP
 Lexeme sequence: [X,Y]
 In entries:
        adjective(X, A)
        synonym(Y, B)
 SAD: {class(NP,B), specifier(NP, A)}]
```

In our notation, a SAD rule opens with a left square bracket, followed by the text "Described lexeme:" and a variable which serves as a 'place holder' for the lexeme that the constructed SAD is to describe. This is followed by the text "Lexeme sequence:", then a sequence of lexemes. Then there is the text "In entries:", which is followed by a set of lexical relationships that must be present in the entries for the lexemes in the lexeme sequence. This is followed by the text "SAD:", which is followed by a USAD and a right square bracket. As the variables in the translation rule are instantiated (i.e. when the lexical constraints are checked), this USAD instantiates to a SAD. This SAD is then one of the SADs output by the SAD generator for this sequence of lexemes.

The rule given above says that, given a sequence of lexemes [X,Y], look in the lexicon for an adjective A which describes X (i.e. a lexeme which satisfies the relationship adjective(X, A)), and a synonym B for Y (i.e. a lexeme which satisfies the relationship synonym(Y, B)). The rule then constructs a SAD for NP containing the relationships class(NP,B) and specifier(NP,A); in other words, it asserts that NP is a type of B, specified with the adjective A.

The above rule could be instantiated as follows:

```
[Described lexeme: NP
 Lexeme sequence: [lemon,aide]
 In entries:
        adjective(lemon, sour)
        synonym(aide, assistant)
 SAD: {class(NP, assistant}, specifier(NP, sour)}]
```

Because the relationships adjective(lemon, sour) and synonym(aide, assistant) were in the lexicon, this rule was able to construct a SAD which is, roughly, 'an assistant who is sour'. Other translation rules would construct other SADs from the same entries. Please see appendix E for a complete list of the rules used in JAPE.

Note that the translation rule is agnostic about what lexeme it is describing. The variable in the 'Described lexeme' slot is simply a placeholder. The translation rule does not know what it is describing; it only knows from which entries to construct a description. This is important for our purposes, since the schemata, which are not constrained to be 'plausible', might assert that the constructed SAD describes something entirely unrelated to the sequence of lexemes given to the SAD generator.

Also note that the resulting SADs contain the same relations as the USADs for that dictionary. The USADs are not used explicitly; however, they do constrain the well-formedness of the rules in the generator. A rule must result in a SAD which contains the relations in a USAD. This is for two reasons: the USADs define, at an abstract level, what is considered to be an adequate description (using relationships as defined for that lexicon); also, the *templates*, which are designed to translate SADs into natural language, correspond to particular USADs, so if a generated SAD is to be used with the templates, then it should contain the same relations as one of the USADs.

The SAD generator is not humour specific; its function is to construct a plausible, small, adequate description of some word or phrase, given lexical information about the words that make it up. This is necessary for the construction of jokes, which use descriptions to evoke the necessary concepts in the mind of the audience; however, it is not specific to jokes.

3.6.2 Heads and modifiers

The SAD generator is given a sequence of lexemes by a schema, to mark the entries from which it is to construct a description. It is given a sequence, rather than a set, so that it can use the last lexeme in the sequence as a source of information about the *head* of the constructed text, and the rest as information about the *modifiers*. In this section, we discuss what is meant by 'head' and 'modifier' in this context.

First, we will examine what it means to be the head of a compound nominal which is genuine (i.e. sensible) but is not represented explicitly in the lexicon — for example, "yellow car" — when constructing SADs. A genuine compound nominal is made up of a head, and modifiers. In our example, the head is the noun "car", and the modifier is the adjective "yellow".

"Yellow car" is certainly a describable concept, so it should be possible to construct a SAD for it. However, since "yellow car" doesn't have an explicit entry in the lexicon (not being a common enough concept), we can't just select relationships specified in a USAD from its entry, as we would for a compound nominal which was in the lexicon. Instead, we have to look at the entries for "yellow" and "car".

Say the entry for "yellow" contained the relationship synonym(yellow, citron), and the entry for "car" contained the relationship synonym(car, automobile). A sensible SAD for "yellow car" might be {specifier(yellow_car, citron), class(yellow_car, automobile)}

— in other words, it might be sensible to describe a yellow car as "a citron automobile".

On the other hand, not all combinations of relationships from the entries for the lexemes that make up the phrase would be appropriate. For example, {synonym(yellow_car, citron), synonym(yellow_car, automobile)} would *not* be a sensible SAD, because "yellow car" is not a synonym for "automobile", nor is it a synonym for "citron". Indiscriminately selecting relationships from the entries for the composite lexemes of the compound nominal will not work. A yellow car is a type of car (a yellow one), not a synonym for car. Likewise, a yellow car is something that is yellow (a car), not a colour. There must be some kind of transformation from the relationships in the entries for the composite lexemes of a compound nominal, and the relationships in a SAD for that compound nominal.

The entry for "car" might also contain the relationship inact_verb(car, drive), and the entry for "yellow" might contain the relationship class(yellow, colour). However, {class(yellow_car, colour), inact_verb(yellow_car, drive)} (i.e. "a colour you can drive") would not make a sensible SAD for "yellow car", simply because a "yellow car" is a type of car, not a type of colour.

For the above reasons, the rules for constructing a (sensible) noun phrase's SAD from its constituent lexemes should distinguish between the head of that phrase, and its modifiers. Some relationships in the resulting SAD should contain information from the entry for the head of the phrase, and some should contain information from the entry(ies) for the modifier(s) of the phrase. For example, the relationship 'class' in a SAD should contain information from the entry from the head of the phrase, so that a "yellow car" is described as a type of car, not a type of colour. Some relationships in the SAD could come from either entry — the 'adjective' relation, for example (a yellow car is both citron and four-wheeled).

Another constraint on this task of constructing a SAD for a sensible compound nominal is that some information must come from the entry for the head of the phrase, and some must come from the modifiers. It is not adequate to describe a yellow car as "a four-wheeled vehicle", since the information about its being yellow has been completely left out.

These considerations result in constraints on the well-formedness of a SAD generator's rules:

1. The resulting SAD must contain information from the entries for the head of the phrase, and from the entries for each of its modifiers.
2. The relationships in the resulting SAD must correspond to those in a USAD.
3. The information in the entry for the head of the phrase is treated differently from that in the entry for the modifiers, in that some relationships (e.g. 'class') in the SAD should contain information taken from the entry for the head, and some relationships (e.g. 'specifier') in the SAD should contain information taken from the entries for the modifiers. Some relationships (e.g. 'adjective') in the SAD can take information from either source.

These constraints are not humour-specific. The SADs for nonsensical compound nominals to be used in jokes are constructed the same way as those for sensible compound nominals which are not in the dictionary. The only difference is that the 'head' and the 'modifiers' of a nonsense compound nominal are specified by the schemata, not deduced from the syntax of the phrase, as they would be with a sensible compound nominal.

3.7 Templates

Templates transform a set of relations (e.g. from a schema) into a suitable surface form for a punning riddle. A template *could* be very complex; however, since our chosen domain is question-answer punning riddles, the templates need only be able to construct question-answer pairs. Although we have selected forms suitable for question-answer punning riddles, these templates are not humour-specific; that is, they can also be used to construct non-joke question answer pairs, given a set of 'sensible' relations.

A template consists of variables which are to be instantiated to text segments and lexemes, constraints on the relations the template can be given, and a sentence form into which the words and lexemes the template is given can be slotted. A sentence form is a piece of text containing some gaps, and grammatical functions for filling those gaps. For example, here is one template:

```
{Relations: describes(NPWF,
              {class(Lex, Class), specifier(Lex, Spec)})
 SF: What do you call np([Spec, Class])? det(NPWF) NPWF.}
```

The constraints on this template could match with the set of relations:

```
{describes(``low-comotive'', {class(Lex, train), specifier(Lex,
depressed)})}
```

producing the instantiated template:

```
{Relations: describes(``low-comotive'', {class(Lex, train),
    specifier(Lex, depressed)})
 SF: What do you call np([depressed, train])?
     det(``low-comotive'') ``low-comotive''.}
```

The functions np/1 and det/1 use a simple grammar to construct, respectively, a syntactically correct compound nominal from a sequence of lexemes, and a determiner appropriate to a word or phrase. The grammar uses only syntactic and spelling information about the words it is given.

Please see appendix C for a full list of the templates used in the model.

3.7.1 Sentence forms

A sentence form, as mentioned above, is a piece of text containing some gaps, and grammatical functions for filling those gaps. Sentence forms are not humour-specific, as they can be used to construct sensible question-answer pairs. However, the sentence forms used in this model were chosen for their suitability for our chosen genre, question-answer riddles. Jokes in these forms are very common in joke books such as [Webb, 1978] and [Ertner, 1993]. For example, jokes of the form:

What do you get when you cross np(NP1) with np(NP2)? *np(NP3).*

are common in joke books. However, they can still be used to produce non-joke question answer pairs. For example:

What do you get when you cross a horse with a donkey? *A mule.*

Standard question-answer sentence forms, such as the above, are used here for convenience only. A template which used sophisticated natural language techniques to generate question-answer pairs from the relations specified by a schema would still produce well-formed punning riddles. For a list of the sentence forms used in JAPE, see appendix D.

3.8 Conclusion

In this chapter, we have developed a simple model for question-answer punning riddles. These punning riddles use juxtaposition, substitution or comparison to exploit low-level linguistic ambiguities to humorous effect.

A punning riddle can be seen as the result of choosing a pun schema, instantiating that schema, choosing an appropriate template, and using that template to generate the surface form of the riddle. To do this, a system must have access to a lexicon, which is a collection of phonological, semantic, and lexical information about a large number of words and phrases. It must also have access to, or be able to generate, small descriptions of both lexicalized and non-lexicalized items. Of the tools used to construct a punning riddle, only the schemata are humour-specific.

Chapter 4

Implementation

4.1 Introduction

In chapter 3, we developed a model for question-answer punning riddles. In this chapter, we describe how that model is implemented in JAPE (Joke Analysis and Production Engine), and issues relating to that implementation. For documentation on how to use JAPE, please see appendix G.

As described in chapter 3, there are three common mechanisms in riddles that use low–level linguistic ambiguity: juxtaposition, substitution, and comparison. These mechanisms construct a non-lexicalized word or phrase, and its description, so that a question about the word or phrase, with the word or phrase as the answer, constitutes a question-answer punning riddle.

In our model, a *schema* (see section 3.5) constrains relations between lexical items, and asserts relations between non-lexical (constructed) items and *small adequate descriptions* (SADs). The SADs themselves are constructed by the *SAD generator* (see section 3.6). A *template* (see section 3.7) translates the relations asserted in an instantiated schema into the surface form of the riddle. Only the schemata are humour–dependent; the SAD generator and the templates can also generate sensible, non-humorous text.

The model shown in chapter 3 attempts to describe all question-answer punning riddles which use spelling ambiguity (i.e. two text segments sound the same but are written differently) or word-sense ambiguity (a text segment has two different meanings). This implementation, however, is slightly narrower in scope. It can generate riddles that:

- as a punning mechanism, use typical subtypes of juxtaposition, substitution or comparison;
- use the constructed word or phrase in the punchline (as opposed to the question) part of the riddle.

These restrictions are chosen largely to reduce the number of schemata and templates required, so that the important factors in JAPE's performance stand out. Also, the computer-readable lexical resources available limit the types of mechanism that JAPE can use (see section 4.3.3). Nonetheless, most of the jokes in our corpus of joke books still fall within this range.

4.2 Flow of processing

JAPE, the implementation of our model, attempts to construct a question–answer riddle from humour–independent lexical information. It has several distinct knowledge bases with which to accomplish this task:

- a lexicon, which contains humour–independent semantic, syntactic, and phonological information about the words and common compound nominals (see section 4.3);
- a set of thirteen schemata[1] (see appendix J);
- a set of 21 templates suitable for the current lexical resources (see appendix C); and
- a SAD (small adequate description) generator, containing ten rules suitable for use with our current lexical resources (see appendix E).

At the top level, JAPE chooses a schema (or it could be specified by the user), and instantiates it. Instantiating a schema involves generating small adequate descriptions, which is done by the SAD generator. The relations asserted in the instantiated schema

[1] Nine of which were used in the evaluation in chapter 5.

are then passed to a template, which generates a suitable surface form for the riddle. All of these stages require information from the lexicon.

It is not inherent in our model that these stages of joke production happen in any particular order, although some choices do constrain others.

4.3 The lexicon

In order to produce punning riddles, JAPE must have semantic, syntactic and phonological information about the words it uses. The more information it has, the greater the quantity and quality of jokes it can produce. Unfortunately, most computer-readable lexical resources are limited, either in the depth of the information they provide, or in the number of words they cover.

We have taken two approaches to solving this problem. The first, used in an early pilot version of JAPE (JAPE-1), was to use a small, hand–built lexicon, constructed by volunteers, in combination with a list of homonyms [Townsend and Antworth, 1993] for the phonological information. The second approach was to combine large, publicly available lexicons: WordNet [Miller et al., 1990], the British English Example Pronunciation dictionary [Robinson, 1996], a list of homonyms [Townsend and Antworth, 1993], and the MRC psycholinguistic database [Wilson, 1987]. This approach was used in the final version of JAPE (JAPE-2). Both approaches work, although both have their drawbacks. They are discussed below.

For the sake of this discussion, a *lexeme* is a symbol unique to one semantic interpretation of a word. Each entry in the lexicon has one, and only one, corresponding lexeme, which in turn is associated with a near–surface form. The near–surface form of a word can have several different associated lexemes (and thus several lexicon entries) or it may have none at all.

A *near–surface form* is a piece of text (a word, phrase, sentence, or complete riddle) in grammatical, understandable English; however, it is not 'pretty printed' (i.e. it doesn't necessarily have capitals at the beginning of sentences, etc). In this implementation, a near–surface form is a Prolog list of words. This is for programming convenience only — once a riddle has been fully generated, it is changed into surface form by the program.

4.3.1 The hand-built lexicon

The lexicon for JAPE-1 was constructed by volunteers, and contains only semantic and syntactic information (see section 5.2.4 for a description of how the volunteers defined the words). A list of homophones was the only phonological information available to JAPE-1, the prototype system. Although this lexicon was designed specifically for JAPE, the information contained in it is general and neutral — the joke–generating power lies elsewhere in the program, particularly in the schemata.

This is a simple slot-based lexicon. Each lexeme can be considered to be a node in a network, linked to other lexemes in the network via the semantic slots in its lexical entry. The values in these semantic slots should be other lexemes with entries in the lexicon. Syntactic slots, on the other hand, contain syntactic information, not lexemes. Please see table 4.1 and table 4.2 for more details of the available syntactic and semantic slots.

Although the values in the semantic slots *should* be other lexemes, in some cases (HAS, ACT_OBJ, LOCATION, and USED_TO_OBJ) a semantic slot takes a near–surface form instead. This is because we are interested in JAPE as an implementation of a model of riddles, rather than as a generator of syntactically–complex sentences. In order to avoid complex (but uninteresting) syntactic generation, the values in some semantic slots are chunks of text (i.e. words put together grammatically in near–surface form) instead of lexemes, so that they can be put directly into a template without further syntactic manipulation. For example, the entry for the lexeme **lion** has, as its LOCATION slot value, the near–surface form of "in the jungle", rather than the lexeme **jungle**.

One word can have several associated lexemes, and thus several entries. For example, the word "jumper" has two entries, one for each meaning:

> lexeme = **jumper_1**
> category = noun
> written_form = "jumper"

Syntactic Slots	*Used With*	*Allowed Values*
CATEGORY	all entries	np, noun, adj, verb
WRITTEN_FORM	all entries	The near-surface form of the lexeme. For nouns, this is taken by convention to be the singular form, and for verbs, the infinitive.
VOWEL_START	np, noun, adj	yes or no (does the near-surface form of the lexeme start with a vowel?)
THIRD	verb	The near-surface, third-person singular form of the verb.
COMP_LEX	np	A list of the lexemes that make up the noun phrase.
COUNTABLE	np, noun	yes or no (is the noun or np countable?)

Table 4.1: Hand-built lexicon syntactic slots

Semantic Slots	*Used With*	*Allowed Values*
CLASS	np, noun	The immediate superclass of the lexeme (e.g. for **lemon**, **fruit**)
SPECIFIER	np, noun	A lexeme that, when used to qualify the class lexeme, defines the entered lexeme reasonably precisely. (e.g. for **lemon**, **citrus**)
IS	np, noun	A lexeme that typically describes the entered lexeme (e.g. for **lemon**, **sour**).
HAS	np, noun	Part(s) of the thing to which the entered lexeme refers, in near-surface form. Should fill "It has ___.". (e.g. for **lemon**, "pips").
ACT_VERB	np, noun	A verb lexeme. Something the thing typically does. (e.g. for **chef**, **cook**)
ACT_OBJ	np, noun	The near-surface form of the object of the ACT_VERB value. (e.g. for **chef**, "food")
INACT_VERB	np, noun	A verb lexeme. Something you typically do to the thing. (e.g. for **horse**, **ride**).
LOCATION	np, noun	The near-surface form of its typical location. (e.g. for **horse**, "in a pasture")
USED_TO	np, noun	A verb lexeme. Something the thing is typically used to do. (e.g. for **spatula**, **flip**)
USED_TO_OBJ	np, noun	The near-surface form of the object of the USED_TO value. (e.g. for **spatula**, "pancakes")
SYNONYM	np, noun, adj	A lexeme of the same category as the entered lexeme, which has a very similar entry — in particular, a lexeme's synonym's synonym is itself. (e.g. for **pillow**, **cushion**)
DESCRIBES_ALL	noun, adj	A lexeme which refers to a thing or class of things which can (almost) always be described by the entered lexeme. (e.g. for **slimy**, **worm**)

Table 4.2: Hand-built lexicon semantic slots

vowel_start = no

countable = yes

class = **clothing**

specifier = **warm**

synonym = **sweater**

lexeme = **jumper_2**

category = noun

written_form = "jumper"

vowel_start = no

countable = yes

describes_all = **kangaroo**

act_verb = **leap**

Advantages and disadvantages of the hand-built lexicon

The main advantage of the hand-built lexicon was that, although it had no humour-related content, it was designed to suit JAPE's needs for riddle generation. Words were selected for entry according to their potential for punning (i.e. homophonous words, compound nominals etc.). There was a wide range of semantic slots to choose from, and volunteers had the option of adding appropriate text strings to a definition.

However, using the hand-built lexicon had two major drawbacks: size and consistency. It was simply not big enough to generate large numbers of output texts. Moreover, the volunteers defining the words were not lexicologists, and so could be inconsistent in their definitions. In addition, the very fact that the lexicon was hand-built, even by volunteers, raises suspicions of humour–specific information being encoded in the lexicon.

4.3.2 The homophone base

The homophone base is simply a list of homonym and alternate–meaning pairs. Homonyms are words that are phonologically identical, but have different spellings. Homonyms should also have different meanings; that is, spelling variants, such as "humor" and "humour", are not considered to be homonyms. The more general term for a pair of texts that are phonologically identical, and may or may not have different spellings, is *homophone* pair.

An alternate–meaning pair, on the other hand, is a pair of lexemes that have identical near–surface forms, but different semantic entries. For example, the lexeme SOLE_1, which refers to a kind of fish, and SOLE_2, an adjective synonymous with "only", are alternate meanings; however, SHOWER_1, a light rain, and SHOWER_2, a bathroom device, are not, since they both refer to water falling from above. In JAPE-1, the entries for the two lexemes had to be *completely* different for them to be an alternate–meaning pair.

JAPE-1's homonyms are from a list [Townsend and Antworth, 1993] of homonyms in American English, which has been shortened considerably for our purposes. Removed from the list were:

- pairs including a proper noun (e.g. "Cain" and "cane")
- pairs including an obscure word (e.g. "buccal" and "buckle")
- pairs which are not homophonous in British English (e.g. "balm" and "bomb")
- pairs including words which are neither adjectives nor nouns
- pairs including abstract nouns
- pairs whose meanings are often confused (e.g. "acclamation" and "acclimation")

JAPE-1, using the hand-built lexicon, treats homonym pairs and alternate-meaning pairs as equivalent, since they seem to play the same role in simple word-word substitution riddles. However, this is not the case in JAPE-2, which uses more complex mechanisms to generate jokes.

4.3.3 WordNet

WordNet is a large on-line lexicon, the organisation of which was inspired by various psycholinguistic theories. At the time of the 1990 documentation [Miller et al., 1990], it had 95,600 different word forms, organised into 70,100 word meanings. It contains only nouns, verbs, adjectives and adverbs, and organises each category in a significantly different way. WordNet's data on verbs and adverbs is not useful to JAPE (see section 4.3.3), so the remainder of this discussion will concentrate on nouns, compound nominals, and adjectives.

WordNet distinguishes between word forms and word meanings. Word forms are organised into synonym sets (synsets), which serve as word meanings. A synset is a set of words which are roughly synonymous, in that they can be substituted for each other in the same context; for example, {board, plank} and {board, committee} are two synsets containing the word "board". If no appropriate synonym is available, sometimes a synset will contain a short gloss, e.g. {board, (a person's meals, provided regularly for money)}. Most relations in WordNet are between synsets, but some are between individual words (i.e. lexemes).

The combination of a synset number (which uniquely identifies a synset) and a word number (which uniquely identifies a word within a synset) constitutes a WordNet *lexeme* (i.e. a unique identifier for a particular sense of particular word).[2]

The relations relevant to nouns and adjectives in the current version of WordNet are:

synonyms: $syn(X, Y)$ if X and Y are both members of a single synset. A relation between words.

antonyms: $ant(X, Y)$ if X and Y are opposites. Antonyms are given for nouns if appropriate, but are more important organisationally for adjectives and adverbs. A relation between words.

hyponyms: $hyp(X, Y)$ if X is a kind of Y. A relation between synsets.

meronyms: $mer(X, Y)$ if X is a part of Y. A relation between synsets.

[2] For the sake of this discussion, a lexeme is represented as a boldface **word**, rather than as a pair of numbers.

Nouns and compound nominals

WordNet's nouns are stored in an IS-A hierarchy with inheritance of distinguishing features. There are three main kinds of distinguishing feature: attributes (modification, as by an adjective), parts (meronymy), and functions (predication, as by a verb). For example, a canary is a bird (hypernym) that is small and colourful (attributes), has a beak and wings (parts), and sings and flies (functions). Only meronymy is currently implemented in WordNet. This is a great pity, as JAPE's output would be much richer if JAPE had access to attributes and functions of words. Some antonymy information is also included, but is not used to organise the nouns.

Rather than being built into a single hierarchy (with, say, entity at the top), WordNet's nouns are organised under twenty-five 'unique beginners' such as act, action, activity, artifact, location, place, motive, etc. Most synsets are somewhere between 4 and 7 levels deep in the hierarchy (e.g. roadster ISA car ISA motor vehicle ISA vehicle ISA conveyance ISA artifact). In constructing this hierarchy, WordNet creators took care to avoid words being their own hypernyms.

Because the noun synsets are organised into an IS-A hierarchy, each synset (except the unique beginners) has a mother (i.e. its hypernym synset), and could have sisters (i.e. other synsets sharing its hypernym).

Three types of meronymy are implemented in WordNet: component-object (e.g. branch/tree), member-collection (e.g. tree/forest), and stuff-object (e.g. wood/tree). Other kinds of meronymy are discussed in the WordNet papers, as are attributes and functions for nouns, but they are not implemented. Antonymy (e.g. victory/defeat, man/woman) is occasionally included, but not consistently.

Adjectives

WordNet divides adjectives into two types: descriptive and relational. Reference-modifying adjectives ("former", "occasional", "alleged" etc.) and colour adjectives are treated as special cases.

Descriptive adjectives ascribe a value of an attribute to a noun (e.g. "low" and "high" are values for the attribute HEIGHT). Descriptive adjectives are related to each other by the *antonymy* relation, either directly ("low" and "high") or indirectly ("moist" has the supertype "wet", which is an antonym of "dry"), and by degree.

Relational adjectives play a role similar to that of a modifying noun, relating the noun they are modifying to some other noun. For example, in the phrase "dental hygiene", "dental" relates "hygiene" to "teeth". Unlike descriptive adjectives, relational adjectives are not scalable. In WordNet, relational adjectives are connected to the noun to which they are relevant, e.g. the entry for "stellar" includes a pointer to "star".

WordNet and JAPE

JAPE relies on there being certain types of lexical relations available in its lexicon. In JAPE-1's hand-built lexicon, there was a range of semantic slots available (see table 4.2): synonym, class ('hyponym' in WordNet), specifying adjective, adjective ('attribute' in WordNet), has ('meronym' in WordNet), active verb (subsumed by 'function' in WordNet), active object, inactive verb (subsumed by 'function' in WordNet), location, used-to verb (subsumed by 'function' in WordNet), used-to object, and describes-all. Although the WordNet papers [Miller et al., 1990] discuss several of these relations in some detail, only synonym, class, and meronym are implemented.

The limited number of WordNet relations limits the types of small adequate description (SAD) JAPE can construct. The number of SADs is restricted in two ways: the number of USADs for the lexicon is smaller (that is, there are fewer ways to describe genuine WordNet entries than there are ways to describe entries in the hand-built lexicon); and the information available to the SAD generator is restricted, so that the preconditions to its rules are met less often, so that fewer SADs can be generated. It is the latter restriction that is relevant to JAPE's ability to generate jokes.

In particular, WordNet, unlike the hand-built lexicon, contains no semantic links between nouns and verbs, since the relation 'function' is not currently implemented. This is a great restriction on JAPE, as some of the schemata require this information, and cannot be instantiated without it. In an attempt to make up for this lack, we constructed a small database (approximately 1000) of noun–verb relations, similar to act_verb and inact_verb of the hand–built lexicon (see table 4.2). These were derived from a children's dictionary [Sansome, 1982], as follows:

- Every time a noun was defined, if that definition contained a verb as its main part, the noun lexeme and the verb lexeme were entered into the database, using act_verb if the verb was a typical action of the noun (e.g. act_verb(**cat, meow**)), and using inact_verb if the verb was an action typically done to the noun inact_verb(**food, eat**).
- The list of relations thus created was checked by an impartial volunteer, who removed any relations she thought to be untypical.
- If a verb lexeme from a noun-verb relation had any synonyms in WordNet, these lexemes were also added to the database, in the same relation to the noun. For example, if the relation inact_verb(**drink, water**) was already in the database, the relation inact_verb(**imbibe, water**) would be added to the database.
- If a noun lexeme from a noun-verb relation had any synonyms or descendants in WordNet's IS-A hierarchy, those lexemes were also added to the database, in the same relation to the same verb. For example, if the relation act_verb(**dog, bark**) was already in the database, the relation act_verb(**beagle, bark**) would be added to the database.

This labour–intensive and not very principled methodology added approximately 1000 noun–verb relations to JAPE's lexicon. Even with these additions, JAPE was only able to generate nine riddles that use noun–verb links, due to other lexical constraints.

Another problem with WordNet is that it contains a vast number of words, many of which are obscure, abstract, or otherwise unsuitable for children's riddles. WordNet also contains a lot of slang, although it does not mark it as such. Without any way of filtering out these words, JAPE often generates obtuse or incomprehensible puns. For example:

> What do you get when you cross a psyche with a linguistic unit? *A soul-fa syllable!*

One solution to this problem is considered in the next section.

4.3.4 The MRC psycholinguistic database

Using only WordNet (section 4.3.3), BEEP (section 4.3.5), and the homonym list [Townsend and Antworth, 1993] JAPE can generate large numbers of punning riddles. There is one problem, however. JAPE is punning without regard to the comprehensibility of the words it is using. For this reason, in its output are puns on words that even most adults would not recognise or understand. The grammar of the output texts is regular, simple, and traditional for this genre, due to the use of templates; it is the vocabulary that is the problem.

Some filtering of output texts according to readability is therefore required. In order to filter the riddles, we need some sort of 'comprehensibility' score for the words in WordNet. The MRC Psycholinguistic Database [Wilson, 1987] is a possible source of such data; however, it uses a combination of different scoring systems, which must be combined into a useful score.

The MRC psycholinguistic database is a compilation of psycholinguistic data from a variety of sources. Much of the data is only available for a subset of the words in the database; for example, only 3503 of the total 150837 words have 'age of acquisition' scores. The database contains scores for eleven psycholinguistic measures, as well as phonological and syntactic data (see table 4.3).

Of these, the most relevant to the problem of filtering words for children are familiarity, concreteness, imagery, and age of acquisition. Unfortunately, the total occurrences quoted in the documentation includes some redundancy — words with multiple pronunciations, for example, are counted several times, even if only one familiarity (for example) score is given for that word. 4924 unique words have at least one of these four measures in the database.

Here are the four measures — familiarity, concreteness, imagery and age of acquisition — as described in the MRC manual [Wilson, 1987]:

"FAM: This stands for 'printed familiarity'. The FAM values were derived from

Measure	*Occurrences*
Number of letters in the word	150837
Number of phonemes in the word	38438
Number of syllables in the word	89402
Kucera and Francis written frequency	29778
Kucera and Francis number of categories	29778
Kucera and Francis number of samples	29778
Thorndike-Lorge frequency	25308
Brown verbal frequency	14529
Familiarity	9392 (unique 4924)
Concreteness	8228 (unique 4295)
Imagery	9240 (unique 4829)
Mean Colerado Meaningfulness	5450
Mean Pavio Meaningfulness	1504
Age of Acquisition	3503 (unique 1904)
Type	44976
Part of Speech	150769
PD Part of Speech	38390
Alphasyllable	15938
Status	89550
Variant Phoneme	1445
Written Capitalised	4585
Irregular Plural	23111
the actual word	150837
Phonetic Transcription	38420
Edited Phonetic Transcription	136982
Stress Pattern	38390

Table 4.3: Fields available in the MRC psycholinguistic database

merging three sets of familiarity norms: Pavio (unpublished), Toglia and Battig [Toglia and Battig, 1978] and Gilhooly and Logie [Gilhooly and Logie, 1980]. The method by which these three sets of norms were merged is described in detail in Appendix 2 of the MRC Psycholinguistic Database User Manual [Coltheart, 1981]. This method may not meet with everyone's approval. FAM values lie in the range 100 to 700 with the maximum entry of 657, a mean of 488 and a standard deviation of 99: note that they are integer values (in the original norms the equivalent range was 1.00 to 7.00). [Higher values indicate more familiar words.]

CONC: This is concreteness, and it too is derived from a merging of the Pavio, Colerado, and Gilhooly-Logie norms: details of merging are given in Appendix 2 of the MRC psycholinguistic database User Manual [Coltheart, 1981]. CONC values are integer, in the range 100 to 700 (min: 158; max 670; mean 438; s.d. 120). [Higher values indicate more concrete words.]

IMAG: This is imageability, derived from merging the three sets of norms referred to above, and having values in the range 100 to 700 (min 129; max 669; mean 450; s.d. 108). [Higher values indicate more imageable words.]

AOA: This is age of acquisition from the norms of Gilhooly and Logie [Gilhooly and Logie, 1980], multiplied by 100 to produce a range from 100 to 700 (min 125; max 697; mean 405; s.d. 120)." [Higher values indicate words learned at a later age.]

We expect that, for the familiarity, concreteness and imageability measures, high scores are desirable for our purposes; whereas, for the age of acquisition score, low scores are preferable. That is, we expect that jokes containing familiar, concrete and imageable words with a low age of acquisition will be easier to understand. This expectation is confirmed by our evaluation (see section 5.3.6).

One problem with the MRC is that it does not contain scores for any of these measures for compound nominals. We can guess at the scores for a compound nominal by combining the scores for the words that make it up; however, this is not very accurate. The phrase "rabbit punch" ("a short chopping blow to the back of the neck", according to WordNet) is probably much less familiar than the word "rabbit" or the word "punch".

As an approximation, we will assume that the score on a particular measure for a multi-word text is the *worst* score (i.e. lowest for FAM, CONC and IMAG, highest for AOA) on that measure for any word in the text. This is based on the idea that the comprehensibility of a text is limited by the least comprehensible word in that text. When scoring punning riddles, it is important to consider *all* the words involved in the pun (i.e. that appear in the instantiated schema), even if they do not appear in the text, since they also affect the comprehensibility of the riddle.

For example, consider the JAPE generated riddle:

> What do you call a fashion near-miss? *A clothes call.*

The key words (i.e. non-template words) in this text are: fashion, near, miss, clothes, close and call. Although "close" does not appear in the riddle text, it was the word replaced by "clothes", and must be understood for the joke to make sense. The MRC scores for these words are:

Word	*FAM*	*CONC*	*IMAG*	*AOA*
fashion	548	356	474	467
near	582	337	408	n/a
miss	586	372	447	n/a
clothes	652	600	629	194
call	559	389	424	225
close	587	391	420	283
WORST SCORE	548	337	408	n/a

Note that the worst score for the age of acquisition measure is not available. This is because the MRC does not have a score for that measure for two of the words in the joke, so the worst score for this text cannot be determined.

The relation between these scores to the comprehensibility of a riddle cannot be decided *a priori*, but must be shown by experiment. The final evaluation of JAPE (see section 5.3.6) showed that familiarity, concreteness and imageability scores for a riddle do correlate weakly with the perceived 'jokiness' of the riddle. This suggests that filtering out the JAPE output texts with low scores on these measures would improve the overall performance of JAPE. In the version of JAPE evaluated in the final experiment (JAPE-2), no automatic filtering was done by the program. Instead, texts which actually *had* scores for these measures were preferred. If a schema generated more texts than required for the evaluation, those with *high* familiarity, concreteness and imageability were preferred. See section 5.3.3.

4.3.5 The British English Example Pronunciation dictionary

The prototype version of JAPE, JAPE-1, uses only one mechanism to generate puns: word–word homophone substitution into noun phrases. To do this, the only phonological information it requires is a list of homonym pairs. JAPE-2, however, with its more complex mechanisms, requires more detailed phonological information about the words it uses.

The British English Example Pronunciation dictionary [Robinson, 1996], or BEEP, contains phonemic transcriptions, using the ARPAbet standard, of 150 000 words. See appendix A for the set of phonemes used in BEEP. Although BEEP was not originally implemented in Prolog, it was easily translated into this form, with the phonemic transcriptions represented as a list of phonemes.

In order to provide adequate information for all JAPE's schemata (see section 4.4), the BEEP-JAPE interface must be able to:

- give pairs of homonyms;
- segment words into their beginnings and endings, so that alliterative pairs, rhyming pairs, and spoonerisms can be found; and
- compare the phonemic transcriptions of word segments with their graphemic transcription (i.e. spelling).

The first task is straightforward. A pair of homonyms is a pair of words that share the same phonemic transcription, but do not have the same spelling. In order to eliminate alternate spellings of a word, which are homophones but not homonyms, homonyms must also not belong to the same WordNet synset (see section 4.3.3) — that is, homonyms must not be synonyms.

The second task is slightly more difficult. If we separate all the phonemic transcriptions of the words into two segments — the *initial consonant sound* (i.e. one or more of {p,b,t,d,k,m,n,l,r,f,v,s,z,hh,w,g,ch,jh,th,dh,sh,zh,y} that precede the first vowel sound, or **nil** if the word starts with a vowel sound) and the *remainder* (i.e. the list remaining after the initial consonant sound has been removed) — then rhyming, alliteration and spoonerism can be defined as follows:

rhyming: Two words rhyme if:

- they have different initial consonant sounds, and
- they have identical remainders.

For example the words "cat" ([k,ae,t]) and "bat" ([b,ae,t]) rhyme.

alliteration: Two words alliterate if:

- they have identical non-nil initial consonant sounds, and
- they have different remainders.

For example, the words "cat" ([k,ae,t]) and ("cake" ([k,ey,k]) alliterate.

spoonerism: Four words (A, B, C and D) form two spoonerizing phrases (AB and CD) if:

- words A and C rhyme,
- words B and D rhyme,
- words A and D alliterate, and
- words B and C alliterate.

For example, the phrases "cat bake" ([k,ae,t] and [b,ey,k]) and "bat cake" ([b,ae,t] and [k,ey,k]) spoonerize.

Note that these definitions of rhyming and alliteration are overly narrow, in that they only allow words with initial consonants to rhyme or alliterate. Moreover, the above definition of rhyming requires that *all* of the remainders of the words be identical, which eliminates rhymes like "locomotion" and "commotion". These restrictive definitions were introduced for two reasons. Firstly, we do not know how phonologically similar two words have to be before they are confusable enough for the purposes of a joke, so we chose to err on the side of caution and make them as similar as possible. Secondly, it is difficult to determine *how much* of the remainders must be identical for two words to rhyme — after all, "indescribably" and "dromedary" are not usually considered to be rhymes, although they share a terminal phoneme. Again, we chose to err on the side of caution.

The third task, comparing the phonemic transcriptions of word segments with their spelling, is quite difficult. Consider the following problem. The word "migraine" has the phonemic transcription [m,iy,g,r,ey,n]. If we remove the phonemes [g,r,ey,n] from the end of the word, what is the graphemic transcription (i.e. spelling) of the word segment ([m,iy]) that remains? To do this task perfectly requires a good understanding of all the heuristics of spelling. Building a system with such an understanding is beyond the scope of this research. However, we have found that the following approach, which only compares consonants and consonant sounds, works reasonably well.

Given the graphemic transcription[3] of the word, Graph (e.g. *[m,i,g,r,a,i,n,e]*), the phonemic transcription of the word, Phons (e.g. [m,iy,g,r,ey,n]), the phonemic transcription of the word segment, SubPhons (e.g. [m,iy]), and the information that SubPhons comes at the beginning of Phons, we can find the graphemic transcription of the word segment as follows:

- Remove the vowels from Graph (e.g. *[m,i,g,r,a,i,n,e]* → *[m,g,r,n]*). Call the remaining list GraphCons.
- Using simple matching rules which match consonant sounds to possible spellings (see appendix B), find the phonemes in Phons which match GraphCons, in order (e.g. [m,g,r,n]). Call this PhonCons.
- Compare PhonCons (e.g. [m,g,r,n]) with SubPhons (e.g. [m,iy]). Find the elements of PhonCons which appear, in order, in SubPhons (e.g. [m]). Call this MatchCons.
- Find the phoneme in PhonCons immediately after[4] MatchCons (e.g. g). Find the grapheme that corresponds to this phoneme (e.g. *g*). Take everything in Graph

[3] For this section, we adopt the convention that graphemes are represented in *italics*.

[4] or before, if SubPhons comes the at end of Phons.

between the beginning and the grapheme (e.g. *[m,i]*). This is the approximate graphemic transcription of the word segment SubPhons.

This system works well, except on cases in which it would be appropriate to split up a series of vowel sounds (e.g. finding the "the" in "theory") or in which a consonant grapheme does not correspond to a consonant sound in British pronunciation (e.g. the "r" in "whisper"). It is also restricted to cases in which the sub-word comes at the beginning or end of the whole word, rather than in the middle.

So, BEEP is able to provide most of the phonological information necessary to construct punning riddles, including those that require some sub-word phonemic analysis. With it, JAPE can find homonyms, rhymes, alliteration and spoonerisms. It can also, to some extent, find appropriate spellings for non-lexicalized phonemic sub-words.

4.4 Schemata

As described in section 3.5, a schema specifies relationships between lexical items, which are explicitly represented in the lexicon, and 'constructed' concepts. It also specifies which lexicon entries the small adequate description (SAD) generator (see section 4.6) can use to construct a set of SADs for the constructed lexeme. The instantiation of a schema is the central humour-related step in JAPE's construction of a punning riddle. The thirteen schemata described in section 3.5 can be found in appendix J.

These schemata were implemented straightforwardly into the Prolog program JAPE-2.

4.4.1 Problems with JAPE's schemata

All of the schemata require lexical information in order to be instantiated. Different schemata require different kinds of information. For example, a schema that substitutes one word into another needs sub-word phonemic and graphemic information, whereas a schema that substitutes a homonym into a compound nominal does not. Also, some schemata are more constrained than others. For example, a comparison schema that uses spoonerisms needs to find a pair of spoonerizing phrases that can both be described in a way that is appropriate for the comparison mechanism; whereas a juxtaposition schema need only find a pair of homonyms. For these reasons, some of JAPE's schemata are more successful at generating jokes than others, from the lexical resources currently available (WordNet, BEEP, the MRC database, and the homophone base).

For example, the following schema generates riddles which compare spoonerizing verb pairs:

```
{Lexical preconditions:
  rhyme(WordA, WordB)
  written_form([LexA], WordA)
  written_form([LexB], WordB)
  verb(LexA)
  verb(LexB)
  rhyme(WordC, WordD)
  written_form([LexC], WordC)
  written_form([LexD], WordD)
  verb(LexC)
  verb(LexD)
  alliterate(WordA, WordC)
  alliterate(WordB, WordD)
SAD constraints:
  described_by([LexA, LexD], SAD1)
  described_by([LexB, LexC], SAD2)
Relations:
  describes_same(
        {inact_verb(Lex1, LexA), inact_verb(Lex1, LexD)}, SAD1)
  describes_same(
        {inact_verb(Lex2, LexB), inact_verb(Lex2, LexC)}, SAD2)}
```

This is the **hopchew** schema, described in section 3.5. It could be instantiated as follows:

```
{Lexical preconditions:
```

```
  rhyme(``bake'', ``break'')
  written_form([bake], ``bake'')
  written_form([break], ``break'')
  verb(bake)
  verb(break)
  rhyme(``boil'', ``broil'')
  written_form([boil], ``boil'')
  written_form([broil], ``broil'')
  verb(boil)
  verb(broil)
  alliterate(``bake'', ``boil'')
  alliterate(``break'', ``broil'')
SAD constraints:
  described_by([bake, broil], {synonym(Lex1, potato)})
  described_by([break, boil], {synonym(Lex2, egg)})
Relations:
  describes_same(
        {inact_verb(Lex1, bake), inact_verb(Lex1, broil)},
        {synonym(Lex1, potato)})
  describes_same(
        {inact_verb(Lex2, break), inact_verb(Lex2, boil)},
        {synonym(Lex2, egg)})}
```

which could generate the riddle:

> What's the difference between a potato and an egg? *One you bake and broil*[5]*, the other you break and boil.* [JAPE]

However, to do so, JAPE must have the following noun–verb relations in its lexicon: inact_verb(**potato, bake**), inact_verb(**potato, broil**), inact_verb(**egg, break**), and inact_verb(**egg, boil**). Only five riddles were generated by JAPE using this schema, due to the schema being so constrained.

[5] "Broil" is roughly synonymous with "grill" in North American usage.

One schema was so constrained that it did not generate any riddles at all. This schema generates riddles using comparison of metathesized verb phrases:

```
{Lexical preconditions:
        alternate_meaning(LexA, LexB)
        verb(LexA)
        noun(LexB)
        alternate_meaning(LexC, LexD)
        verb(LexC)
        noun(LexD)
 SAD constraints:
        described_by([LexA, LexD], SAD1)
        described_by([LexB, LexC], SAD2)
 Relations:
        describes_same({act_verb(Lex1, LexA), act_obj(Lex1, LexD)},
                       SAD1)
        describes_same({act_verb(Lex2, LexB), act_obj(Lex2, LexC)},
                       SAD2)}
```

This is the **coatshed** schema, as described in section 3.5. This could be instantiated as follows:

```
{Lexical preconditions:
        alternate_meaning(shed_1, shed_2)
        verb(shed_1)
        noun(shed_2)
        alternate_meaning(coat_1, coat_2)
        verb(coat_1)
        noun(coat_2)
 SAD constraints:
        described_by([shed_1, coat_2], {synonym(Lex1, dog)})
        described_by([shed_2, coat_1], {synonym(Lex2, painter)})
 Relations:
```

```
describe_same(
  {act_verb(Lex1, shed_1), act_obj(Lex1, coat_2)},
  {class(Lex1, dog), adj(Lex1, hairy)})
describe_same(
  {act_verb(Lex2, shed_2), act_obj(Lex2, coat_1)},
  {synonym(Lex2, painter)})}
```

Which could generate the joke:

> What's the difference between a hairy dog and a painter? *One sheds a coat and the other coats a shed.* [Webb, 1978]

However, to generate this riddle, JAPE would require the following relations in its lexicon: act_verb(**dog_2, shed_1**), inact_verb(**shed_2, coat_1**), act_verb(**painter, coat_1**), and inact_verb(**coat_2, shed_1**). These relations are unlikely to be found in any lexicon, and are not in JAPE's lexicon as it stands. Indeed, they are more world knowledge than lexical knowledge. For this reason, JAPE is unable to instantiate this schema using the information in its lexicon, and thus cannot use it to generate punning riddles.

To avoid this problem, a schema similar to the one above, but with fewer lexical constraints, was included in JAPE. This schema also compares metathesized verb phrases. The schema is as follows:

```
{Lexical preconditions:
  inact_verb(LexA, LexB)
  written_form([LexA], WordA)
  written_form([LexB], WordB)
  homonym(WordA, WordC)
  homonym(WordB, WordD)
  written_form([LexC], WordC)
  written_form([LexD], WordD)
  not(inact_verb(LexD, LexC))
```

```
SAD constraints:
Relations:
  describes_same({synonym(LexA, LexA)},{inact_verb(LexA, LexB)})
  describes_same({synonym(LexC, LexC)},{not(inact_verb(LexC, LexD))})}
```

This is the **vn** schema.

Note that no SADs are generated in the instantiation of this schema. This is because all of the key words needed are already specified in the lexical constraints. This is unusual in a pun. Note also that one of the lexical constraints is negative; that is, it specifies a relation that should *not* be found in the lexicon. This is not ideal, since we know that JAPE's lexicon is far from comprehensive, so the absence of a relation does not guarantee its falsity. Despite these flaws, this schema can generate some (weak) riddles, unlike its more constrained predecessor.

The above schema could be instantiated as follows:

```
{Lexical preconditions:
  inact_verb(sea, sail)
  written_form([sea], ``sea'')
  written_form([sail], ``sail'')
  homonym(``sea'', ``see'')
  homonym(``sail'', ``sale'')
  written_form([see], ``see'')
  written_form([sale], ``sail'')
  not(inact_verb(sale, see))
SAD constraints:
Relations:
  describes_same({synonym(sea, sea)}, {inact_verb(sea, sail)})
  describes_same({synonym(sale, sale)}, {not(inact_verb(sale, see))})}
```

This could generate the riddle:

> What's the difference between a sea and a sale? *You can sail a sea, but*

you can't see a sale. [JAPE]

This weakening of lexical constraints allows JAPE to generate riddles of this type, although it could be argued that the riddles thus generated are of poor quality. Even this less-constrained schema can be used to generate only four riddles, using JAPE's current lexical resources.

Two other schemata were weakened by JAPE's limited lexical resources. The **lotus**, **jumper**, **rhyming_lotus**, **woolly** and **elan** schemata, described in section 3.5, all contain the lexical relation component_lexemes/2, which relates a compound nominal lexeme to the lexemes of the words that make it up. Unfortunately, this information is not represented in WordNet, although it was represented in the hand-built lexicon. For example, although WordNet does contain the information that the compound nominal lexeme **grizzly_bear** is composed of two words, "grizzly" and "bear", it does not say which *senses* of the words are used in the phrase. If JAPE chooses the wrong sense, it produces what is effectively a *double* pun, by performing an extra 'substitution'. For example, WordNet contains a sense of "bear" which is "an investor with a pessimistic market outlook", and has the hypernym "investor". If JAPE decides that this sense of "bear" is meant by the second word in the phrase "grizzly bear", then the **woolly** schema could generate the text:

What do you call a gruesome investor? *A grisly bear.*

which is an unintended double pun, in that two substitutions have taken place: **grisly** for **grizzly**, and **bear_1** (the investor) for **bear_2** (the animal). Double puns are not bad in and of themselves, but our exploratory evaluation (see section 5.2) suggested that they are significantly more difficult to understand. This problem affects the **woolly** and **jumper** schemata more than the others, as the component lexemes are used to generate the SADs for these schemata. For this reason, these two schemata were not implemented in the version of JAPE evaluated in section 5.3.

Despite the problems described above, nine of the schemata described in section 3.5 were successfully implemented in JAPE. These are (with JAPE generated examples):

phonsub: This schema substitutes a word for a homophonous segment of another word. The SAD is constructed from the entries for both words. For example:

> What do you call a depressed train? A low-comotive. [JAPE]

vn: This schema negatively compares two confusable verb-object phrases, constructed by reversing the order of two pairs of homophones. For example:

> What's the difference between a sea and a sale? *You can sail a sea, but you can't see a sale.* [JAPE]

hopchew: This schema negatively compares two confusable verb-verb phrases, constructed with metathesis. For example:

> What's the difference between money and and a bottom? *One you spare and bank, the other you bare and spank.* [JAPE]

negcomp: This schema, discussed above, negatively compares two confusable compound nominals, constructed with metathesis. For example:

> What's the difference between a criminal bag and a miserable rear? *One's a bad sack and the other's a sad back.* [JAPE]

poscomp: This schema constructs a word-sense ambiguous phrase, and positively compares its two senses. For example:

> How's a nice girl like a sugary bird? *They're both sweet chicks.* [JAPE]

lotus: This schema, discussed above, substitutes a homophone for the first word in a lexicalized compound nominal. The SAD is constructed from the entries for the homophone and the noun phrase. For example:

> What do you call a murderer with fibre? *A cereal killer.* [JAPE]

rhyming-lotus: Similar to **lotus**, this schema substitutes a rhyming word for the first word in a lexicalized compound nominal. The SAD is constructed from the entries for the rhyming word and the compound nominal. For example:

What do you call a bath tour? *a tub crawl.* [JAPE]

elan: This schema substitutes a homophone for the last word in a lexicalized compound nominal. The SAD is constructed from the entries for the homophone and the compound nominal. For example:

What do you call a naked bruin[6]? *A grizzly bare.* [JAPE]

bazaar: This schema, discussed above, juxtaposes two homophones to form a compound nominal. The SAD is generated from the entries for the homophones. For example:

What do you call a strange market? *A bizarre bazaar.* [JAPE]

The success of these schemata in generating punning riddles was evaluated in the final experiment, and is discussed in section 5.3.6.

4.5 Templates

As described in section 3.7, templates transform a set of relations (e.g. from the **Relations** slot of an instantiated schema) into a suitable surface form for a punning riddle. In this implementation, a template consists of variables which are to be instantiated to text segments and lexemes, constraints on the relations the template can be given, and a sentence form into which the words and lexemes the template is given can be slotted. A sentence form is a piece of text containing some gaps, and grammatical functions for filling those gaps. The instantiation of a template, using the relations from a schema, generates the surface form of a joke.

All of the templates given in appendix C are implemented in JAPE. When JAPE is used with WordNet, rather than with the hand-built lexicon, not all of these templates are necessary, since only a more limited range of SADs can be constructed from the information contained in WordNet. In this implementation, the task of grammatically filling the gaps in a sentence form is done by a tiny Prolog grammar — and if it

[6] A type of bear, according to WordNet.

produces the wrong form for, say, an irregular verb, we allow ourselves to fix the output by hand[7].

4.6 The generation of small adequate descriptions

A small adequate description, or SAD, is a set of instantiated relations which purports to describe a particular concept, whether lexicalized or not. When generating jokes, JAPE must build SADs for non-lexicalized items, such as the words and phrases constructed by the schemata. These SADs are constructed from the entries for the lexemes specified by the schema for that purpose, using translation rules. When a translation rule is instantiated with information from the lexicon, it generates a plausible SAD for the lexemes it is given. An example of such a rule is:

```
[Described lexeme: NP
 Lexeme sequence: [X,Y]
 In entries:
        adjective(X, A)
        synonym(Y, B)
 SAD: <NP: {class(NP,B}, specifier(NP, A)}]
```

This rule says that, given a described (place-holder) lexeme NP and a sequence of lexemes [X, Y], look in the lexicon for an adjective A which describes X (i.e. a lexeme which satisfies the relationship adj(X, A)), and a synonym B for Y (i.e. a lexeme which satisfies the relationship synonym(Y, B)). The rule then constructs a SAD for NP containing the relationships class(NP, B) and specifier(NP, A); in other words, it asserts that NP is a type of B, specified with the adjective A.

When a schema is being instantiated, the SAD generator is given a a sequence of lexemes to use to build its SAD. The SAD generator then goes through the SAD rules, trying to instantiate them. A rule is instantiated as follows. First, the sequence of lexemes is matched with the 'Lexeme sequence' slot. Then, the lexical constraints are

[7] We allow ourselves this luxury because our current lexical resources do not contain the required morphological information to do the job.

satisfied — that is, the lexicon is consulted to see if the required lexical relations are present, and instantiates any variables remaining in the relations. At this point, all of the variables in the rules should be instantiated, so that the final slot, 'SAD', contains the constructed lexeme paired with a complete SAD.

These rules are not universal; they are specific to the lexicon with which they are used. See appendix F for a list of the rules that work with the hand-built lexicon, and appendix E for a list of the rules that work with WordNet. All of these rules have been successfully implemented in JAPE.

4.7 Conclusion

This chapter has described how our model of simple punning riddles, described in chapter 3, has been implemented in JAPE, a program which generates such riddles. We have also discussed some of the problems with this implementation. The most important problem is finding lexical information in enough quantity and quality to support the generation of jokes. Even using several large computer readable data sources, such as WordNet (section 4.3.3) and BEEP (section 4.3.5), some of JAPE's schemata do not have enough information to be instantiated. Nonetheless, enough of the model has been implemented to generate large quantities of output texts.

Chapter 5

Evaluation

5.1 Introduction

This chapter describes the evaluation that took place during the development of the model described in chapter 3 and the implementation of the program described in chapter 4.

Two main types of evaluation took place. The purpose of the first, an exploratory evaluation, was to look at JAPE's progress and to suggest new avenues for improvement. The purpose of the second, the confirmatory evaluation, was to analyse in detail the behaviour of the final system, and to examine its successes and failures.

5.2 Exploratory evaluation

5.2.1 Purpose

The system evaluated in this section is the prototype of JAPE, here referred to as JAPE-1. The evaluation here described took place early in the system's development. The purpose of the exploratory evaluation was twofold. Primarily, it was to point the way to improvements in the model behind the system. This information was then be used both to improve JAPE-1's design, and to guide the path of the research.

This evaluation also provided both a relatively unbiased input (in the form of lexicon entries provided by volunteers) for an early version of JAPE-1, and a rough assessment of JAPE-1's abilities. Humour is notoriously subjective; nonetheless, the value of this research would be minimal if the data could only be entered by one person, and the resulting jokes understood by only that person. It was therefore necessary to test that lexicon entries can be made by people unfamiliar with JAPE-1's workings; that the riddles produced are actually recognizable as such; and that at least some of the jokes are moderately funny. Only after this evaluation was completed could we move on with confidence to extend the model and the program.

This evaluation was *not* intended to be a rigorous examination of JAPE-1's abilities. The goal of this evaluation was instead to provide unbiased data for JAPE-1's lexicon, give an indication of how well it works, and suggest directions for improvement.

5.2.2 Differences between JAPE-1 and JAPE-2

JAPE-1 was an early prototype of JAPE, and as such had some differences from the implementation described in chapter 4.

The most important difference between JAPE-1 and JAPE-2 is that, in JAPE-1, there was no small adequate description (SAD) generator; instead, a schema was linked directly to a set of templates which would produce an appropriate surface form. As in JAPE-2, the schema would mark the lexemes from which the semantic information used in the joke could be taken; however, as there was no SAD generator, the templates themselves would select which information would be used in the final joke.

As discussed in sections 4.3 and 5.2.4, JAPE-1 used a small, hand–built lexicon. JAPE-2, however, relies primarily on WordNet [Miller et al., 1990] for its semantic and syntactic information.

Finally, in JAPE-1, some very simple post–production checks were implemented. It checked that the constructed compound nominal used in a joke was not *real* (i.e. present in the lexicon), and it checked that none of the key lexemes used to build the riddle were identical (unless required to be so by the schema). These checks are neither necessary nor useful in JAPE-2.

5.2.3 Pre-run conjectures

It was possible to make certain pre-run predictions about JAPE-1's performance which, if confirmed, could be introduced as adjustments to the model or as selection heuristics in later versions of the program. Some of the suggested heuristics would *order* the output of the program by quality; others would *filter* the output, by reducing the number of jokes produced.

All of these conjectures were made after the lexical data was gathered, but before JAPE-1 was run to produce the set of jokes to be judged (see section 5.2.4 for details of the methodology).

JAPE-1's design suggested that there were six main factors that influenced the quality of the output. These were:

- The quality and quantity of the lexical data;
- The design of the schemata;
- The design of the templates;
- The appropriateness of the template-schema pairings;
- The amount and quality of post-production filtering; and,
- The genre of the riddles generated.

How each influences the quality of the output is considered below.

The lexicon

The lexicon should store all the information about the English language that is necessary for building puns, in a form that is easily manipulated by other parts of the program.

In JAPE-1, the only phonological information in the lexicon was a list of homophone pairs [Townsend and Antworth, 1993]. The semantic information in the lexicon was provided by a group of volunteers (see section 5.2.4), producing a small, hand-built, yet humour-independent lexicon. The lexicon itself was a simple slot-based lexicon with inheritance, as described in section 4.3.

JAPE-1 created jokes by creating a non-lexicalized (i.e. not already in the lexicon) compound nominal from real words and phrases. For example, it could construct the phrase "cereal killer" by substituting "cereal" for "serial" in the lexicalized phrase "serial killer". It then 'built a meaning' for the constructed phrase by marking the entries from which the templates could select information to appear in the final joke. This information was then used to construct a question which referred to the constructed compound nominal. The question and the constructed compound nominal made up the riddle. For example:

> What do you get when you cross a breakfast food with a murderer? *A cereal killer.*

Each of these steps relies on the lexicon being clear, detailed and compatible with JAPE-1's templates and schemata. For this reason, it was anticipated that the experiment would show the following heuristics to be valuable in improving the quality of JAPE-1's output. Please keep in mind that, at this stage of development, we were not yet using the MRC psycholinguistic database (see section 4.3.4).

- Semantic information should be included in the lexicon only if it is typical of, and specific to, the word being entered. For example, dogs do sleep, but if the entry for "dog" includes that information, then JAPE-1 is likely to ask questions like "What sleeps and ...", expecting "dog" to be among the concepts evoked in the mind of the reader.

  Although volunteers were asked to provide only 'typical' information about the words they defined, it was expected that some of the definitions would be overgeneral, containing information more appropriate for a super-type of the word being defined. This was expected to produce riddle questions which do not sufficiently evoke the concepts necessary for the riddle to be understood.

- The information in the lexicon should be *common knowledge*. This applies both to the words chosen to be entered, and the entries they are given. It's no use

JAPE-1 making a joke about a horse's frog, if most people don't know that "frog" can refer to part of a horse's foot, as well as to a green amphibian.

- Jokes should use concrete words (e.g. "wooden", and "cat"), rather than abstract words (e.g. "happiness", "attitude"). This is because the constructed concept is more likely to be funny if it can be visualised.

  For example, the idea of a "toe truck" probably has more humour potential than that of an "optical allusion". This may, however, depend on the age or vocabulary of the audience.

- Jokes should avoid using very general words (e.g. "structure", "substance", "object") because such words have a huge number of possible instances. For example, although a Buddhist monk is indeed a person, the word "person," without any other information, is unlikely to bring the image of a Buddhist monk to mind — so a joke depending on that evocation would probably fail.

- Homonym pairs should not be easily confused (i.e. their distinct meanings should be common knowledge). For example, a pun based on the homonyms "aural" and "oral" would probably not work very well, as these words are often misspelled and misused.

The schemata

The schemata, as described in section 3.5 and section 4.4, were based on observed features of punning riddles, so were expected to be reasonable models of inter-word relations in puns.

JAPE-1 used only six schemata. In section 3.5, JAPE-2's schemata are described; of these, JAPE-1 had only the *lotus, elan, woolly, jumper, double_pun,* and *ginger*. As in JAPE-2, these schemata define the lexical constraints on components of a pun.

All of JAPE-1's schemata use only one mechanism: homophone substitution. Each constructs a nonsensical compound nominal by substituting one or more homophones into an existing compound nominal. It was anticipated that some of these schemata would produce a greater proportion of texts which would be classified as jokes by the judges than others.

It was anticipated that the *double_pun* schema would not provide enough information for the audience to 'solve' both homophone-substitutions, and would therefore produce incomprehensible jokes. For this reason, it was expected to be the least successful schema.

It was expected that the *woolly* and *jumper* schemata would produce slightly better jokes than the *lotus* and *elan* schemata, because the latter two rely on good, clear lexicon entries for compound nominals. Such entries are difficult to make because, in general, compound nominals describe more complex concepts than individual words, so we could expect the lexicon to be less accurate in defining compound nominals. Also, a compound nominal tends to have a very similar definition to one of the words in the phrase, leading to some muddling of concepts in the joke. For example, the phrase "grizzly bear" and the noun "grizzly" have almost identical meanings, which could cause confusion.

The *elan* schema substitutes a homophone for the second word in a compound nominal, and generates the meaning of the constructed compound nominal from the homophone and the original compound nominal. This causes some confusion when trying to 'understand' the punchline, because it is difficult to determine which is the object being modified by the first word in the punchline — the second word (i.e. the substituted homophone), or the original compound nominal (see section 5.2.3 for a discussion of word order in the punchline). For example, in the following text, which the *elan* schema generated:

> What do you use to hit a prompt? *A pool cue.*

the word "pool" is not very strongly evoked by the question. This is because the entry for "pool cue" says only that they are used to hit balls, which does not evoke "pool" specifically.

Because of this confusion, the *elan* schema was expected to be considerably less successful than the similar *lotus* schema.

The templates

The templates, as described in section 3.7 and section 4.5, were designed to produce a surface form text that is comprehensible and recognisably a joke (i.e. in some standard joke format).

In JAPE-1, there were no small adequate descriptions, or SADs (see section 3.6), to serve as the inputs to the templates. Instead, the templates used the information from the schemata, supplemented by lexical information, directly. A template itself would select, in a somewhat ad-hoc manner, which semantic information from the entries marked by the schema would appear in the final joke. This caused flaws in the 'linguistic logic' of the joke.

All of the schemata in JAPE-1 manipulated compound nominals. The resulting constructed compound nominals were not 'real' (i.e. they would not normally be lexicalized); however, the riddlee still tries to understand the constructed compound nominal as a real compound nominal. That is, the riddlee expects a two-word compound nominal to be made up of a noun preceded by an adjective (or a noun acting as an adjective) which qualifies it. For example, one would expect a "fire fighter" to be someone who combats flames, rather than someone who burns combatants.

Some of the templates did not support this expectation about a noun phrase. Instead, these templates would use the *second* word in the phrase as if it were the modifier, and the first as if it were being modified. Some templates were not expected to perform very well, for this reason.

For a more lengthy discussion of this issue, and JAPE-2's solution to the problem, see section 3.6.1.

The schemata-template pairs

In the version of JAPE (JAPE-1) tested in the exploratory evaluation, schemata and templates were paired together. It was important that the paired schemata and templates be compatible.

It was anticipated that certain schemata would work well with some templates, and not so well with others. It was difficult to predict which would be the successful pairs; however, we expected the distinction to be clear. In fact, we predicted that the elimination of certain schema-template pairings would improve the data considerably. No attempt to remove such pairs was made before evaluation, however, in the hope that the difference between successful and unsuccessful schema-template pairs would be large enough to justify the separating of bad pairs.

The post–production checking

After JAPE-1 had produced a near–surface form question–answer riddle, it would check that the riddle satisfied certain simple constraints. At the time of testing, JAPE-1 checked only that the punchline of the riddle was not a real compound nominal (i.e. that it was not in the lexicon), and that none of the key lexemes used to build the riddle were accidentally identical (i.e. they were only identical if linked by an identity link).

Genre

Unfortunately, the type of riddle JAPE-1 produces (punning riddles) are not very popular among people aged over eleven years old [Ruch et al., 1990]. For this reason, even the best of JAPE-1's jokes were not expected to get high ratings from the joke judges, who were adults. However, for this initial exploratory evaluation, it was felt that the ability of adults to communicate qualitative criticisms of JAPE-1's output was more important than their enjoyment of the jokes.

5.2.4 Methodology

There were three stages to this evaluation: *data acquisition*, *common knowledge judging* and *joke judging*. During the data acquisition stage, volunteers unfamiliar with JAPE-1 were asked to make lexical entries for a set of words given to them. These definitions were then sifted by a 'common knowledge judge', and entered into JAPE-1's lexicon. The jokes produced by JAPE-1 from these sets of words were then judged by another group of volunteers. Their opinions (both quantitative and qualitative) were then analysed, and compared to the conjectures made in section 5.2.3.

Lexical data acquisition stage

This is the phase in which the words JAPE-1 used to build jokes were defined. The lexicon was intended to be neutral, with all JAPE-1's joke-making knowledge stored in the schemata and templates. It was therefore important that unbiased volunteers define words for JAPE-1, as it is entirely possible that someone familiar with the workings of the system would (unconsciously) bias their entries towards making jokes. Although the volunteers did not know how JAPE-1 worked, they were aware that the words they were defining would be used for joke generation, and this might have affected their responses.

Before the volunteers could define the words, however, an appropriate set of words had to be chosen. First, the homophones to be defined were picked from a list of homophones, supplied by [Townsend and Antworth, 1993]. Homophone pairs were eliminated from this list if they were:

- not adjectives or nouns
- abstract or obscure
- often confused with each other
- alternate spellings of a word
- different mainly in their syntactic category (e.g. "bare" the adjective and "bare" the verb)

Common compound nominals which contained at least one word on the homophone list were then added to the list of lexemes to be defined. Finally, the other words used in the common compound nominals were added to the list. Twenty-one compound nominals and fifty-nine words were then on the list, adding up to eighty lexemes needing to be defined.

The list was divided into ten sets of eight lexemes each, with no two words from the same compound nominal or homophone pair in each set. Ten volunteers were each given a set of lexemes and instructions on how to define them in accordance with the specification of JAPE-1's lexicon (see table 4.2). In their instructions, it was emphasized that they should provide only *typical*, *specific* information about the lexemes they were to define.

Common knowledge judging

After the lexical data had been collected, it was discovered that some of the volunteers had not followed the instructions, and had tried to fill every available slot in each entry. This was not always appropriate; for example, it is difficult to think of a noun which "serial" always describes, for the DESCRIBES_ALL slot. Moreover, some volunteers had left words undefined, while others were perhaps excessively creative in their use of English.

As a lexicon containing these entries would not meet JAPE-1's specifications, a 'common knowledge judge' was recruited (i.e. not the experimenter) to sift the entries. She could take only the following actions:

- **veto** slot values
- **veto** an entire entry
- **move** a slot value into a different slot
- **define** a lexeme, only if it had no entry, either because the original definer did not know what the word meant or because the common knowledge judge had vetoed the entry. In this case, the experimenter would then have veto power only over the new entry. (This did not happen often. Two lexemes were left undefined by the original volunteers, and the common knowledge judge vetoed only two more complete entries.)

After the entries had been sifted in this way, they were entered into JAPE-1's lexicon. JAPE-1 then produced a set of 188 jokes in near-surface form.

Joke judging stage

The 188 jokes produced by JAPE-1 were put into surface form and distributed to fourteen adult 'joke judges' recruited from the experimenter's acquaintances. As each joke was given to two judges, each volunteer had about 25 jokes to judge.

The questionnaire given to the judges had three sections. The first was a list of jokes, each based on a different compound nominal, for them to rate from 0 to 5 on the following scale:

0. Not a joke. Doesn't make any sense.
1. A joke, but a pathetic one.
2. Not so bad.
3. OK. Might actually tell it to someone.
4. Quite good.
5. Really good.

The second was a list of several sets of jokes for them to rate from zero to five, *and* put in order within each set. All the jokes in a set were based on the same compound nominal. This section was necessary because some compound nominals produced a huge number of jokes, while others produced very few, making an even distribution impossible. Unfortunately, this high concentration of similar jokes made it very likely that the judges would suffer from 'joke fatigue', and dislike the repetition.

The third section asked for qualitative information, such as how the jokes might be improved, and if they'd heard any of the jokes before.

Each volunteer completed their questionnaire, and the results were collected, collated and analysed.

Problems with methodology

As mentioned earlier, this testing was not meant to be statistically rigorous. However, this lack of rigour caused some problems.

A major problem was the lack of a control group. We suspect that jokes of this genre (simple question-answer punning riddles) are not very funny even when they are produced by humans; however, the experiment did not show how human-produced jokes would fare if judged in the same way JAPE-1's jokes were, so it was difficult to make the comparison. There was no control group for two reasons: the number of volunteers (having an adequate control group of texts would have doubled the number of volunteers required), and the difficulty of choosing which human-generated jokes would be judged.

The makeup of this group of volunteers was also unusual, which could well have affected the data. Due to the method of distribution of questionnaires (e-mail to friends and associates), all the volunteers were adults with regular access to a computer. Moreover, all the volunteers were either comedians or involved in AI and so had expertise in one aspect or another of this research. This undoubtedly influenced their judgement.

This experiment would have been more useful if the lexicon entries had been made by a large group of adults without any particular interest in computers or comedy, and the intersection of these entries put into JAPE's lexicon. It would also have been improved by mixing the JAPE-1 output with similar human–generated jokes (from [Webb, 1978], for example).

These considerations were taken into account in the confirmatory evaluation (see section 5.3).

5.2.5 Results

The results of the testing are summarised in figure 5.1. The average point score for all the jokes JAPE-1 produced from the lexical data provided by volunteers was 1.5 points, over a total of 188 jokes. Most of the jokes were given a score of 1 ("a joke, but a pathetic one"). Interestingly, all of the nine jokes that were given the maximum score of five by one judge were given low scores by its other judge — three got zeroes, three got ones, and three got twos.

Overall JAPE-1 produced, according to the scores the judges gave, "jokes, but pathetic ones". On the other hand, the conjectures in section 5.2.3 were, for the most part,

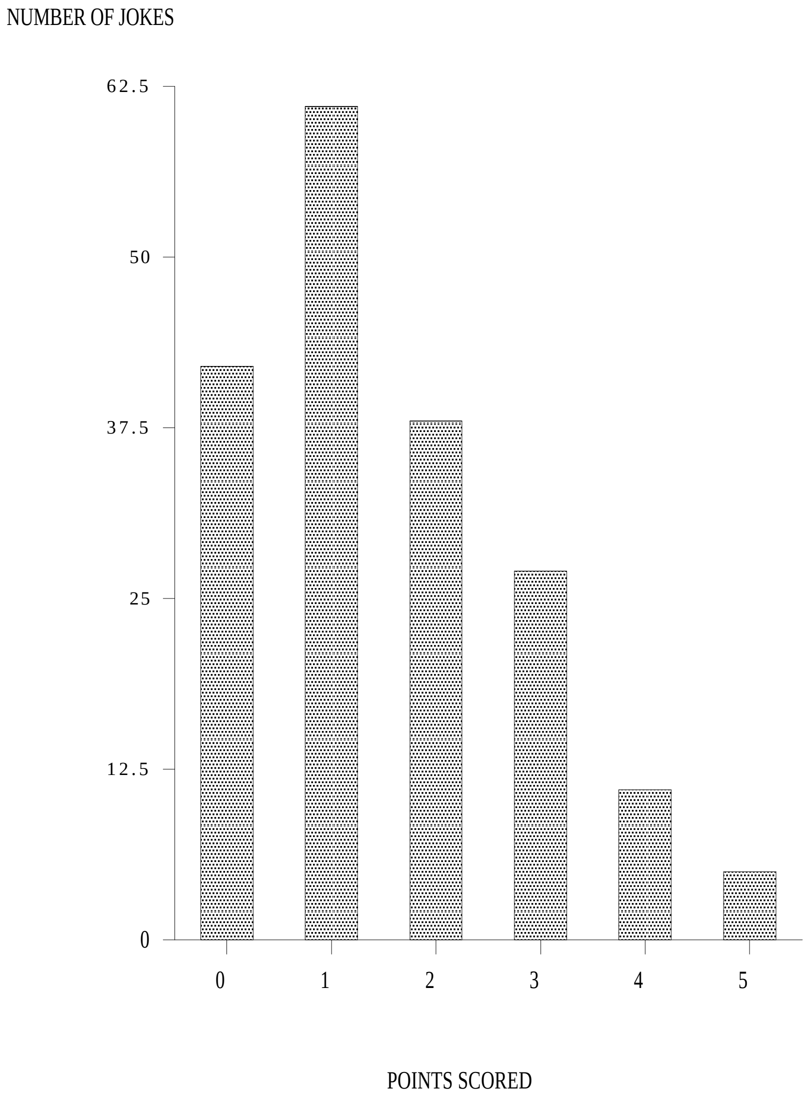


Figure 5.1: The point distribution over all the output

correct, and the results (see the discussion, below) suggest that implementing even the simplest of these hypotheses would improve the resulting jokes. Moreover, the top end of the output (i.e. those jokes that would survive the implementation of trimming heuristics) are definitely of Crack-a-Joke [Webb, 1978] book quality, including:

> What do you call a murderer that has fibre? *A cereal killer.*
> What kind of pig can you ignore at a party? *A wild bore.*
> What kind of emotion has bits? *A love byte.*

For the purpose of informing the further development of JAPE the comments of the joke judges were especially useful. For the most part, they confirmed the conjectures described in subsection 5.2.3. We have concentrated on these comments in the discussion below.

The lexicon and the homophone list

It was clear from the comments that the quality of the lexicon greatly influenced the quality of the resulting jokes. Although we tried to ensure that the volunteer word definers understood and followed the lexicon requirements, and attempted to sift out inappropriate data, several 'bad' entries (or parts thereof) survived to be used in joke construction. It is difficult to trace exactly what went wrong in a failed joke; however, the judge's comments suggest that many of the flaws arose from entries in the lexicon that did not meet the criteria described in section 5.2.3:

- **Semantic information should be included in the lexicon only if it is typical of the word being entered.** With reference to the rather poor joke:

  What do you use to sniff a drilling tool? *A wild bore.*

  one judge asked, "Does 'to wild' mean 'to sniff'?" clearly trying to understand what sniffing has to do with the punchline. It came from the entry for "boar": the definer wanted to express the idea that boars can be used to sniff for truffles. Unfortunately, sniffing as such is not typical of boars, and so does not bring them to mind at all. The common knowledge judge did not veto this particular entry.

- **The information in the lexicon should be *common knowledge.*** This came up several times. Regarding these two jokes:

  What do you use to clothe a coniferous tree? *A fir coat.*

  What do you call a passenger ship you can drink? *Ferry liquid.*[1]

  one judge remarked:

  > ...the fir coat one is funny because most people would know what a coniferous tree is ...whereas the ferry liquid [joke] isn't very obvious.

[1] "Fairy Liquid" is a well-known brand of detergent in Britain.

A related problem was that the judges were from all over the world, and dialects differ. A Canadian judge said that there were:

> ...too many 'Britishisms' not common to North American English, e.g. "stag", "hart", and "queue".

while a British judge pointed out an 'Americanism':

> The bridal shower jokes rely on knowing that a wedding party is called a bridal shower in North America.

- **Jokes should use concrete words ... rather than abstract words.**

  This prediction was not confirmed. The only abstract words included in this lexicon were "love" and "number", neither of which seemed to cause any problems.

- **Jokes should avoid using very general words.**

  Most overly general words were sifted out of the lexicon by the common knowledge judge, as she was specifically instructed to watch for them. Too general words were not, therefore, a particular problem.

Schemata

As predicted, there was not a lot of difference in the average point scores of the three best schemata, *lotus*, *woolly* and *jumper*. For jokes using these schemata, the important factors affecting their scores seemed to be the choice of template and/or compound nominal used to build the joke. The *elan* schema did not do as well as the others, as expected, because there is often some confusion over which words are referring to which concepts.

Templates

The templates varied a great deal in the quality of jokes they produced. A general comment that was made several times was that some of the questions were not 'logically coherent' in some sense. This incoherence was often the result of using one of the templates with inappropriate word ordering. For example, the joke:

> What kind of ship can clean dishes whilst caring for your hands? *Ferry liquid.*[2]

received the comment "[This joke] lost points for saying 'what kind of ship,' and then having an answer that wasn't a ship,", and an average score of 1 point. The corresponding reversed template with better word ordering produced the following joke with the same schema and compound nominal:

> What kind of detergent can cross water? *Ferry liquid.*

This is still not brilliant, but it is more logically coherent, and received an average score of 2 points.

Schema–template pairs

Although some schema–template pairs produced better texts than others, no consistent pattern was seen in this exploratory evaluation. In JAPE-2, the direct linking of schemata and templates was abandoned completely. Instead of linking schemata directly to templates, JAPE-2 uses a small adequate description as an intermediate representation of the lexical information required by the templates to form a riddle (see section 3.6).

Post–production heuristics

JAPE-1 had almost no post–production checking of the riddles produced. It checked only that the constructed compound nominal was not 'real' (i.e. in the lexicon), and that none of the key lexemes used to construct the joke were identical unless required to be so by a schema.

Many of the comments from the joke judges suggested that some (considerably more sophisticated) post–production heuristics could be used to roughly order JAPE's output according to expected funniness. For example:

[2] Hand care is an advertised feature of "Fairy Liquid".

question length: "[The criteria used to judge the jokes were] in order of importance: gut reaction, cleverness, delivery, rhythm. The rhythm and delivery [of these jokes] could be improved. They have to be 'bang-bang' jokes."

alliteration and rhyming: "Phrasing is important. Numbers aren't funny, but 'quirky quantifier' is." This comment was referring to the joke:

> What do you call a quirky quantifier? *An odd number.*

'funny letters': "The grizzly bare joke [What do you call a ferocious nude? *A grizzly bare*] makes good use of the funny letter Z."

subject matter: "I don't find 'gay' jokes funny." This comment was referring to the joke:

> What do you call a homosexual that cleans dishes whilst caring for your hands? *Fairy liquid.*

accidental associations: "Keep the ones with the double meanings, e.g. [the one about] a gay being drunk." This was with reference to the joke:

> What do you call a homosexual you can drink? *Fairy liquid.*

The judge who made this comment gave the joke in question a five and commented on the "obscene" image it brought to mind, while the other judge for this joke gave it a zero, possibly due to its subject matter.

Jokes heard before

Perhaps one of the most positive results of this evaluation was that several of the judges claimed to have heard similar or identical jokes before. This was taken as evidence that JAPE-1 produced some good examples of the punning riddle genre.

Four judges claimed to have heard these JAPE-1 jokes, all variations on the "cereal killer" pun:

> What do you call a murderer that has fibre? *A cereal killer.*
> What do you get when you cross grain with a murderer? *A cereal killer.*

What do you call breakfast food that murders people? *A cereal killer.*

Other JAPE-1 jokes that had been heard before included:

What kind of tree can you wear? *A fir coat.*

What kind of rain has presents? *A bridal shower.*

What do you call a good-looking taxi? *A hansom cab.*

What do you call a perforated relic? *A holey grail.*

What do you get when you cross a savage pig with a drilling tool? *A wild bore.*

Another interesting point was that, well after the evaluation had taken place, it was noticed that JAPE-1 had produced two jokes very similar to ones in the Crack-a-Joke Book [Webb, 1978]. JAPE-1's jokes were:

What kind of pig can you ignore at a party? *A wild bore.*

and

What do you use to help a lemon? *Lemon aid.*

whereas the Crack-a-Joke versions were:

What runs around forests making other animals yawn? *A wild bore.*

and

What do you give a hurt lemon? *Give it lemonade, of course.*

5.2.6 Conclusions

This exploratory evaluation of the prototype system, JAPE-1, was useful in several ways. It confirmed that it was possible for an implementation of our model to generate recognizable jokes from a relatively unbiased lexicon. More importantly, it suggested some ways that the system could be improved:

- The description of the lexicon could be made more precise, so that it would be easier for people unfamiliar with the system to make appropriate entries. Alternatively, a general-purpose, systematically-constructed lexicon could be used, rather than the small, hand-built lexicon used in this evaluation.
- Templates which use the lexemes given to them in the 'wrong' order (i.e. an order that suggests the words in the punchline should be reversed) could be removed.
- The interface between schemata and templates could be made more sophisticated, so that the surface form of the text is appropriate for its semantic content.

Other comments suggested some high–level heuristics that might be used to roughly order the output texts according to their expected funniness. These would:

- Improve the rhythm of the output, by preferring short "bang-bang" jokes to longer jokes, all else being equal.
- Attempt to make the punchline alliterate or rhyme when possible.
- Maximise the number of 'funny letters'.
- Stick to inherently funny subject matter.
- Generate jokes, then test them for serendipitous associations.

However, most of these heuristics are quite sophisticated, and would take more than a simple check for the presence (or absence) of certain lexical items.

Some of these ideas were incorporated into JAPE-2, others were not. For a discussion of how the results from the evaluation were incorporated into the model, please see chapter 3.

5.3 Confirmatory evaluation

5.3.1 Introduction

Once the development of the JAPE-2 model was completed, and its implementation finished, it was necessary to rigorously evaluate its performance. The purpose of this confirmatory evaluation was to determine whether or not the JAPE-2 program was able to generate punning riddles of a similar quality to those generated by human joke experts. A secondary goal was to discover which of JAPE-2's output texts were of the highest quality, and why. These goals can be reformulated in terms of the following research questions:

1. Are JAPE-2's output texts jokes?
2. If so, how funny are they? Are they as funny as human-generated jokes of the same type?
3. What in JAPE-2's jokes contribute to their quality? Is it the way in which the pun is constructed, the subject matter of the joke, or some other factor?
4. Has JAPE-2 replicated any human-generated jokes?

In order to answer these questions, JAPE-2 output texts, human-generated joke texts, and non-joke texts were evaluated by a large number of children.

5.3.2 Hypotheses

The purpose of this study was to summatively evaluate [Mark and Greer, 1993] the behaviour of a pun-generating system, JAPE-2. Since the behaviour being evaluated was pun generation, the experiment compared JAPE-2's output with human-generated puns, and with a control group of non-puns. We hoped to show that:

1. JAPE-2's output texts are more joke-like than non-jokes of a similar form. That is, joke 'experts' judge JAPE-2's output texts to be jokes significantly more often than they judge non-jokes (in question-answer form) to be jokes.

2. JAPE-2's output texts are jokes. That is, on average, joke 'experts' do not judge JAPE-2's output texts to be jokes significantly less often than they judge human-generated punning riddles to be jokes.

3. JAPE-2's output texts are not less funny than puns of the same type generated by humans. That is, joke 'experts' do not give JAPE-2's output texts significantly lower funniness scores than they give to human-generated punning riddles.

Before the above hypotheses could be evaluated, it was necessary to establish that our joke judges were, in fact, experts. That is:

4. The joke 'experts' , on average, judge non-jokes to be jokes significantly less often than they judge human-generated punning riddles to be jokes.

There were also some secondary questions that this experiment aimed to answer. These were exploratory questions that could inform further development of the model, or aid other humour research.

- Are there any correlations between the age, year in school, or reading ability of the joke judges, and the judged quality of the human-generated jokes?
- Is there any relationship between the form of the texts and their perceived funniness or joke/non-joke status?
- Is there any relationship between the schemata used to generate the JAPE-2 jokes and the perceived funniness or joke/non-joke status of the joke?
- Have any of JAPE-2's jokes been heard before?

The experiment was designed so that the main hypotheses could be addressed, and the data gathered could inform the secondary questions.

5.3.3 Design

Subjects

For this experiment, there were several constraints on the judges, or joke experts. They had to be human experts on puns. They had to be able to competently determine whether or not a given text is a pun. It was also necessary that they be able to rate the funniness of a given text.

Other experiments (e.g. [Yuill and Easton, 1993]) have suggested that native English speaking children aged 8-11 years old are able to judge whether or not a spoken text is a pun. This age group is also able to judge the funniness of a spoken text. That is, children in this age group, having heard a text, can say whether or not that text is a pun, and can say how funny they thought it was, at least on a scale of 0-2. They can also be expected to consistently judge at least twenty texts without losing concentration. Moreover, children of that age group are likely to both appreciate and understand jokes of the punning riddle genre [Ruch et al., 1990].

For the above reasons, 8-11 year old children were the best choice for experts on punning riddles for the purposes of this experiment.

However, there were also some problems with this group of judges. We did not know whether a child's judgements would be internally consistent, or consistent with those of other children. Also, we could expect the reading ability in this age group to vary considerably. Finally, we could expect children in this age group to be influenced by the subject matter and reading level of the texts as well as their pun nature (or lack thereof).

Although 8-11 year olds could still serve as pun experts, the texts they judged had to be carefully selected to control for subject matter and reading level. Moreover, each child also had to judge whether or not non-puns (i.e. sensible or nonsense questions and answers, with no punning element) were jokes, so that their ability to distinguish jokes from non-jokes could be established. Also their year at school and age were recorded, so that variations in these could be taken into consideration.

Since the children's reading abilities were uncertain, they were exposed to the texts in both written and recorded form. In order to avoid bias in the experimenter's reading of the texts, an actor was asked to read all the texts onto tape with the same voice and general intonation pattern.

Materials

The initial materials were the set of texts that the judges were to evaluate. Because the purpose of this experiment was to compare JAPE-2-generated texts with human-generated punning riddles, representative examples of each were included. The judges' ability to distinguish jokes from non-jokes also needed to be checked; for this reason, non-jokes were also included.

There are two relevant kinds of non-jokes — sensible question-answer pairs, and nonsense question-answer pairs. A sensible question-answer pair is a two sentence text in which the second sentence is a truthful, expected answer to the question in the first sentence. For example:

> What do you get when you cross a horse and a donkey?
> *A mule.*

A nonsense question-answer pair is a two sentence text with the syntactic form of a question and answer, but without any connection, sensible or punning, between the topic(s) of the question and that of the answer. For example:

> What do you get when you cross a murderer and a ferry?
> *A citrus fruit.*

Thus, four sets of materials had to be generated: the JAPE-2 generated texts, the human-generated jokes, the sensible non-jokes, and the nonsense non-jokes. The latter two sets of texts were controls for the evaluation of the first two sets.

Since we are only interested in humour caused by the pun nature of a text, it was also necessary to control for subject matter and form of the joke. For this reason, all the texts in the experiment had subjects selected from the same set of possible subjects, where the 'subject' of a text is the set of nouns, adjectives and verbs used in that text (see appendix H for a list of the permitted vocabulary). Also, all the texts in the experiment had forms selected from the same set of possible forms, where the 'form' of a text is one of JAPE-2's twelve sentence forms (see appendix I).

It was important that each text be judged by several different children, so that their evaluations could be compared. However, children in this age group are easily bored, and their ability to judge texts can be expected to deteriorate if they are shown large numbers of texts. We estimated, after [Yuill and Easton, 1993], that children in this age group could judge twenty texts without becoming too distracted from their task. For this reason, the texts were divided into sets of twenty. Each set had approximately the same number of each type of text. In order to eliminate ordering effects such as boredom, each set of texts was also randomised into several different sequences of texts.

A set of JAPE-2 texts was selected to be representative of JAPE-2's output. In order to obtain data for this experiment's secondary question on the relationship between readability and funniness (see subsection 5.3.2), it was important that all of the vocabulary used in the texts also be in the MRC psycholinguistic database (see section 4.3.4 for a description of this database).

The following procedure was used to generate the texts to be used as the materials in this experiment.

1. For each of JAPE-2's nine schemata used in the evaluation (see appendix J), JAPE-2 generated as many output texts as could be generated using words from the psycholinguistic database (so that psycholinguistic data was available for all potentially testable JAPE-2 texts).

2. For each schema, *if* the schema generated more than one–ninth (nine schemata were tested) of the required number of texts, we chose the texts with the highest familiarity, concreteness, and imageability scores (see section 4.3.4), yet which had different subjects (i.e. did not share any nouns, verbs or adjectives). The *bazaar*, *lotus*, *rhyming*, *elan* and *phonsub* schemata all over-generated.

3. Some schemata generated *fewer* than one–ninth of the required number of texts.

The shortfall was made up by choosing the texts with the highest familiarity, concreteness and imageability scores generated by the schemata that produced more texts than required. The *negcomp*, *poscomp*, *vn*, and *hopchew* schemata all under-generated.

These texts were the JAPE-2 output texts. To find the set of human-generated texts:

1. We went through the selected JAPE-2 texts, and determined which of JAPE-2's sentence forms (see appendix D) were used in the generation of these texts. These, stripped of JAPE–related constraints, became the set of allowable forms (see appendix I).

2. We went through the selected JAPE-2 texts, and made a list of all the nouns, verbs and adjectives used in these texts. This set of words, and all of their sister and daughter nodes in WordNet (see section 4.3.3), became the set of allowable vocabulary items (see appendix H).

3. From published books of jokes, not examined during JAPE's development, we selected all the jokes which use only the allowable forms and subjects. Minor adjustments of sentence forms and subjects were done by an impartial adult, in order to fit the human-generated jokes to the experimental criteria. In other words, if one of the jokes was almost in an allowable form, minor syntactic changes could be made; likewise, minor changes in a joke's vocabulary could be made to give it an allowable subject. What constituted a 'minor' change was left to the impartial adult's discretion, as long as the resulting text was, to their judgement, a joke.

   For example, the joke:

   > What do sea-monsters eat? *Fish and ships.* [Young, 1993]

   was adjusted in both vocabulary and form to read:

   > What kind of food does an octopus eat? *Fish and ships.*

   so that both its form and its subject were allowable.

The books used were: "The A-Z of Animal Jokes" [Anderson, 1987], "Mirthful Kombat - Jokes with BYTE!" [Byrne, 1995], "The Cat Joke Book" [Abbott, 1993], "Cackles from the Crypt" [Fremont, 1993], "The Six-Million-Dollar Cucumber" [Churchill, 1976], "The Wackysaurus Dinosaur Joke Book" [Phillips, 1991], "The Raspberry Joke Book" [Jam, 1991], "Ready Teddy Go Joke Book" [Rayner, 1991], "The Schoolkids' Joke Book" [Girling, 1988b], "Smart Alec's Spooky Jokes for Kids" [Alec, 1987], "Super-Duper Jokes" [Young, 1993], "The Teddy Bear Joke Book" [Brandreth, 1990], "The Upside Down Joke Book" [Hegarty, 1992], and "The Vampire Joke Book" [Forrester, 1994].

4. There were more suitable human-generated jokes than required, so we randomly chose the required number. This formed the set of human-generated texts.

To find the set of sensible non-jokes:

1. The set of permissible subjects and the set of permissible sentence forms were given to an impartial adult. She was asked to fill in the blanks of the forms with the subjects in as many ways as she could to produce 'true' questions and answers.

2. The resulting set of texts were given to a second impartial adult. He was asked to eliminate any which did not "make sense". The remainder formed the set of sensible non-jokes.

3. The resulting set was larger than required, so a suitable number of texts were randomly chosen from the set.

In order to determine the set of nonsense non-jokes:

1. Allowable subjects were inserted randomly into the permissible sentence forms[3].

2. If any of the resulting texts accidentally happened to be either sensible question-answer texts or punning riddles, as judged by an impartial adult, then they could

[3] This was done using the sentence form module of JAPE-2, detached from the rest of the program, as a DCG with the set of subjects in random order as its lexicon.

be eliminated from the set. This situation, however, did not arise. The remaining texts formed the set of nonsense non-jokes.

3. The resulting set was larger than required, so we randomly chose a suitable number of texts from the set to use in the experiment.

The initial materials were generated using the above procedure. They consisted of: a set of JAPE-2-generated texts, fairly evenly distributed across the schemata, each with different subjects; a set of human-generated jokes, using the same subjects and forms as the JAPE-2 texts; a set of sensible non-jokes, again using the same subjects and forms; and a set of nonsense non-jokes, also using the same subjects and forms. All these texts had subjects selected from the same set of subjects (i.e. the subjects used in the JAPE-2 texts), and forms selected from the same set of forms (i.e. JAPE-2's sentence forms). Moreover, they were all rateable using the MRC psycholinguistic database, since all the subject words were selected from the set of words in that database.

The initial set of texts was then divided into test sets of twenty texts each. Texts of each type (i.e. JAPE-2 generated, human generated, sensible and nonsense) were spread evenly across the sets.

Each test set was then randomised and recorded. Since we could not be sure that each judge would finish judging their set of texts, and to eliminate ordering effects, each test set was randomised into several different sequences, so that each text was likely to be judged the same number of times. To generate and record these sequences:

1. We took one test set, and randomised the order of the texts. This generated one sequence of texts.

2. We repeated step 1 four times, so that this set was in five different orderings.

3. We repeated steps 1 and 2 for each set, until there were five different orderings of each of the sets.

4. We recorded each of these sequences on a separate tape, marking it carefully (e.g. Test Set 3, Ordering 2). All the texts were recorded with the same voice, using the same intonation patterns.

In addition, one human-generated joke and one sensible non-joke was also recorded, to be used as examples for the judges. The example texts were not from the test material set.

Equipment and setting

In choosing the setting and equipment for this experiment, the main consideration was how to allow the judges to evaluate the texts in such a way that other factors interfered as little as possible.

The equipment required for the experiment was as follows: tapes of texts (as described above), one example tape, several tape recorders with headsets, electrical outlets and extensions, chairs and desks, writing implements, and a sufficient number of cover sheets, example sheets, and response sheets (see appendix K for a typical response sheet).

The response sheets contained: some simple printed instructions to supplement those given by the experimenter; an area for the the child to fill in their name, age, year or form at school, and whether or not they like jokes; and twenty numbered response areas, one for each text. In each response area, there were three questions relating to the text heard on the tape:

- Was that a joke? In response, they circle either a "YES" or a "NO".
- How funny was it? In response, they circle one of five simple faces: frowning mouth open, frowning mouth closed, flat mouthed, smiling mouth closed or smiling mouth open. Under the faces there is text saying "not funny at all" "not very funny" "not sure" "funny" and "very funny".
- Have you heard it before? In response, they circle either a "YES" or a "NO".

There was also some space for comments at the end of the response sheet, which both the experimenters and the judges could fill out if necessary. The comments were not taken as part of the formal evaluation. Each response 'sheet' was made up of several pages, stapled together.

The example response sheets had two numbered response areas like those on the main response sheets. No other information was recorded on these; they were used only to familiarise the children with the procedure.

Staff requirements for the experiment were minimal. Aside from the children, only the experimenter and an assistant were present. The experimenter explained the procedure to the children and conducted the example, while the assistant made sure that the children were sitting comfortably and were equipped with a working tape recorder and headset, a tape, a matching response sheet, and a pen or pencil. Either answered any questions that came up.

Procedure

Before the experiment began, the experimenter set up and tested the tape recorders in the experiment room. The tape (i.e. sequence of texts) to go in each machine was chosen randomly. Meanwhile, the assistant organised the first group of children to participate in the experiment. She checked that they met the criteria (between eight and eleven years old, native English speaker, able to read). Group size was limited to five.

Before the experiment started, the experimenter explained that we needed their help in deciding whether some 'things' are jokes or not. The experimenter told them they were to listen to the tape in their machine, and tell us on the response sheet if what they heard was a joke, how funny it was and whether they had heard it before. These instructions were repeated briefly at the beginning of each tape, and at the top of each response sheet. The experimenter also explained that, should they wish to stop at any point, they should raise their hand and they would be allowed to go. Finally, the experimenter told the children not to tell any jokes heard during the experiment to other children, because the other children might want to participate in the experiment too. No mention was made of the fact that some of the texts were computer-generated.

The experimenter then asked the children to listen to the example tape, and fill in the example response sheet. Any obvious misunderstandings about the procedure (as opposed to the nature of puns, the meanings of words etc.) were corrected at this point by the experimenter or by the assistant.

The experimenter and the assistant then helped the children fill in the first part of the response sheet, which asked for the age and year or form at school of the child. It also asked if they like jokes or not. The experimenter and the assistant then started each tape recorder, ensuring that the children could hear the tapes clearly.

Once the tapes had started, both the experimenter and the assistant watched carefully for any problems. If a child raised their hand, or looked confused, the tape was stopped and any problems resolved. If a child wanted to stop doing the experiment, or misbehaved in such a way that they were disturbing the other children, they would have been be allowed to go (accompanied by the assistant, if necessary), and the point (i.e. text number) at which the tape was stopped noted on the response sheet.

As the tapes finished, the experimenter and the assistant asked if the children had any comments about what they heard. If the children could not write the comments themselves, the experimenter made brief notes for them on the sheet. The children were then allowed to go.

Including instructions and the writing of comments, the experiment took no more than 20 minutes of each child's time. Including turn around, each cycle took no more than 30 minutes. Each tape (i.e. sequence of texts) was judged at least once.

5.3.4 The pilot study

The purpose of the pilot study was to iron out problems in the experimental design, and suggest modifications to the research questions.

One hundred texts were generated as described in section 5.3.3. Of these, fifty were JAPE-2 generated texts, thirty were human-generated jokes, ten were sensible non-jokes, and ten were nonsense non-jokes. These were evenly divided into five sets of twenty texts each, which were then randomised into a total of twenty sequences of texts. These texts were read onto tape by an actor, who had been asked to read each with the same deadpan voice and intonation.

The pilot study took place at Craiglockhart Primary School. Twenty children took part. The children were selected randomly from four classrooms (five children from each) by the assistant head teacher. Two of the classes were year four (mostly eight year olds) and two were year six (mostly ten year olds). The children were divided evenly into four groups, so that there were one or two children from each class in each group of five children.

As there were twenty children, and twenty tapes, each sequence of texts was judged once only. Since each text appeared in four sequences, this meant that each text was judged by four different children.

We were given the 'Resources' room at the school, which was adequately supplied with tables, chairs and power outlets. The children were brought into the room by their teacher, and collected twenty five minutes later.

Pilot results

The purpose of the pilot was not so much to gather data, as to test and adjust the design of the experiment. As it turned out, very little adjustment was required. There were also some significant results, even with this relatively small group of children.

We found that it was important to emphasize that the experiment was not a test of their abilities in any way, and that we were asking for their judgement as 'joke experts'. It was also important to say that each tape was different, as a few children were tempted to copy from their neighbours. They stopped as soon as they were shown that their neighbour's sheet was quite different.

Two aspects of the results were analysed: the 'jokiness' of a text (i.e. the text's success rate in being judged a joke), and the 'funniness' of a text (i.e. the average score, from 1 to 5, given to that text in response to the "How funny is it?" question). For a more detailed discussion of the analysis of 'jokiness' and 'funniness' data, please see section 5.3.6. The results for the pilot were as follows.

- The difference in the 'jokiness' between the two sets of non-joke texts was not significant ($p > .1$). For this reason, the non-joke texts were treated as a single category for the rest of the analysis.

- The 'jokiness' of the human-generated jokes was significantly higher than that of the non-joke texts ($p < .005$). This confirms hypothesis 4 in section 5.3.2.
- The 'jokiness' of the JAPE-2 generated texts was significantly higher than that of the non-joke texts ($p < .005$). This confirms hypothesis 1 in section 5.3.2.
- The 'jokiness' of the human-generated jokes was significantly higher than that of the JAPE-2 generated texts ($p < .005$). This fails to confirm hypothesis 2 in section 5.3.2.
- There was no significant difference between the 'funniness' of the texts judged to be jokes, regardless of the source of the text ($p > .1$). That is, jokes from any particular source were not judged to be significantly more or less funny than jokes from any other source.

None of the secondary research questions were investigated during the pilot study.

The only adjustments to the design of the experiment suggested by the study were clarifications to the instructions given to the children. All other aspects of the design remained the same for the main experiment.

5.3.5 The main experiment

The main experiment took place over two days at the Edinburgh International Science Festival on an April weekend.

122 children took part in the experiment, most aged between eight and eleven years old, although a few slightly older or younger siblings were permitted to participate at the request of their parents.

Two hundred texts were judged in the experiment. Of these, one hundred (five sets of twenty) were identical to those used in the pilot experiment; a further hundred were generated according to the instructions in section 5.3.3. All in all, there were one hundred JAPE-2 generated texts, sixty human-generated texts, twenty sensible non-jokes, and twenty nonsense non-jokes. These were evenly divided into ten sets of twenty texts, which were then randomised into forty sequences. Each sequence of twenty texts was judged by at least three children, meaning that each text was judged approximately twelve times.

There were no technical hitches, although two minor errors in the materials were detected too late to be fixed:

- One JAPE-2 text was included in two sets.
- One questionnaire (seen by three children) contained one incorrect text. The correct text was on the tape.

Almost all of the children were able to follow the instructions without any problems. One child did not fill in any funniness data, while another missed a page in his questionnaire. One child seemed to have significant difficulties reading the texts, and this was noted on his questionnaire. The remainder of the questionnaires were correctly filled out.

All of the children behaved well, and all completed the full experiment.

Our 'room' was a corner of a large room, separated from the rest of the space with room dividers. It was quiet enough for our purposes, and the somewhat more exciting experiments next door attracted many potential subjects to our corner.

5.3.6 Results

This confirmatory evaluation provided adequate data to assess the hypotheses described in section 5.3.2. It also gave some significant answers to some of the secondary research questions.

During the experiment, questionnaires were given out to 122 children. Of the forty sequences of texts, thirty-eight were evaluated three times, and two were evaluated four times. This means that each of the ten sets of texts was evaluated at least twelve times.

Some of the 122 questionnaires returned contained data that was flawed in some way:

- Two questionnaires were filled in by seven year olds (both siblings of other subjects). Both said they were almost eight.

- Four questionnaires were filled in by twelve year olds (again, siblings of other subjects). Three said they had just turned twelve.
- One text sequence was marred by a mismatch between the tape and the questionnaire. One text was read correctly on the tape, but the questionnaire contained a different text. This flawed sequence was evaluated by one child before it could be corrected.
- One child had obvious difficulty reading the questionnaire.
- One of the seven year olds did not fill in any funniness data.
- One child missed a page in the questionnaire. We were unable to tell whether the remainder of his responses corresponded to the appropriate texts.

Of these, only the last two were discarded completely. The three containing the mismatch were assumed to be correct otherwise, and only the data on the mismatched text was discarded. The rest have been included in the data, but the problems with them have been noted.

Even after these deletions, most of the two hundred texts have been evaluated by twelve children, and all have been evaluated by at least nine.

For each text, three types of evaluation were given by the children.

Jokiness: For each text, each child was asked whether or not that text was a joke. Each text is given a zero score if evaluated as a non-joke, and a score of one if evaluated as a joke. If not evaluated (i.e. that part of the questionnaire was not filled in), no score is given. The average of all the scores for that text is taken to be the 'jokiness' of the text (i.e. the proportion of the children who judged it to be a joke).

Funniness: Each child gave each text a score. The scores were:

1. "not funny at all"

2. "not very funny"
3. "not sure"
4. "funny"
5. "very funny"

If a text was not evaluated for funniness, it is not given a score at all.

The children were not given instructions on how to rate the funniness of a non-joke. For this reason, if a child rated a text as a non-joke, the funniness score that child gave that text was discarded.

The average of all the "How funny is it?" scores for a text is taken to be the 'funniness' of the text.

Heard before: Each child was also asked if they had heard the text before or not. Each text is given a zero score if not heard before, and a score of one if heard before. The average of all the "Have you heard it before?" scores for a text is that text's 'heard before' score.

The 'jokiness', 'funniness', and 'heard before' scores for each text have been given in appendix L. The texts are ordered first by 'jokiness', then by 'funniness'.

Jokiness

The next stage in the analysis was to compare the average 'jokiness' of the texts grouped by their source: JAPE-2 generated jokes, human generated jokes, sensible non-jokes, and nonsense non-jokes. The average 'jokiness' of each type of text was calculated, and is shown in figure 5.2. Then the significance of the differences in 'jokiness' was calculated, using the Wilcoxon Signed Rank Test [Greene and D'Oliveira, 1992]. It was found that:

- The children found sensible non-jokes and nonsense non-jokes equally 'joke–like'. That is, there is no significant difference ($p > 0.05$) between the 'jokiness' scores of the two types of non-jokes. For this reason, we do not distinguish between the two types of non-jokes for the rest of the 'jokiness' analysis.

- The children could distinguish human jokes from non-jokes. That is the 'jokiness' of the human-generated texts is significantly ($p < 0.01$) higher than that of the non-joke texts. This confirms hypothesis 4 in section 5.3.2.

- The 'jokiness' of the JAPE-2 generated texts is significantly ($p < 0.01$) higher than that of the non-joke texts. This confirms hypothesis 1 in section 5.3.2.

- The 'jokiness' of the human-generated jokes is significantly ($p < 0.01$) higher than that of the JAPE-2 generated texts. This fails to confirm hypothesis 2 in section 5.3.2.

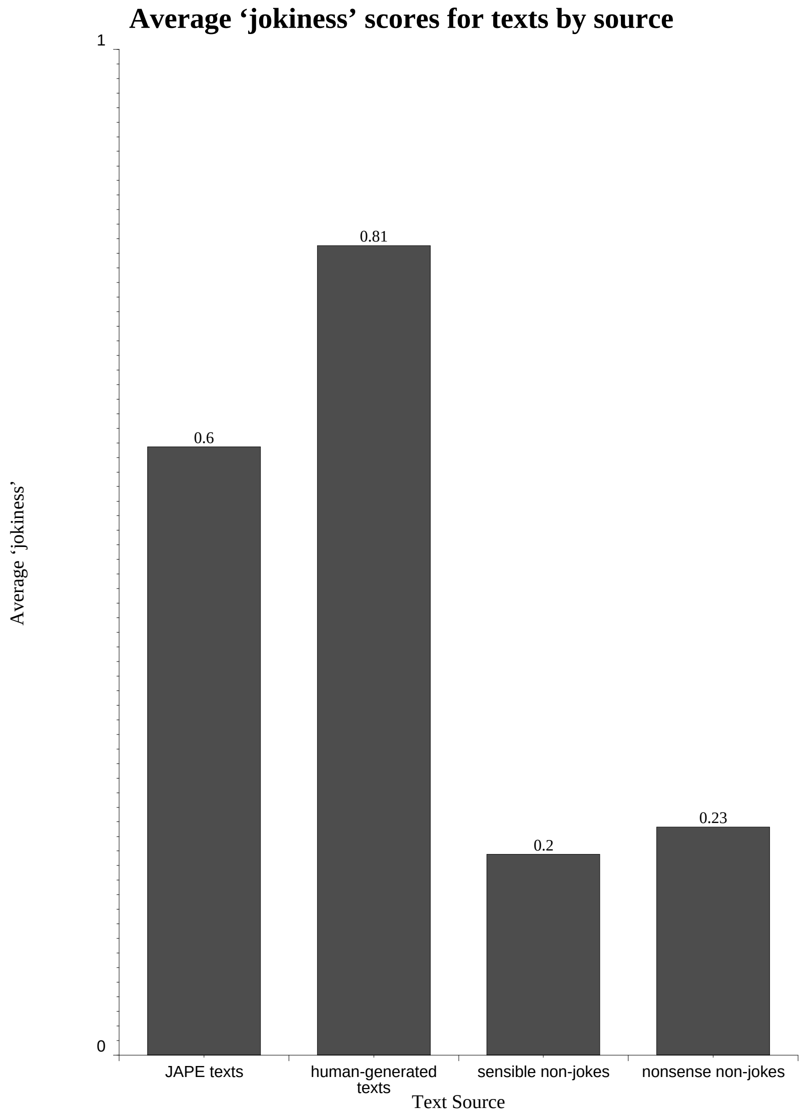


Figure 5.2: Average 'jokiness' scores for texts from each source.

A secondary research goal (see section 5.3.2) was to compare the success of JAPE-2's various schemata at generating jokes. To do this, the JAPE-2 generated texts have been categorised according to the schema that generated them. The 'jokiness' scores for each type have then been compared. See figure 5.3 for the results.

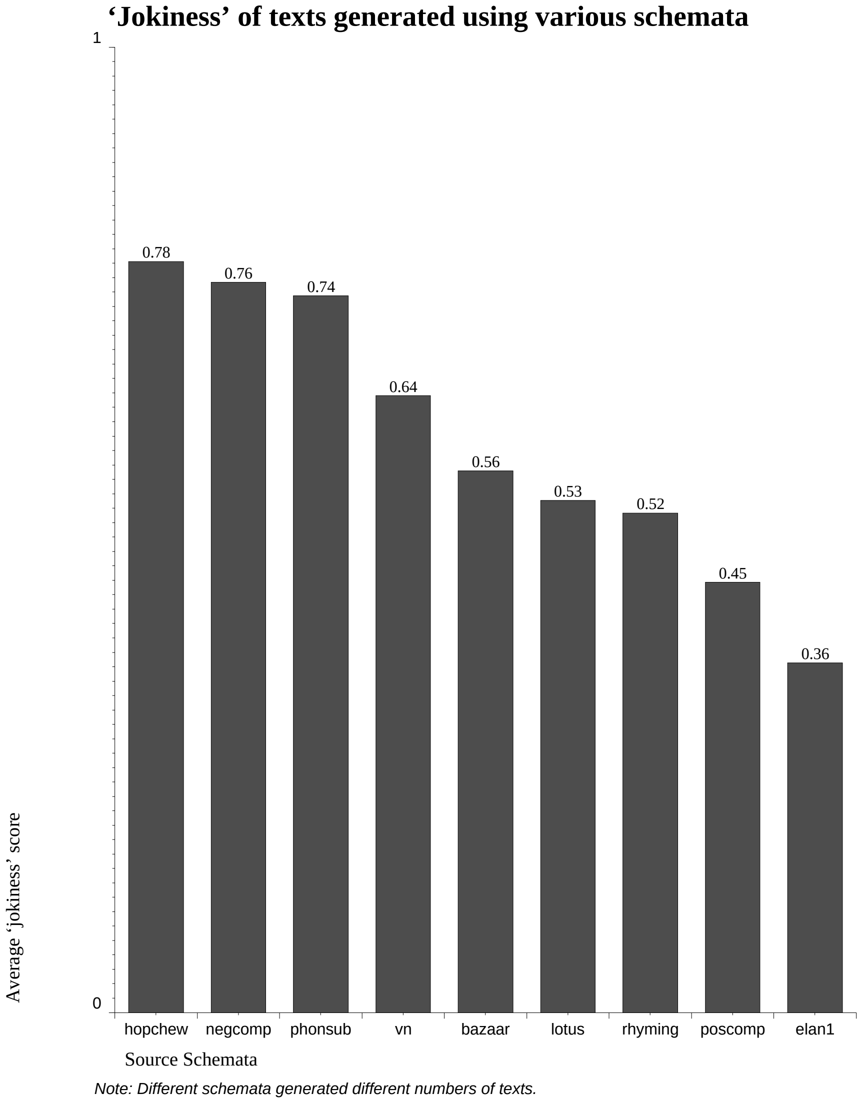


Figure 5.3: Average 'jokiness' scores for texts generated by various schemata.

Because not all schemata were able to generate a large number of texts, most of the differences are not significant. The exceptions are that the **phonsub** schema generated texts with significantly higher 'jokiness' scores than both the **lotus** schema and the **rhyming** schema.

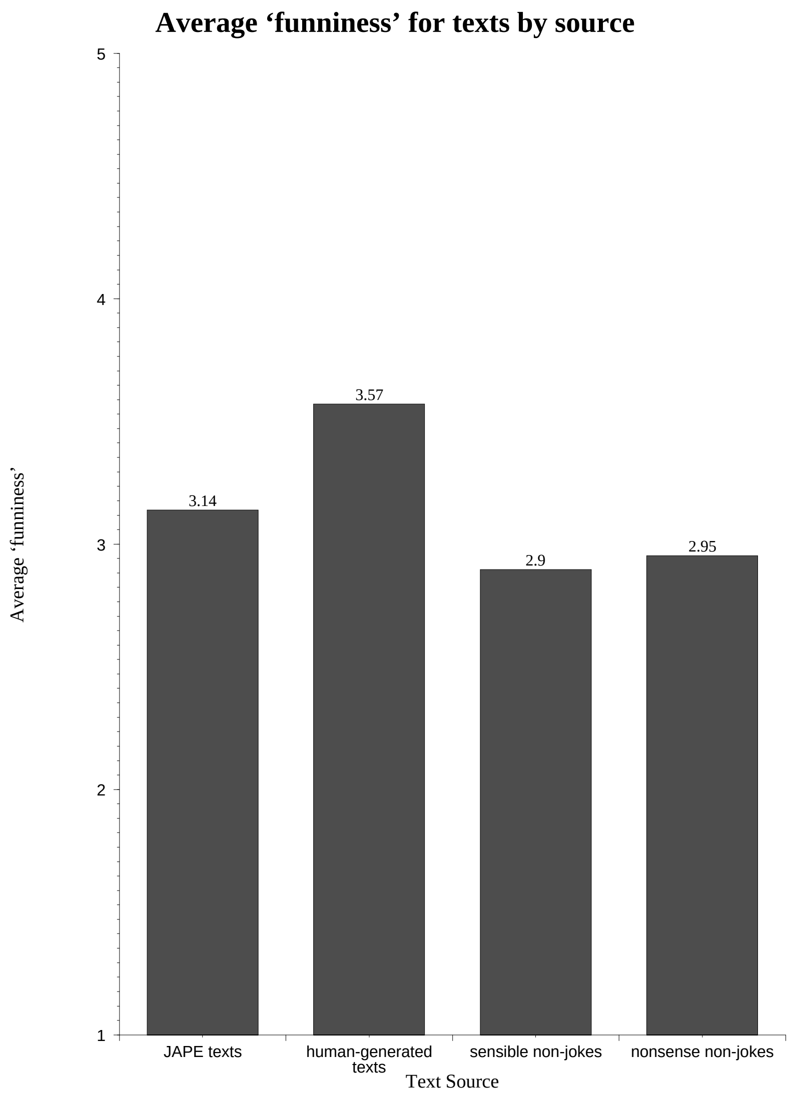


Figure 5.4: Average ‘funniness’ scores for texts from each source.

Funniness

Similar calculations were done for the ‘funniness’ scores of the types of text (see figure 5.4). The results are:

- There is no significant difference in funniness between the two types of non-joke ($p > 0.05$).
- Human-generated jokes are significantly funnier than non-jokes ($p < 0.05$).
- JAPE-2 generated jokes are significantly funnier than non-jokes ($p < 0.05$).
- Human-generated jokes are significantly funnier than JAPE-2 jokes ($p < 0.05$).

Recall that a particular funniness score is only used if the child who gave it also judged that text to be a joke. This is because children were not given any instructions on how to judge the funniness of a non-joke text, and they adopted several different, and inconsistent, strategies. When evaluating the funniness of texts that they judged not to be jokes, some gave only the lowest score, some gave a range of scores, and some gave no funniness score at all.

Readability

Several of the secondary research questions (see section 5.3.2) relate to the 'readability' of the texts. Data from the MRC psycholinguistic database was used to assess the 'readability' of all the JAPE-2 or human-generated texts. See section 4.3.4 for a discussion of the MRC psycholinguistic database and its role in JAPE-2.

Each text is given four scores from the psycholinguistic database: familiarity, concreteness, imageability and age of acquisition. To do this, each key word (i.e. non-template words) in the text is given its scores for these four measures from the database. We assume that the readability of the whole text is limited by the *least* readable of the words that make it up. That is, a text is only as familiar (or imageable, or concrete) as the least familiar word it contains. For this reason, the scores for the whole text are taken to be the *worst* (i.e. lowest for familiarity, concreteness and imageability, and highest for age of acquisition) of the scores for the key words in that text. If a key word has no score for a particular measure, then no score for that measure for that text is recorded. Because age of acquisition data is quite sparse in the database, this procedure leaves very few texts with age of acquisition scores, and that measure is abandoned for the remainder of the analysis.

For example, to calculate the psycholinguistic scores for the text:

> How is a shark like a bass? *They are both fish.*

we take the set of key words ("shark", "bass" and "fish"), and find their scores in the MRC database. They are:

shark: Familiarity: 516

Concreteness: 611

Imageability: 602

Age of aquisition: No score

bass: **Familiarity:** 540

Concreteness: 547

Imageability: 544

Age of aquisition: No score

fish: **Familiarity:** 548

Concreteness: 597

Imageability: 615

Age of aquisition: No score

The worst scores for each measure are:

Familiarity: 516

Concreteness: 547

Imageability: 544

Age of aquisition: No score

So these are taken as the scores for this text.

The first step in the readability analysis was to compare the average readability scores for human-generated texts and for JAPE-2 texts. Please see figure 5.5. All of the differences are significant; that is, human-generated jokes are significantly more familiar ($p < .01$), concrete ($p < .005$), and imageable ($p < .001$) than JAPE-2–generated jokes.

The next step was to find out if there was a correlation between any of the three psycholinguistic measures and the 'jokiness' scores for the human-generated jokes. No significant correlation was found for any of the measures. The same test was then performed for JAPE-2 jokes alone. Again, none of the correlations were significant.

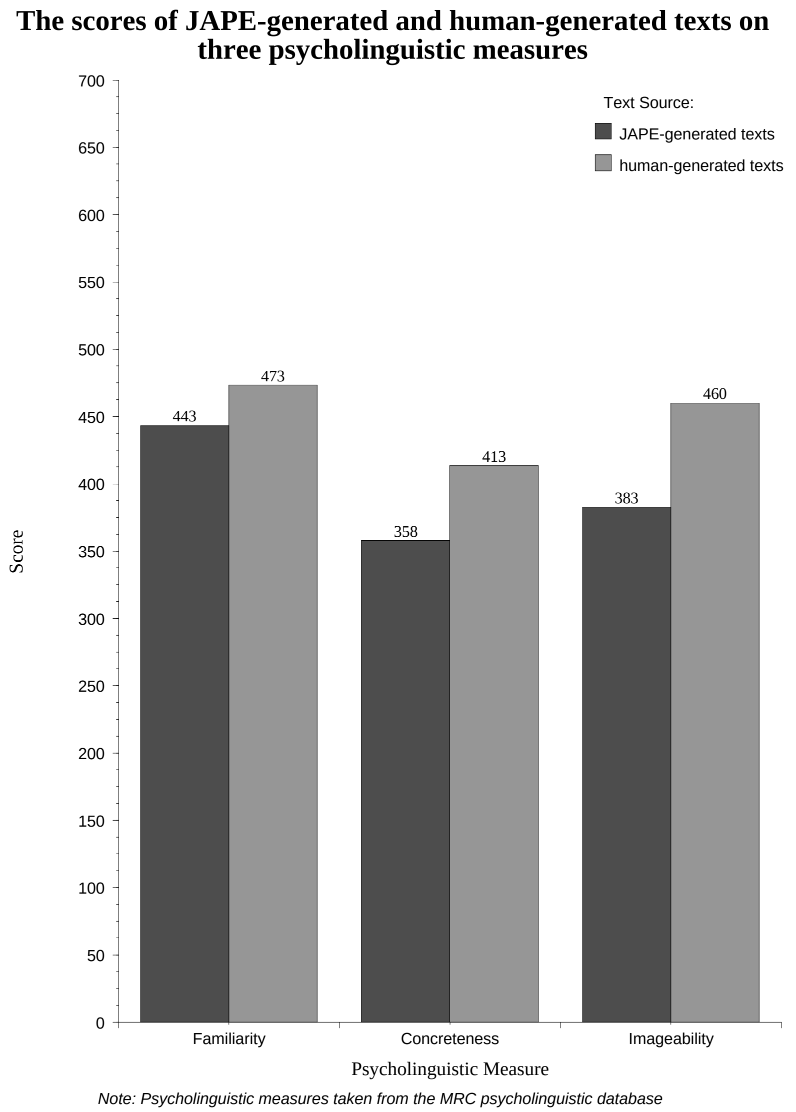


Figure 5.5: Psycholinguistic data for JAPE-2 texts and human texts compared.

The data for the human-generated texts and the JAPE-2 generated texts were then grouped together, to see if this larger set of texts would show a significant correlation between the psycholinguistic scores and the 'jokiness' of the texts. In fact, three correlations were found. A significant ($p < .01$) correlation between the familiarity score for a text and its jokiness was found, with a Spearman coefficient of .23. A correlation between the concreteness of a text and its 'jokiness' was also significant ($p < .002$), with a Spearman coefficient of .2878, as was the correlation between imageability and 'jokiness' ($p < .001$), with a coefficient of .2818.

This would seem to indicate that there is a small but significant correlation between readability, in the form of the three psychoinguistic measures, and the jokiness of a text (i.e. the fraction of children rating it as a joke).

We then tried to correlate readability and 'funniness' scores for human-generated jokes. None of the correlations were significant. The same test was then performed for JAPE-2 jokes alone. Again, none of the correlations were significant. We then grouped together the data for the human-generated texts and for the JAPE-2-generated texts. We found that there was a correlation beteween familiarity and funniness ($p < .02$, coefficient .2121), concreteness and funniness ($p < .001$, coefficient .3152), and imageability and funniness ($p < .001$, coefficient .3697). This indicates that there is a small but significant correlation between the readability and the funniness of a text.

Other Results

The 'heard before' data for human-generated jokes and JAPE-2 jokes was also compared. For those texts judged to be jokes, more children claimed to have heard the human-generated before than claimed to have heard the JAPE-2 jokes before. This result is both statistically significant and completely unsurprising.

It was also expected (see section 5.3.2) that there would be some correlation between the age of the children and their ability to identify jokes. The ages of the participants in the experiment were compared with their ability to identify jokes. A child's 'joke recognising ability' is taken to be the proportion of human-generated jokes that they successfully recognised as jokes. Using the Spearman correlation test [Siegel and Jr, 1988], it was found that there was a significant positive correlation ($p < 0.002$) between a child's age and that child's ability to recognise a joke. The Spearman correlation coefficient between the age of the participant and their ability to identify jokes was .2596. This is not a large correlation; it is, however, significant.

Improvement by elimination

One of the purposes of this evaluation was to test systematic ways of eliminating poor output texts automatically. This could be done by removing parts of JAPE-2 which do not work as hoped, or by constructing an output filter, that eliminates text that do not meet some criteria.

The analysis of the results shows that some schemata are significantly better than others. In particular, the schemata which require information not readily available in WordNet (see section 4.3.3) — **lotus, rhyming, poscomp** and **elan** — performed badly. It was possible that these schemata, hindered by lack of information, were producing poor outputs texts which, in turn, were bringing down JAPE-2's average performance.

To check this, we eliminated all the texts produced by the underinformed schemata, then recalculated the averages and significance. Although this increased the average jokiness of the JAPE-2 output texts to 0.68, the difference between this and the average jokiness of the human generated texts was still significant ($p < 0.05$). However, if the **bazaar** schema, which uses the weak juxtaposition mechanism (see section 3.2.5), is also removed, difference in jokiness between human generated texts and the remaining 28 JAPE-2 generated texts is no longer significant ($p = 0.2$). This suggests that eliminating poor or underinformed schemata would improve the overall quality of JAPE-2's output.

Another approach would be to filter JAPE-2's output after generation, according to the psycholinguistic scores for the texts. To check this, we eliminated all the JAPE-2 texts with any psycholinguistic score below the average for that score for human-generated jokes. This did not significantly improve the quality of JAPE-2's output texts. Then, we removed the texts scoring (on any measure) below 350, 375, 400 and 450, and charted the results (please see figure 5.6). At the 400 threshold, the difference between the jokiness of the human-generated texts (0.80) and that of the remaining 20 JAPE-2 texts (0.72) was no longer significant ($p = 0.12$). At the 450 threshold, the difference between the jokiness of the human-generated texts and the remaining 9 JAPE-2 generated texts is not significant at all ($p = 1$). This suggests that eliminating those texts with low psycholinguistic scores would improve the overall quality of JAPE-2's output.

It could be argued that comparing the 'best' (i.e. most 'readable') JAPE-2 jokes with the full set of human–generated jokes is somehow unfair; however, it should be kept in mind that *all* the human–generated jokes were sifted for comprehensibility and humour content by the writers and editors of the joke books in which they were found.

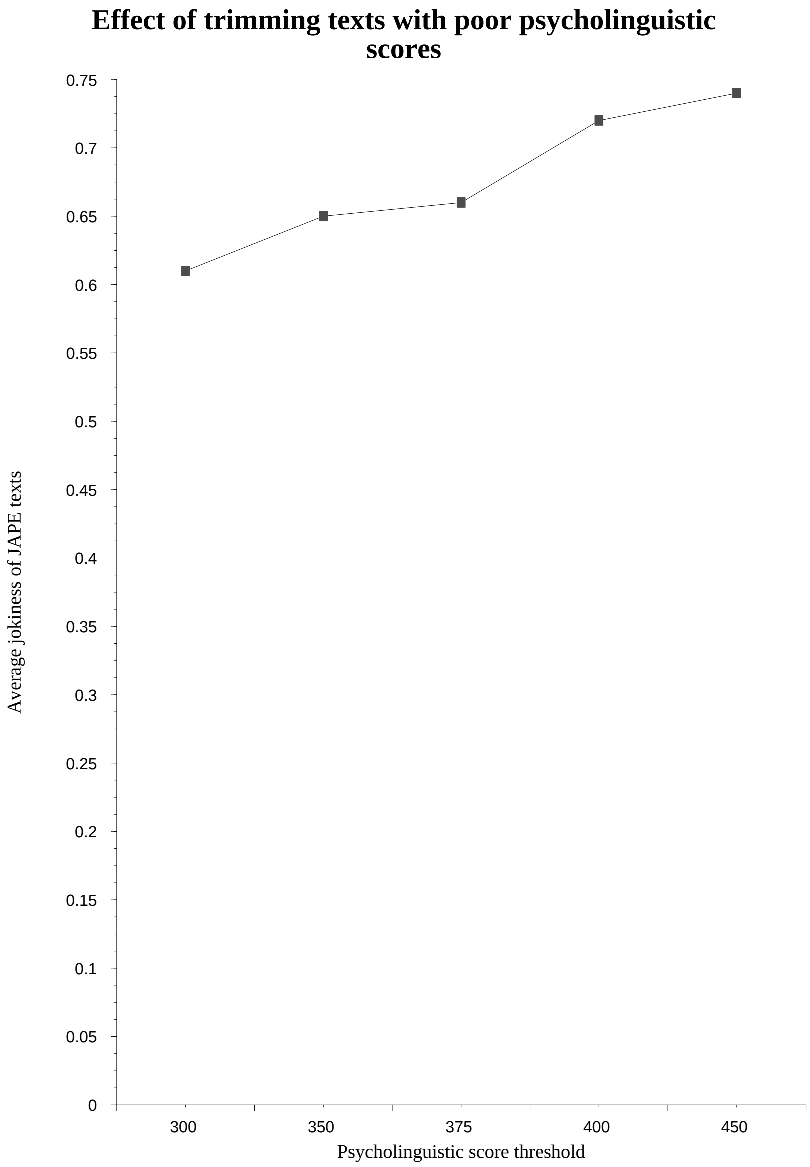


Figure 5.6: The effect of trimming JAPE output texts with (any) psycholinguistic score beneath a given threshold.

5.4 Conclusion

Two evaluations were described in this chapter. The first, the exploratory evaluation (described in section 5.2), was conducted on an early prototype of JAPE, JAPE-1. It was meant to qualitatively evaluate the model implemented in JAPE-1, and suggest directions for improvement. The second evaluation was the confirmatory evaluation (described in section 5.3), conducted on the final version of JAPE, JAPE-2. Its purpose was to rigorously evaluate JAPE-2's performance, and measure its success in generating punning riddles. Both evaluations achieved their goals.

The exploratory evaluation confirmed that JAPE-1 is able to generate punning riddles from a humour-independent lexicon, although the quality of the jokes produced is not very high. The comments from the judges were especially helpful in identifying the factors which contribute to less-than-successful JAPE-1 output.

The final evaluation confirmed a number of hypotheses about JAPE-2's performance, although it failed to confirm several others. It was shown that:

- Children consider sensible non-jokes and nonsense non-jokes to be equally 'joke-like'. That is, there is no significant difference in the 'jokiness' scores of the two types of non-jokes.
- Children can distinguish human jokes from non-jokes. That is, the 'jokiness' of the human-generated texts is significantly higher than that of the non-joke texts. This confirms hypothesis 4 in section 5.3.2.
- Children can distinguish JAPE-2 jokes from non-jokes. That is, the 'jokiness' of the JAPE-2 generated texts is significantly higher than that of the non-joke texts. This confirms hypothesis 1 in section 5.3.2.
- The 'jokiness' of the human-generated jokes is significantly higher than that of the JAPE-2 generated texts. This fails to confirm hypothesis 2 in section 5.3.2. However:
  - If the poor or underinformed schemata are eliminated, the difference between the jokiness of the remaining JAPE-2 texts and the human generated jokes is no longer significant.
  - If those texts with low (< 400) psycholinguistic scores are eliminated, the difference between the jokiness of the remaining JAPE-2 texts and the human generated jokes is no longer significant.
- There is no significant difference in funniness between the two types of non-joke.
- Human-generated jokes are significantly funnier than non-jokes.

- JAPE-2 generated jokes are significantly funnier than non-jokes.
- Human-generated jokes are significantly funnier than JAPE-2 jokes.
- Human-generated jokes score significantly higher than JAPE-2 generated texts on three psycholinguistic measures (familiarity, imageability and concreteness) associated with readability.
- There are small but significant correlations between a text's familiarity, imageability and concreteness scores, and its 'jokiness'.
- There are small but significant correlations between a text's familiarity, imageability and concreteness scores, and its 'funniness'.
- There is a small but significant correlation between a child's age (in the range eight to eleven) and his or her ability to recognise human-generated jokes as jokes.

The significance of these results for JAPE will be discussed in chapter 6.

Chapter 6

Discussion

6.1 Introduction

In this chapter, we analyse some of the issues raised by the implementation and evaluation of our model of simple punning riddles. The relevance of this work to the fields of humour studies and AI is discussed, and directions for further research are suggested.

JAPE's final evaluation suggests that, although JAPE does generate jokes, they are neither as funny, nor as 'joke-like', as human-generated punning riddles. These apparent shortcomings come from a variety of sources, including problems with the evaluation (see chapter 5), the implementation (see chapter 4), and the model (see chapter 3). The problem with the largest effect on JAPE's performance is the lack of lexical resources of adequate size and quality (JAPE's current lexical resources are described in section 4.3).

6.2 Evaluation issues

In the confirmatory evaluation of JAPE-2 (see section 5.3), we found that JAPE-2's jokes were significantly more joke-like than non-jokes, but that they were significantly less joke-like than human-generated jokes. Similarly, JAPE-2's jokes were funnier than those non-joke texts that were judged to be jokes, but less funny than the human-generated jokes. We also found a correlation between the 'readability' of the joke and its judged 'jokiness', and between the age of the judge and their ability to recognise human-generated jokes as jokes. Here we discuss whether any of these results were features of the evaluation design, and what the wider significance of the results might be.

6.2.1 Subjects

Based on psychological research into children's humour (e.g. [Yuill and Easton, 1993]), we decided that 8-11 year olds would be an appropriate age group to act as joke judges for this experiment. This age group is mature enough to be able to follow the experimenter's instructions, and has been shown to recognise and appreciate punning riddles [Ruch et al., 1990]. However, the confirmatory evaluation showed a significant correlation between the age of the judge and her or his ability to recognise a human-generated punning riddle as a joke. Also, we also found a correlation between the estimated 'readability' of a joke text and the ability of the judge to recognise the text as a joke.

It is likely that these two results are related. If joke recognition goes up with the readability of the text, and if older children tend to be better readers, then older children should be better able to understand the joke texts, and therefore be better able to recognise them as jokes. This would suggest that a slightly older age group might be appropriate for this kind of study, despite evidence that 8-11 year olds *appreciate* puns most. Our interpretation of these results could be confirmed with further experimentation.

6.2.2 Materials

There were some minor errors in the materials, but it is unlikely that they had a significant effect on the results (see section 5.3.5).

More importantly, the questionnaires did not include a "not sure" option in their yes/no questions. This forced the children to judge whether or not a given text was a joke, even if they were not sure. This may have had a positive effect, however. We suspect that many children, faced with an unfunny joke, would have marked the "not sure" box had it been available — rather than the "yes" option for 'jokiness' and the "not funny at all" option for 'funniness', as we would have wanted them to. Since many of the texts were not very funny jokes, this might have been a serious problem. Not having a "not sure" box forced the children to give a definite answer to "Was that a joke?", and may well have biased the results (for all texts) towards "yes".

Also, no clear instructions were given to the children on how to mark the funniness of texts that were judged to be non-jokes. As a result, a variety of strategies were followed, such as giving the non-jokes the lowest funniness score, or not giving non-jokes a funniness score at all. This inconsistency meant that we were unable to decide whether or not the children thought non-jokes could be funny, which would have been an interesting secondary result.

6.2.3 Filtering

One of the main purposes of the evaluation was to compare jokes generated by JAPE with those generated by humans. However, the human–generated jokes had one significant advantage: they were also *filtered* by humans.

For a joke to be remembered and retold, it must be quite good; for it to be included in an edited collection of jokes, it must be very good (compared to the range of possible jokes). All of the jokes used in this study came from published books of jokes, so must have gone through some sort of filtering process. JAPE's jokes, on the other hand, were minimally filtered (see section 5.3.3) before the evaluation (although the evaluation did suggest some ways in which they might be filtered in the future).

Although we would claim that JAPE's output texts are all well-formed *puns*, they are not all good *jokes*. Unfortunately, an automatic filtering process, to parallel the human filtering described above, would require a system that could *appreciate* humour. Such a system would be very difficult to design, for reasons introduced in section 1.2.3. A more feasible approach would be to collect a set of heuristics that, given well-formed puns, could order them according to expected funniness. Such heuristics could include preferring short jokes to long ones, preferring jokes which use slang or 'rude' words, and preferring jokes which contain accidental (i.e. not required by a schema) alliteration or rhyme (see section 5.2.5). However, such simplistic heuristics are bound to be quite crude, and would probably be theoretically uninteresting.

6.3 The significance of the results

The results of the final evaluation are significant in two main ways. They justify our approach to modelling and generating punning riddles, in that the JAPE-generated texts have been judged to be joke-like; and they suggest several ways in which the model and the system could be improved.

If the output texts and their various successes and failures (see appendix L) are qualitatively compared, two main kinds of failure stand out. Some contain words that the average 8-11 year old is unlikely to know. For example:

> What do you call a lenient shelter? *A lax deduction.*[1] [JAPE]

Not only are "lenient", "lax", "tax shelter" and "deduction" quite difficult vocabulary, but the joke is based on the compound nominal "tax deduction" — a phrase with which most children are unfamiliar.

In other jokes, the words themselves are familiar, but the *sense* of at least one of the words used in the joke is not:

> What kind of curve has cheek? *A nerve ball.*

In this joke, all of the words themselves should be familiar. However, understanding the joke requires that the listener be familiar with the term "curve ball", and also know that "cheek" can mean "nerve" (as in "She has a lot of cheek saying that!"). Most children would not have this knowledge. Moreover, the joke uses a mixture of American and British slang; the listener would have to be familiar with both to get the 'joke'.

Some jokes apparently failed because they simply do not make sense, linguistically. For example, WordNet contains the information that "running away" is a compound nominal, and is a kind of "feat". This led to the 'joke':

> What do you call a clever feat? *Cunning away.* [JAPE]

[1] For those who don't 'get it', JAPE figures that a tax deduction is a kind of shelter, so a lax deduction would be a lenient shelter.

As discussed in section 3.6, JAPE assumes that the last word in a compound nominal is the noun being modified, and the other words are the modifiers. In "running away", this is not the case, resulting in both the constructed phrase ("cunning away") and its description ("a clever feat") not being well-formed linguistically.

Finally, a few poor jokes resulted from schemata not performing as expected. For example:

> How is an ugly insect like a deep kinswoman? *They're both bass aunts.* [JAPE]

This is a positive comparison riddle which uses the phrases "base ant" (ugly insect) and "bass aunt" (deep kinswoman). Unfortunately, the schema that generated this text only gives *one* of the constructed phrases, assuming its homophonous pair will also be brought to mind. A better example of a riddle generated by this schema is:

> How is a nice girl like a chocolate birdie? *They are both sweet chicks.* [JAPE]

This particular problem can be corrected by constraining the **poscomp** schema to use words with two senses, rather than homonyms (which are spelled differently), to construct its jokes.

We would draw three conclusions from the above.

- Jokes rely on close psychological associations between words, often using one word or phrase to bring another word or phrase into the listener's mind. If a joke relies on a word or association which is not familiar to the listener, or which is too weak to bring the target concept to mind, then the joke will probably fail.

  Failures on this account are due to shortcomings in the lexical resources used by JAPE. These are discussed in section 6.3.1, below.

- Joke texts must follow (most of) the linguistic principles governing the syntax and semantics of grammatical texts, *even though the resulting texts may be nonsensical.* That is, semantic constraints derived from the syntax of the text must be satisfied, even though the text may not have a semantic interpretation which makes sense in the 'real world'.

Failures on this account are due to flaws in the model, and are discussed in section 6.3.2, below.

- Jokes are similar to puzzles, in that the mental effort required to 'solve' them is part of their pleasure. However, if a joke is too complex, it may not be understood at all.

  Failures on this account are due to flaws in the schemata, and are discussed in section 6.3.2, below.

The key questions, then, are what improvements are required, and how these improvements should be integrated into the model or its implementation. The following two sections discuss the sources of some of the flaws in JAPE's performance, and suggest possible fixes.

6.3.1 Implementation issues

Joke texts rely on certain associations being present in the listener's mind. For example, the joke:

> What kind of murderer has fibre? *A cereal killer.* [JAPE]

would only be understood by someone who:

- is familiar with the words "murderer", "fibre", "cereal" and "killer", and the phrase "serial killer";
- knows that cereal has fibre and that a serial killer is a kind of murderer.

In other words, the listener must be familiar with the words used in a joke *and* their senses (as used in the joke) for the joke to be understood. Most of the poor JAPE jokes seem to have failed on one or the other of these points. Although both problems can be fixed to some extent, it is unlikely that both can be fixed completely; after all, human joke tellers often misjudge the knowledge of their audience, resulting in a failed joke.

To ensure that the listener could understand all the words in a given joke, we would have to know her or his entire vocabulary. We cannot assume that the meanings of some words can be picked up from the context (as we could in non-joke texts), since words in jokes are often presented out of their usual context. As an approximation, we could use the measures from the MRC psycholinguistic database to estimate the readability (i.e. likelihood that all the words will be familiar to the reader) of a joke text. This approximation will fail sometimes, but the correlation found between the 'readability' scores for a joke text and its judged 'jokiness' suggests that it is a usable approach.

The second problem is more difficult to solve, since it relates to the *sense* of the word. The sense of a word consists of the relations between the word and other words. For example, the listener may be familiar with both the word "fare" and the word "menu", but if they do not know that "fare" can refer to a menu in a restaurant — that is, that the relation synonym(fare, menu) holds — then the following joke would not make sense:

> What do you call a reasonable menu? *A fair fare.* [JAPE]

Unfortunately, no currently available resources give psycholinguistic data about the senses of words — that is, data with which we could, for example, compare the familiarity of "bear" the investor and "bear" the animal — on a large scale. It is possible that a very large knowledge base such as CYC [Lenat, 1995] might be used for this purpose.

Another important question is *how* familiar a key word and its sense must be for a joke to be understood and appreciated. If the listener knows that "fare" can be used to mean "menu", but would never use it that way herself, perhaps a joke based on this association would fail. On the other hand, it is possible that *too* close an association in the mind of the listener would take away the surprise element of the joke. Further experimentation is required to resolve this issue.

6.3.2 Model issues

The results of the final evaluation showed that some schemata produced more successful jokes than others (see section 5.3.6). There are several possible reasons for a schema to perform poorly:

- It could be over-constrained, so that no jokes of a sufficient readability can be generated.
- It could be under-constrained, and not capture the key features of the type of pun it was designed to produce.
- It could require linguistic information not contained in the available resources to perform well.
- It could violate the linguistic constraints required for the joke to "make sense".
- It could be over complex, producing puns that are too difficult to unravel.

In JAPE-2, several schemata, particularly the **coatshed**, **hopchew**, **negcomp**, and **poscomp** schemata (see appendix J), were highly constrained. These produced very few jokes. However, those jokes that were produced by these schemata were, for the most part, of good quality (see figure 5.3), so over-constrained schemata do not seem to be a problem.

The least constrained schema is the **bazaar** schema, which produces jokes like:

> What do you call a strange market? *A bizarre bazaar.* [JAPE]

Many jokes of this type were produced and, on average, they had a 'jokiness' score of approximately 0.56, which is just below the overall average for JAPE-generated jokes. Jokes using the juxtaposition mechanism (see section 3.2.5), which the **bazaar** schema is based on, are generally considered to be quite weak; perhaps this weakness is in part a result of the joke type being under-constrained.

Several schemata need linguistic resources not available in the current system. The **woolly** and **jumper** schemata, for example, need to be able to find the lexemes which make up a lexicalized compound noun; for example, that the phrase "train fare" is made up of the words "train" (with the sense of locomotive) and "fare" (with the sense of price charged for a ticket). This information is not available in WordNet [Miller et al., 1990]. As implemented, the WordNet interface randomly chooses senses to use for the words which make up a phrase to be used in a joke. This is not as catastrophic as it might sound, since an incorrect choice would simply lead to a more complex pun than intended; however, these accidental complex puns are often much more difficult to understand than the puns these schemata were meant to model. For example, WordNet contains a sense of "bear" which is "an investor with a pessimistic market outlook", and has the hypernym "investor". If JAPE decides that this sense of "bear" is meant by the second word in the phrase "grizzly bear", then the **woolly** schema could generate the text:

> What do you call a gruesome investor? *A grisly bear.* [JAPE]

Although this is still a joke of sorts, it is a more complex pun than intended, in that both the words in the phrase have been substituted — "grizzly" with its homophone "grisly", and "bear" (the animal) with "bear" (the investor).

It is possible that some jokes failed because they violated certain linguistic constraints required for the joke to "make sense". The most obvious constraint — that the constructed meaning for a constructed phrase should be such that the head of the phrase is modified by the other words in the phrase — was suggested by the exploratory evaluation, and implemented in JAPE-2 before the final evaluation. Few such violated constraints are obvious in the failed JAPE-2 jokes (aside from the "cunning away" example discussed above), however, so a more sophisticated set of experiments would be required to determine if jokes not "making sense" is still a problem.

Overall, it would seem that the main problem with the schemata is that some require linguistic information not available in the set of linguistic resources currently used by JAPE.

6.4 Relevance to other work

In this section, we discuss how JAPE's performance could be improved by more sophisticated knowledge resources, and how our results might influence other work, both in AI and in humour studies.

6.4.1 The knowledge resources

Using linguistic resources such as WordNet [Miller et al., 1990], the MRC psycholinguistic database [Wilson, 1987], the British English Example Pronunciation dictionary [Robinson, 1996], and the homonym list [Townsend and Antworth, 1993], we have been able to generate large numbers of simple punning riddles. However, the evaluation of JAPE-2 suggests that, in order to generate riddles of the same quality as those generated by humans, more sophisticated resources are required. In particular, the system would need:

- A wider range of semantic links between words, especially between words of different syntactic categories. For example, a link between a object and an action that object is likely to perform (e.g. between "bomb" and "explode") would greatly enrich the associations which could be used in jokes.
- Psycholinguistic information relating to the familiarity of particular senses of words, so that comprehensible jokes can be constructed based on associations that the listener is likely to know.

We believe that large knowledge bases containing such information would be useful for other applications in natural language research as well.

Even better jokes could be constructed with a knowledge base that contained associations not likely to be found in a dictionary; for example, associations between a type of person and their stereotypical traits (e.g. between construction workers and whistling at women). Many jokes are founded on such cultural associations, but a system which only has access to linguistic information would be unable to generate such jokes.

6.4.2 Relation with the General Theory of Verbal Humour

Although the General Theory of Verbal Humour (GTVH) [Ruch et al., 1993] is too general to be tested experimentally, it is also the main linguistic theory of humour currently available. For this reason, it would be useful, for the sake of comparison, to be able to show that our model is consistent with the GTVH. A summary of the GTVH is given in section 2.3.1.

Our model could be seen as a shell produced by constraining or holding constant some of Attardo and Raskin's parameters. In terms of their six knowledge resources, our model could be described as follows:

Language The surface form of the joke is determined by the template. The choice of template is constrained by the choice of schema, and the instantiation of the schema provides the template with information it needs to construct the surface form.

Narrative Strategy The narrative strategy is fixed, since these jokes are all, by definition, question-answer riddles.

Target There is no explicit target in any of the set of aimed-for riddles, although it is possible that some object of ridicule might be accidentally incorporated. If the lexicon were adjusted to include stereotypical (instead of typical) semantic links, then jokes with butts would probably arise more often.

Logical Mechanism The logical mechanisms used by our riddles are juxtaposition, substitution and comparison. The different (uninstantiated) schemata could be seen as slight variations on these basic mechanisms.

Situation The situation is variable, though 'choosing the situation' really means choosing words/concepts which are consistent with the logical mechanism (i.e. instantiating the schema).

Script Opposition If we see a lexical entry for a lexeme as a very simple kind of script, then it could be argued that scripts are being 'opposed' via the mechanisms of substitution, juxtaposition and comparison (see section 3.2.5). It is unclear *which* of Raskin's opposition types might be taking place — real vs. unreal perhaps.

The model described in chapter 3, then, is consistent with Attardo and Raskin's General Theory of Verbal Humour (GVTH), although most of their parameters are either fixed or constrained. The features of our model correspond to three GTVH parameters: an uninstantiated schema corresponds to a particular *logical mechanism*; instantiating the schema results in a specific *situation*; and filling a template produces the surface-form *language* of a joke.

Our model is consistent with the GTVH; however, it does not emphasize the same features of joke structure. In particular, Attardo and Raskin hold that script opposition is absolutely essential for a piece of text to be considered a joke; whereas it is not even represented in our model. Either script opposition is not actually very important in question-answer punning riddles, or one script opposition is implicit in and fixed for all such jokes. Moreover, the GTVH is so general that it could be consistent with any number of more specific models — the success of one, ours, does not mean that the GTVH has been vindicated.

6.4.3 General theories vs. micro-models

As suggested in the previous section, some humour researchers have developed very general theories of humour, whereas others (such as ourselves) have concentrated on building testable micro-models of particular types of joke. Ideally, the general theories provide a framework for more specific investigation, and the testable micro-models inform and constrain (by providing experimental results and details) the more general theories. However, at present, the gulf between these two methodological approaches is too wide for the two kinds of research to be combined in this way. The general theories do not predict any particular phenomena that a micro-model might represent and test; and even if a micro-model is consistent with some general theory, its success or failure does not say much about the validity of that theory.

Solving this problem requires some work by both sides. General theorists should make their theories testable, by making specific predictions about the ramifications of their ideas. Likewise, micro-modellers should try to make their model consistent with a general theory (as we have attempted to do in the previous section), so that the success or failure of their model says something about the general theory. Until this happens, the linguistics of humour will be little more than untestable theories and taxonomies of specific phenomena.

6.5 Further work

There are several different directions that this research suggests should be explored. Some are more speculative than others.

As discussed in section 6.4.1, the quality and range of the jokes generated by JAPE could be greatly improved by attaching the system to a larger, more general knowledge base than the lexical resources currently used. One reason punning riddles were chosen as the genre to be modelled in this work was that large lexicons are commonly available in computer-readable form. If a more general knowledge base were available, it is possible that quite complex linguistic humour, or even non-linguistic humour, might be generated.

As discussed in section 6.3.2, many of JAPE's failures were due to inadequate psycholinguistic information about the words and their senses being available, leading to misjudgements about what associations were available to the listener. This problem could be finessed by building a system for *human-assisted* pun generation. Such a system would work much the same way JAPE does, but it would not rely on its lexical resources for good word-word associations. Instead, it would prompt the user for typical associations; for example, it could ask the user what a "bomb" typically does (hopefully getting the answer "explode"), rather than rely on its lexicon to provide this information, which it may not be able to do.

The genre of pun researched here is punning riddles. The riddle form was chosen in part because of its simplicity, regular structures, and commonness; however, there are other common genres of pun which use the same basic mechanisms. One promising type is the 'story pun', in which a fable-like story ends with a punning moral (usually based on a common proverb). For example, there is a well-established proverb in the

English language *"People who live in glass houses shouldn't throw stones"* (meaning that those who are vulnerable to a particular form of criticism should not themselves make that criticism of others). This has been used as the basis of the following joke:

> Once upon a time, many years ago, there was a chieftain in a remote tropical village who owned an old and battered throne of which he was very fond. One day, a visiting dignitary gave him a brand new and ornate throne, which the chieftain had to adopt immediately out of politeness. However, he could not bear to part with the old throne which had served him so well, so he stowed it away in the roof area of his grass hut, in case it should be useful in the future. Unfortunately the interior structure of his hut was too flimsy to support the weight of the large object, and it crashed through the grass ceiling, falling on the chieftain and killing him.
>
> The moral is that people who live in grass houses shouldn't stow thrones.

Although the punning mechanism used in story puns is very similar to that used in punning riddles, the stories are much more complex than the simple question-answer forms used in riddles. Nonetheless, it is possible that such stories could be modelled and generated by computer (see [Binsted and Ritchie, 1996] for more speculations on this subject).

Finally, there has been some interest from the software agents research community in the possibility of integrating humour generation into a natural language using agent. Some headway has been made on this (e.g. [Loehr, 1996]), but much more research is required before the use of humour by agents is fluent enough to have the desired effect. A necessary step would seem to be to modify JAPE so that its mechanisms could be used by a more general natural language generation system.

6.6 Conclusion

JAPE has successfully demonstrated that it can generate punning riddles. Most are good examples of the genre, although some are not very funny, and a few fail altogether. Even these failures are interesting, as they indicate some fixable weaknesses in our model and its implementation. Most of the less successful jokes were due to weaknesses in the knowledge bases available to JAPE, resulting in jokes which may be incomprehensible to their intended audience.

The model implemented in JAPE is consistent with more general theories of humour; however, these theories are so general that JAPE's performance says very little about their validity.

Some further work is suggested, including expanding the model to cover more types of linguistic and non-linguistic humour, using richer knowledge bases, and integrating JAPE into more general natural language using systems.

Overall, JAPE's successful generation of punning riddles is evidence that our model captures the essential features of the genre.

Chapter 7

Conclusion

In the introduction to this thesis, we described our research goals. They were:

- To develop a falsifiable, formal and implementable model of punning riddles, in such a way that their key features and mechanisms are captured (see chapter 3);
- To implement that model in a program which generates punning riddles, and *only* punning riddles (see chapter 4);
- To evaluate the performance of that program, by comparing the reaction of human joke judges to both the program's output and human-generated jokes (see chapter 5); and,
- To draw conclusions, based on the performance of the program, about the nature of this subtype of humour (see chapter 6).

Here, we discuss whether or not those goals have been achieved.

The model we developed is indeed falsifiable, formal and implementable, as demonstrated by the program, JAPE, in which it has been implemented and evaluated. Whether or not it has captured the key features and mechanisms of punning riddles is another question. The model we developed consists of four main parts: the schemata, which specify the relations between the lexical items used to build a joke; the SAD generator, which generates small adequate descriptions of real or constructed lexical items; the templates, which turn lexical items and their descriptions into question–answer pairs; and the lexical resources, which provide all the lexical information required for the other parts to work. Of these, all but the first are humour–independent, and represent linguistic competences that any ‘pun generator’ can be expected to have.

The schemata, on the other hand, use the mechanisms of substitution, juxtaposition and comparison to build the essential elements of a pun. The choice of these mechanisms was not arbitrary, but was based on an analysis of a large number of human–generated riddles. The fact that such a small number of schemata could describe such a wide range of punning riddles suggests that we have, indeed, captured at least some of the key features of the genre.

The second research goal was to implement the model in a program, JAPE. This we did. The schemata, templates and SAD generator were unproblematic to implement in Prolog. Finding appropriate lexical resources was more difficult. Initially, we used a hand–built lexicon with definitions by volunteers, but eventually adequate lexical resources (in the form of WordNet [Miller et al., 1990], BEEP [Robinson, 1996], the homonym list [Townsend and Antworth, 1993], and the MRC psycholinguistic database [Wilson, 1987]) were found.

Whether or not the output of JAPE consisted only of punning riddles could only be tested by experiment. In our final confirmatory evaluation, 8-11 year old children evaluated JAPE output in comparison with human–generated jokes, sensible non-jokes, and nonsense non-jokes. The results showed that the children could easily distinguish both JAPE and human jokes from the non-jokes, although JAPE's were not as ‘joke-like’ as human-generated jokes. However, when the JAPE jokes with low ‘readability’ scores, based on psycholinguistic data, were removed, the JAPE jokes were indistinguishable from the human–generated jokes. This was quite an accomplishment, considering that the human jokes used came from published joke books, and thus have been through a considerable filtering process to weed out the non-jokes.

Based on the evaluation, we were able to draw several conclusions about our model, its implementation, and puns in general. We found that most of the poor JAPE jokes were due to one of three reasons: words or word senses that were probably unfamiliar to the young audience; joke texts violating grammatical constraints at some level; or overly complex pun structures that were very difficult to ‘solve’. Some of these can be overcome through modifications to JAPE; however, JAPE's performance would be most improved by the addition of a knowledge base including more sophisticated information about words and the relations between them.

So, we have a working implementation of a model of punning riddles that generates jokes of that genre, and that reveals some key features of puns in general. Simple puns, at least, can be modelled formally, and can be generated by a program.

Bibliography

[Abbott, 1993] Abbott, S. (1993). *The Cat Joke Book.* Random House Children's Books, London.

[Alec, 1987] Alec, S. (1987). *Smart Alec's Spooky Jokes for Kids.* Ward Lock Limited.

[Anderson, 1987] Anderson, S. (1987). *A-Z of Animal Jokes.* Young Corgi Books.

[Attardo, 1994] Attardo, S. (1994). *Linguistic Theories of Humour.* Mouton de Gruyter, Berlin.

[Attardo and Raskin, 1991] Attardo, S. and Raskin, V. (1991). Script theory revis(it)ed: joke similarity and joke representation model. *Humor*, 4(3):293–347.

[Attardo and Raskin, 1994] Attardo, S. and Raskin, V. (1994). Non-literalness and non-bona-fide in language: Approaches to formal and computational treatments of humor. *Pragmatics and Cognition*, 2(1):31–69.

[Barsalou, 1982] Barsalou, L. W. (1982). Context-independent and context-dependent information in concepts. *Memory and Cognition*, 10:82–93.

[Barsoux, 1994] Barsoux, J.-L. (1994). *Funny Business.* Cassell.

[Binsted, 1995] Binsted, K. (1995). Using humour to make natural language interfaces more friendly. In Kitano, H., editor, *Proceedings of the IJCAI workshop on AI and Entertainment.*

[Binsted and Ritchie, 1994] Binsted, K. and Ritchie, G. (1994). An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, USA. Available as Research Paper 690, Department of Artificial Intelligence, University of Edinburgh.

[Binsted and Ritchie, 1996] Binsted, K. and Ritchie, G. (1996). Speculations on story puns. In *Proceedings of the International Workshop on Computational Humour*, Enschede, Netherlands.

[Binsted and Ritchie, 1997] Binsted, K. and Ritchie, G. (1997). Computational rules for generating punning riddles. *Humor.* To appear.

[Boden, 1991] Boden, M. A. (1991). *The creative mind: myths and mechanisms.* Weidenfeld and Nicholson.

[Booth, 1974] Booth, W. (1974). *A Rhetoric of Irony.* University of Chicago Press.

[Brandreth, 1990] Brandreth, G. (1990). *The Teddy Bear Joke Book*. Armada.

[Buchanan, 1988] Buchanan, B. G. (1988). Artificial intelligence as an experimental science. In Fetzer, J., editor, *Aspects of Artificial Intelligence*, pages 209–250. Kluwer.

[Byrne, 1995] Byrne, J. (1995). *Mirthful Kombat - Jokes with BYTE!* Red Fox.

[Chapman and Foot, 1976] Chapman, A. and Foot, H., editors (1976). *Humour and Laughter: Theory, Research and Applications*. John Wiley and Son, London.

[Chiaro, 1992] Chiaro, D. (1992). *The language of jokes: analysing verbal play*. Routledge, London.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

[Chomsky, 1965] Chomsky, N. (1965). *Aspects of the theory of Syntax*. MIT Press, Cambridge, Mass.

[Churchill, 1976] Churchill, E. R. (1976). *The Six-Million-Dollar Cucumber*. Piccolo Pan Books.

[Coltheart, 1981] Coltheart, M. (1981). The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(A):497–505.

[Curcó, 1996] Curcó, C. (1996). The implicit expression of attitudes, mutual manifestness, and verbal humour. UCL Working Paper Vol 8, University College London.

[Ephratt, 1990] Ephratt, M. (1990). What's in a joke. In Golumbic, M., editor, *Advances in AI: Natural Language and Knowledge Based Systems*, pages 43–74. Springer Verlag.

[Ertner, 1993] Ertner, J. D. (1993). *Super Silly Animal Riddles*. Sterling Publishing Co. Inc., New York.

[Forrester, 1994] Forrester, M. (1994). *The Vampire Joke Book*. Puffin Books.

[Freedman and Hofman, 1980] Freedman, M. and Hofman, P. (1980). *How Many Zen Buddhists Does It Take to Screw in a Light Bulb?* St Martin's.

[Fremont, 1993] Fremont, E. (1993). *Cackles from the Crypt*. Random House Children's Books.

[Freud, 1976] Freud, S. (1976). *Jokes and their relation to the unconscious*. Harmondsworth Penguin.

[Gilhooly and Logie, 1980] Gilhooly, K. J. and Logie, R. H. (1980). Age of acquisition,imagery, concreteness, familiarity and ambiguity measures for 1944 words. *Behaviour Research Methods and Instrumentation*, 12:395–427.

[Giora, 1988] Giora, R. (1988). On the informativeness requirement. *Journal of Pragmatics*, 12(5/6):547–565.

[Giora, 1991] Giora, R. (1991). On the cognitive aspects of the joke. *Journal of Pragmatics*, 16:465–485.

[Girling, 1988a] Girling, B., editor (1988a). *The Schoolkids' Joke Book.* Armada.

[Girling, 1988b] Girling, B. (1988b). *The Schoolkids' Joke Book.* Armada.

[Greene and D'Oliveira, 1992] Greene, J. and D'Oliveira, M. (1992). *Learning to use statistical tests in psychology.* Open University Press.

[Grice, 1975] Grice, H. P. (1975). Logic and conversation. In Cole, P., editor, *Syntax and Semantics Volume 3: Speech Acts*, pages 41–58.

[Hegarty, 1992] Hegarty, J. (1992). *The Upside Down Joke Book.* Random Century Children's Books.

[Hofstadter, 1995] Hofstadter, D. (1995). *Fluid concepts and creative analogies.* Basic Books.

[Hofstadter et al., 1989] Hofstadter, D., Gabora, L., Raskin, V., and Attardo, S. (1989). Synopsis of the workshop on humor and cognition. *Humor*, 2-4.

[Hulstijn and Nijholt, 1996] Hulstijn, J. and Nijholt, A., editors (1996). *Automatic interpretation and generation of verbal humour (IWCH'96).* University of Twente.

[Jam, 1991] Jam, E. (1991). *The Raspberry Joke Book.* Knight Books.

[Katz, 1993] Katz, B. F. (1993). A neural resolution of the incongruity-resolution and incongruity theories of humour. *Connection Science*, 5(1):59–75.

[Katz, 1994] Katz, B. F. (1994). Towards a unified theory of humour. CSRP 356, University of Sussex.

[Lakoff, 1986] Lakoff, G. (1986). *Women, Fire and Dangerous Things: What Categories Tell Us About the Nature of Thought.* University of Chicago Press, Chicago.

[Lenat, 1995] Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11).

[Loehr, 1996] Loehr, D. (1996). An integration of a pun generator with a natural language robot. In *Proceedings of The Student Conference in Computational Linguistics*, Montreal.

[Mark and Greer, 1993] Mark, M. A. and Greer, J. E. (1993). Evaluating methodologies for intelligent tutoring systems. *AI and Education: Special Issue on Evaluation*, 4(2/3):129–153.

[Metcalf, 1994] Metcalf, F., editor (1994). *The Penguin Dictionary of Jokes.* Penguin Books.

[Miller et al., 1990] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K., and Tengi, R. (1990). Five papers on WordNet. *International Journal of Lexicography*, 3(4). Revised March 1993.

[Minsky, 1963] Minsky, M. (1963). Steps towards artificial intelligence. In Feigenbaum, E. and Feldman, J., editors, *Computers and Thought*, pages 406–450. McGraw-Hill.

[Minsky, 1980] Minsky, M. (1980). Jokes and the logic of the cognitive unconscious. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[Minsky, 1986] Minsky, M. (1986). *The society of mind.* Simon and Schuster, New York.

[Newell and Simon, 1976] Newell, A. and Simon, H. (1976). Computer science as an empirical inquiry: symbols and search. *Communications of the ACM*, 19:113–126.

[Paulos, 1980] Paulos, J. (1980). *Mathematics and Humour.* University of Chicago Press.

[Pepicello and Green, 1984] Pepicello, W. and Green, T. A. (1984). *The Language of Riddles.* Ohio State University.

[Phillips, 1991] Phillips, L. (1991). *Wackysaurus Dinosaur Joke Book.* Puffin Books.

[Prince, 1981] Prince, E. (1981). Toward a taxonomy of given-new information. In Cole, P., editor, *Radical Pragmatics*, pages 223–255. Academic Press, New York.

[Raskin, 1985] Raskin, V. (1985). *The Semantic Mechanisms of Humour.* Dordrecht Reidel.

[Rayner, 1991] Rayner, S. (1991). *Ready Teddy Go Joke Book.* Puffin Books.

[Ritchie, 1994] Ritchie, G. (1994). Learning from AM. In Johnson, J., McKee, S., and Vella, A., editors, *Artificial Intelligence in Mathematics*, pages 55–66. Oxford University Press, Oxford.

[Robinson, 1996] Robinson, T. (1996). The British English example pronounciation dictionary.

[Ruch et al., 1993] Ruch, W., Attardo, S., and Raskin, V. (1993). Toward an empirical verification of the general theory of verbal humour. *Humor*, 6(2):123–136.

[Ruch et al., 1990] Ruch, W., McGhee, P., and Hehl, F.-J. (1990). Age differences in the enjoyment of incongruity-resolution and nonsense humour during adulthood. *Psychology and Aging*, 5:348–355.

[Sansome, 1982] Sansome, R. (1982). *The Puffin Junior Dictionary.* Puffin Books.

[Schubert, 1986] Schubert, L. K. (1986). Are there preference trade-offs in attachment decisions? In *Proceedings of AAAI'86*, pages 601–605.

[Siegel and Jr, 1988] Siegel, S. and Jr, N. J. C. (1988). *Nonparametric statistics for the behavioral sciences.* McGraw-Hill Book Company.

[Sperber and Wilson, 1986] Sperber, D. and Wilson, D. (1986). *Relevance: Communcation and cognition.* Blackwell, Oxford.

[Takizawa, 1991] Takizawa, O. (1991). DUJAL - a spoken joke understanding system aimed at advanced speech recognition. In *Proceedings of the Korea-Japan Joint Symposium on Acoustics.*

[Takizawa, 1993] Takizawa, O. (1993). Characteristics of some sorts of connotations in spoken language. *Journal of the Communications Research Laboratory*, pages 171 – 178.

[Toglia and Battig, 1978] Toglia, M. P. and Battig, W. R. (1978). *Handbook of Semantic Word Norms.* Erlbaum, New York.

[Townsend and Antworth, 1993] Townsend, W. and Antworth, E. (1993). *Handbook of Homophones (online version).*

[Webb, 1978] Webb, K., editor (1978). *The Crack-a-Joke Book.* Puffin, London.

[Weiner and de Palma, 1993] Weiner, E. J. and de Palma, P. (1993). Some pragmatic features of lexical ambiguity and simple riddles. *Language and Communication*, 13(3):183–193.

[Wilson and Sperber, 1992] Wilson, D. and Sperber, D. (1992). On verbal irony. *Lingua*, 87:53–76.

[Wilson, 1987] Wilson, M. (1987). The MRC psycholinguistic database. Technical report, Informatics Division, Science and Engineering Research Council, Rutherford Appleton Laboratory, Oxford.

[Young, 1993] Young, F. (1993). *Super-Duper Jokes.* Farrar, Straus and Giroux.

[Yuill and Easton, 1993] Yuill, N. and Easton, K. (1993). The role of linguistic ambiguity in understanding and improving children's text comprehension. CSRP 296, University of Sussex.

Appendix A

BEEP phoneme set

ARPAbet	MRPA	Edin.	Alvey	Example	Relative frequency (percent)
p	p	p	p	put	3.1
b	b	b	b	but	2.3
t	t	t	t	ten	6.8
d	d	d	d	den	4.1
k	k	k	k	can	4.7
m	m	m	m	man	3.1
n	n	n	n	not	6.5
l	l	l	l	like	5.5
r	r	r	r	run	5.4
f	f	f	f	full	1.8
v	v	v	v	very	1.2
s	s	s	s	some	6.6
z	z	z	z	zeal	3.6
hh	h	h	h	hat	0.8
w	w	w	w	went	0.9
g	g	g	g	game	1.3
ch	ch	ch	tS	chain	0.5
jh	jh	j	dZ	Jane	0.8
ng	ng	ng	9	long	1.6
th	th	th	T	thin	0.3
dh	dh	dh	D	then	12.2
sh	sh	sh	S	ship	1.2
zh	zh	zh	Z	measure	0.1
y	y	y	j	yes	0.8
iy	ii	ee	i	bean	1.4
cont...					

ARPAbet	MRPA	Edin.	Alvey	Example	Relative frequency (percent)
aa	aa	ar	A	barn	0.9
ao	oo	aw	O	born	1.0
uw	uu	uu	u	boon	1.0
er	@@	er	3	burn	0.7
ih	i	i	I	pit	10.0
eh	e	e	e	pet	2.4
ae	a	aa	&	pat	2.5
ah	uh	u	V	putt	1.5
oh	o	o	0	pot	1.6
uh	u	oo	U	good	0.4
ax	@	a	@	about	7.2
ey	ei	ai	eI	bay	2.0
ay	ai	ie	aI	buy	1.6
oy	oi	oi	oI	boy	0.2
ow	ou	oa	@U	no	1.5
aw	au	ou	aU	now	0.4
ia	i@	eer	I@	peer	0.7
ea	e@	air	e@	pair	0.2
ua	u@	oor	U@	poor	0.2

Appendix B

Phoneme–grapheme translation

These are very simple heuristic matchings between some consonant phonemes and graphemes. Only those cases in which the grapheme is almost certain to map onto the phoneme have been included. See section 4.3.5 for discussion.

b $\longleftrightarrow$ [b]	b $\longleftrightarrow$ [b,b]	ch $\longleftrightarrow$ [c,h]
ch $\longleftrightarrow$ [t,c,h]	d $\longleftrightarrow$ [d]	d $\longleftrightarrow$ [d,d]
dh $\longleftrightarrow$ [t,h]	f $\longleftrightarrow$ [f]	f $\longleftrightarrow$ [f,f]
f $\longleftrightarrow$ [p,h]	g $\longleftrightarrow$ [g]	g $\longleftrightarrow$ [g,g]
hh $\longleftrightarrow$ [h]	jh $\longleftrightarrow$ [j]	jh $\longleftrightarrow$ [g]
k $\longleftrightarrow$ [c,k]	k $\longleftrightarrow$ [k]	k $\longleftrightarrow$ [k,k]
k $\longleftrightarrow$ [c]	l $\longleftrightarrow$ [l]	l $\longleftrightarrow$ [l,l]
m $\longleftrightarrow$ [m]	m $\longleftrightarrow$ [m,m]	m $\longleftrightarrow$ [mb]
n $\longleftrightarrow$ [n]	n $\longleftrightarrow$ [n,n]	n $\longleftrightarrow$ [k,n]
ng $\longleftrightarrow$ [n]	ng $\longleftrightarrow$ [n,g]	p $\longleftrightarrow$ [p]
p $\longleftrightarrow$ [p,p]	r $\longleftrightarrow$ [r]	r $\longleftrightarrow$ [r,r]
s $\longleftrightarrow$ [s]	s $\longleftrightarrow$ [s,s]	s $\longleftrightarrow$ [c]
sh $\longleftrightarrow$ [s,h]	sh $\longleftrightarrow$ [t,i]	sh $\longleftrightarrow$ [s]
t $\longleftrightarrow$ [t]	t $\longleftrightarrow$ [t,t]	th $\longleftrightarrow$ [t,h]
v $\longleftrightarrow$ [v]	w $\longleftrightarrow$ [w]	y $\longleftrightarrow$ [y]
z $\longleftrightarrow$ [z]	z $\longleftrightarrow$ [s]	z $\longleftrightarrow$ [z,z]
zh $\longleftrightarrow$ [z]	zh $\longleftrightarrow$ [s]	zh $\longleftrightarrow$ [z,z]

Appendix C

JAPE-2 templates

The following are the templates implemented in JAPE-2 and the summative evaluation. Templates link a USAD and a suitable sentence form. See section 3.7 for a description of what templates are and what they are meant to do[1], and appendix D for the sentence forms they link to.

```
{Relations:
  describes(NPStr, {cross_between(_,X,Y)},
 SF:
  cross(X,Y,NPStr)}

{Relations:
  describes(NPStr, {spec_is(NP, X), class(NP, Y)}
 SF:
  spec_class(X,Y,NPStr)}

{Relations:
  describes(NPStr, {class(NP, X), has(NP, Y)})
 SF:
  class_has(X, Y, NPStr)}

{Relations:
  describes(NPStr, {class(NP, X), act_verb(NP, Y)})
 SF:
  act_verb(X,Y,NPStr)}

{Relations:
  describes(NPStr, {class(NP, X), inact_verb(NP, Y)})
 SF:
  inact_verb(X,Y,NPStr)}
```

[1] Here we follow the Prolog convention of '_' being a wildcard variable.

```
{Relations:
  describes(NP1Str, {spec_is(NP1, Spec1), class(NP1, Class1)})
  describes(NP2Str, {spec_is(NP2, Spec2), class(NP2, Class2)})
 SF:
  negcompare([Spec1, Class1], [Spec2, Class2], NP1Str, NP2Str)}

{Relations:
  describes(NP1Str, {synonym(_, Syn1)})
  describes(NP2Str, {spec_is(NP2, Spec2), class(NP2, Class2)})
 SF:
  negcompare([Syn1], [Spec2, Class2], NP1Str, NP2Str)}

{Relations:
  describes(NP1Str, {spec_is(NP1, Spec1), class(NP1, Class1)})
  describes(NP2Str, {synonym(_, Syn2)})
 SF:
  negcompare([Spec1, Class1], [Syn2], NP1Str, NP2Str)}

{Relations:
  describes(NP1Str, {synonym(_, Syn1)})
  describes(NP2Str, {synonym(_, Syn2)})
 SF:
  negcompare([Syn1], [Syn2], NP1Str, NP2Str}

{Relations:
  describes(NP1Str, {spec_is(NP1, Spec1), class(NP1, Class1)})
  describes(NP1Str, {spec_is(NP2, Spec2), class(NP2, Class2)})
 SF:
  poscompare([Spec1, Class1], [Spec2, Class2], NP1Str)}

{Relations:
  describes(NP1Str, {synonym(_, Syn1)})
  describes(NP1Str, {spec_is(NP2, Spec2), class(NP2, Class2)})
 SF:
  poscompare([Syn1], [Spec2, Class2], NP1Str)}

{Relations:
  describes(NP1Str, {synonym(_, Syn1)})
  describes(NP1Str, {synonym(_, Syn2)})
 SF:
  poscompare([Syn1], [Syn2], NP1Str)}
```

```
{Relations:
  describes(NP1Str, {spec_is(NP1, Spec1), class(NP1, Class1)})
  describes(NP1Str, {synonym(_, Syn2)})
 SF:
  poscompare([Spec1, Class1], [Syn2], NP1Str)}

{Relations:
  describes_same({spec_is(NP1, Spec1), class(NP1, Class1)},
                 {act_verb(NP1, Verb11), act_verb(NP1, Verb12)})
  describes_same({spec_is(NP2, Spec2), class(NP2, Class2)},
                 {act_verb(NP2, Verb21), act_verb(NP2, Verb22)})
 SF:
  vvcompare([Spec1, Class1], [Spec2, Class2], Verb11, Verb12,
            Verb21, Verb22)}

{Relations:
  describes_same({synonym(NP1, Syn1)},
                 {act_verb(NP1, Verb11), act_verb(NP1, Verb12)})
  describes_same({spec_is(NP2, Spec2), class(NP2, Class2)},
                 {act_verb(NP2, Verb21), act_verb(NP2, Verb22)})
 SF:
  vvcompare([Syn1], [Spec2, Class2], Verb11, Verb12, Verb21, Verb22)}

{Relations:
  describes_same({spec_is(NP1, Spec1), class(NP1, Class1)},
                 {act_verb(NP1, Verb11), act_verb(NP1, Verb12)})
  describes_same({synonym(NP2, Syn2)},
                 {act_verb(NP2, Verb21), act_verb(NP2, Verb22)})
 SF:
  vvcompare([Spec1, Class1], [Syn2], Verb11, Verb12, Verb21, Verb22)}

{Relations:
  describes_same({synonym(NP1, Syn1)},
                 {act_verb(NP1, Verb11), act_verb(NP1, Verb12)})
  describes_same({synonym(NP2, Syn2)},
                 {act_verb(NP2, Verb21), act_verb(NP2, Verb22)})
 SF:
  vvcompare([Syn1], [Syn2], Verb11, Verb12, Verb21, Verb22)}

{Relations:
  describes_same({spec_is(NP1, Spec1), class(NP1, Class1)},
                 {inact_verb(NP1, Verb1)})
  describes_diff({spec_is(NP2, Spec2), class(NP2, Class2)},
                 {inact_verb(NP2,Verb2)})
 SF:
  vncompare([Spec1, Class1], [Spec2, Class2], [NP1], Verb1,
            [NP2], Verb2)}
```

```
{Relations:
  describes_same({synonym(NP1, Syn1)},
                 {inact_verb(NP1, Verb1)})
  describes_diff({spec_is(NP2, Spec2), class(NP2, Class2)},
                 {inact_verb(NP2,Verb2)})
 SF:
  vncompare([Syn1], [Spec2, Class2], [NP1], Verb1, [NP2], Verb2)}

{Relations:
  describes_same({spec_is(NP1, Spec1), class(NP1, Class1)},
                 {inact_verb(NP1, Verb1)})
  describes_diff({synonym(NP2, Syn2)},
                 {inact_verb(NP2,Verb2)})
 SF:
  vncompare([Spec1, Class1], [Syn2], [NP1], Verb1, [NP2], Verb2)}

{Relations:
  describes_same({synonym(NP1, Syn1)},
                 {inact_verb(NP1, Verb1)})
  describes_diff({synonym(NP2, Syn2)},
                 {inact_verb(NP2,Verb2)})
 SF:
  vncompare([Syn1], [Syn2], [NP1], Verb1, [NP2], Verb2)}
```

Appendix D

JAPE-2 sentence forms

These are the sentence forms implemented in JAPE-2. See section 3.7 for a description of their function.

- cross(L1, L2, Str) →
  What do you get when you cross np(L1) and np(L2)? Str.
- spec_class(L1, L2, Str) →
  What do you call np(L1,L2)? Str.
- class_has(Cl, Has, Str) →
  What kind of wf(Cl) has wf(Has)? Str.
- act_verb(L1, L2, Str) →
  What kind of np(L1) can verb(L2)? Str.
- inact_verb(L1, L2, Str) →
  What kind of np(L1) can you verb(L2)? Str.
- vncompare(NPA, NPB, NP1, Verb1, NP2, Verb2) →
  What is the difference between np(NPA)and np(NPB)? You verb(Verb1) np(NP1) but you cannot verb(Verb2) np(NP2).
- vvcompare(NP1, NP2, Verb11, Verb12, Verb21, Verb22) →
  What is the difference between np(NP1) and np(NP2)? One verb(Verb11) and verb(Verb12) the other verb(Verb21) and verb(Verb22).
- poscompare(NP1, NP2, NPStr) →
  How is np(NP1) like np(NP2)? They are both NPStr.
- negcompare(NP1, NP2, NPStr1, NPStr2) →
  What is the difference between np(NP1) and np(NP2)? One is NPStr1 and the other is NPStr2.

Appendix E

The SAD transformation rules (WordNet)

The following are the rules used to construct small adequate descriptions (SADs) for the lexeme in the 'Described lexeme' slot, using the WordNet entries (as supplemented for the evaluation) for the lexemes in the 'SAD constraints' slot. For more information, see section 3.6.

```
[Described lexeme: NP
 Lexeme sequence: [V1, V2]
 In entries:
        act_verb(Noun, V1)
        act_verb(Noun, V2)
 SAD: {synonym(NP, Noun)}]

[Described lexeme: Lex
 Lexeme sequence: [N1]
 In entries:
        synonym(N1, Syn)
 SAD: {synonym(Lex, Syn)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1,X)
        category(X, noun)
        synonym(M2, Y)
        category(Y, noun)
 SAD: {cross_between(NP, [X], [Y])}]
```

```
[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        category(X, noun)
        act_verb(M2, Verb)
 SAD: {class(NP, X), act_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        category(X, noun),
        inact_verb(M2, Verb)
 SAD: {class(NP, X), inact_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        act_verb(M2, Verb)
 SAD: {class(NP, X), act_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        inact_verb(M2, Verb)
  SAD: {class(NP, X), inact_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M2, X)
        category(X, noun)
        has(M1, Y)
        category(Y, noun)
  SAD: {class(NP, X), has(NP, Y)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M2, X)
        category(X, noun)
        synonym(M1, Y)
        category(Y, noun)
  SAD: {class(NP, X), has(NP, Y)}]
```

```
[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, Adj)
        category(Adj, adj)
        synonym(M2, Class)
        category(Class, noun)
  SAD: {spec_is(NP, Adj), class(NP, Class)}]
```

Appendix F

The SAD transformation rules (hand-built lexicon)

The hand–built lexicon has more kinds of slot than WordNet does, allowing richer descriptions to be constructed. When hooked up to the hand–built lexicon, JAPE can use the following SAD transformation rules, *in addition* to the rules suitable for WordNet (see appendix E).

```
[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        specifier(M1, Adj)
        synonym(M2, Y)
SAD: {specifier(NP, Adj), class(NP, Y)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        specifier(M1, Adj)
        describes_all(M2, Y)
SAD: {specifier(NP, Adj), class(NP, Y)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        class(M2, Y)
        specifier(M2, Z)
SAD: {cross_between(NP, X, [Y,Z])}]
```

```
[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, W)
        specifier(M1, X)
        class(M2, Y)
        specifier(M2, Z)
SAD: {cross_between(NP, [X,W], [Z,Y])}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        specifier(M1, Adj)
        act_verb(M2, Verb)
SAD: {class(NP, [Adj, X]), act_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        specifier(M1, Adj),
        inact_verb(M2, Verb)
SAD: {class(NP, [Adj,X]), inact_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        describes_all(M2, Y)
SAD: {cross_between(NP, X, Y)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        describes_all(M1, X)
        describes_all(M2, Y)
SAD: {cross_between(NP, X, Y)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        category(X, noun)
        act_verb(M2, Verb)
        act_obj(M2, Obj)
SAD: {class(NP, X), act_verb(NP, Verb), act_obj(NP, Obj)}]
```

```
[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        category(X, noun)
        inact_verb(M2, Verb)
        location(M2, Loc)
SAD: {class(NP, X), inact_verb(NP, Verb), location(NP, Loc)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        describes_all(M1, X)
        act_verb(M2, Verb)
SAD: {class(NP, X), act_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        describes_all(M1, X)
        act_verb(M2, Verb)
        act_obj(M2, Obj)
SAD: {class(NP, X), act_verb(NP, Verb), act_obj(NP, Obj)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        describes_all(M1, X)
        inact_verb(M2, Verb)
SAD: {class(NP, X), inact_verb(NP, Verb)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        describes_all(M1, X)
        inact_verb(M2, Verb)
        location(M2, Loc)
SAD: {class(NP, X), inact_verb(NP, Verb), location(NP, Loc)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        specifier(M1, Adj)
        act_verb(M2, Verb)
        act_obj(M2, Obj)
SAD: {class(NP, [Adj, X]), act_verb(NP, Verb), act_obj(NP, Obj)}]
```

```
[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        specifier(M1, Adj)
        inact_verb(M2, Verb)
        location(M2, Loc)
SAD: {class(NP, [Adj,X]), inact_verb(NP, Verb), location(NP, Loc)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        synonym(M1, X)
        category(X, noun)
        used_to(M2, Verb)
SAD: {used_to(NP, Verb), used_to_obj(NP, X)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        specifier(M1, Adj)
        used_to(M2, Verb)
SAD: {used_to(NP, Verb), used_to_obj(NP, [Adj, X])}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        describes_all(M1,X)
        used_to(M2, Verb)
SAD: {used_to(NP, Verb), used_to_obj(NP, X)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        act_verb(M2, Verb)
        act_obj(M2, Obj)
SAD: {class(NP, X), act_verb(NP, Verb), act_obj(NP, Obj)}]

[Described lexeme: NP
 Lexeme sequence: [M1, M2]
 In entries:
        class(M1, X)
        inact_verb(M2, Verb)
        location(M2, Loc)
SAD: {class(NP, X), inact_verb(NP, Verb), location(NP, Loc)}]
```

Appendix G

How to use JAPE

JAPE is a research program and, as such, does not have an user-friendly interface. If you would like to try JAPE, contact me at kimb@dai.ed.ac.uk, and I will send you the program with full documentation. What follows is a brief outline of how to use JAPE, and the main predicates it uses.

JAPE is written in Sicstus prolog, although it has been successfully loaded into other prologs. The files required to load the program as described in chapter 4 are:

beep.pl A prolog version of the British English Example Pronunciation (BEEP) lexicon [Robinson, 1996].

binterface.pl An interface between BEEP and the main program.

homo.pl A prolog version of the homonym database [Townsend and Antworth, 1993].

matcher.pl The implementation of the phoneme-grapheme matching algorithm described in section 4.3.5.

mrc.pl A prolog version of the MRC database [Wilson, 1987].

filter.pl Functions for calculating the psycholinguistic scores for a text from information in the MRC database.

schemata.pl JAPE's schemata, as described in section 4.4.

templates.pl JAPE's templates and sentence forms, as described in section 4.5.

sadgen.pl The SAD generator, as described in section 4.6.

wn.pl Loads the WordNet database [Miller et al., 1990].

wn_interface.pl An interface between JAPE and WordNet.

verblinks.pl Verb–noun links, as described in section 4.3.3.

jape.pl The main program.

Once these files have been loaded, JAPE generates jokes through calls to its top level predicate, pun/3, described below. The main predicates available to the user are:

pun(?Schema, -Text, -Score): Generates a joke Text, using Schema. Score is a list of the Text's scores on the four psycholinguistic measures (familiarity, concreteness, imageability and age of acquisition) described in section 4.3.4. JAPE does not have a randomizer; it simply tries to satisfy the constraints given in the schema, in order, by looking in its lexicon, starting at the top.

instantiate_schema(+Schema, -Relations, -KeyWords): Instantiates a Schema, including the Relations it asserts. Also gives the KeyWords (i.e. the lexemes used to construct the joke text), so that they can be rated according to the psycholinguistic measures.

fill_template(+Relations, -Text, -KeyWords): Takes a list of asserted Relations, and generates the joke Text. Also keeps track of KeyWords to be rated.

rate(+KeyWords, -Scores): Given a list of KeyWords, returns a list of the psycholinguistic Scores (familiarity, concreteness, imageability and age of acquisition) for the KeyWords.

f_to_f(?Lexeme, ?Relation, ?Filler): The most general lexical access function. Relates Lexeme (a tag for a lexical entry) to Filler via some Relation. Used for syntactic, semantic and phonological information.

make_entry(+Lexeme, +WrittenForm, +SlotsAndFillers): Adds information to the lexicon for a particular Lexeme. WrittenForm is the near–surface form of the Lexeme (i.e. a word or word in a list), and SlotsAndFillers is a list of pairs of slot names (e.g. 'category') and fillers (e.g. 'noun').

Only the top-level function, pun/3, is required to generate jokes.

For more JAPE documentation, please contact me (kimb@dai.ed.ac.uk).

Appendix H

Allowable vocabulary items

The following is a list of allowable vocabulary items for the texts used in the confirmatory evaluation. WordNet sister and daughter nodes (see section 4.3.3) of the words below were also allowable vocabulary items.

adult, age, ale, alien, animal, ant, ape, apricot, aunt, away, back, bad, bag, bail, bake, bale, ball, bank, bare, bargain, base, basement, bass, bath, beach, beak, bear, beast, beat, beech, beet, beloved, better, bill, bird, bitter, blade, blunder, boil, bolt, bond, bottom, bow, boy, brake, bread, break, bright, broil, brush, bump, burn, buy, by, call, car, cast, caste, cede, cellar, cent, cheek, child, clever, close, clothes, clothing, clown, coarse, colour, corn, course, crude, crush, cunning, cure, curiosity, curve, dancing, dark, dear, deduction, deep, deer, depressed, desolate, dirty, dolt, door, doorway, dormitory, dough, draw, dye, earth, education, egg, end, engine, entrance, error, fail, failure, fair, fare, fast, feat, final, fir, fish, flair, flare, flash, fleece, fool, foul, fowl, frail, frank, full, fun, fur, garbage, genuine, gilt, girl, golden, groan, grown, guilt, hail, hall, hare, haul, hobby, hoe, horse, hour, house, human, ice, idea, in, inn, insect, iron, jam, jolt, jump, just, kinswoman, knight, labyrinth, lament, last, lax, leave, leg, lenient, level, light, line, link, lobby, locomotive, lodge, low, mail, maize, male, mammal, manner, manor, maze, melt, menu, mighty, miss, mistake, mite, moan, money, monkey, nerve, nice, night, noise, note, nude, odour, old, one, opinion, out, pain, pane, pause, peach, pelt, penny, period, person, personality, place, plain, plane, play, pleasant, poetry, position, post, potato, pouch, power, pupil, purse, rake, rancid, rarity, real, rear, reasonable, reel, regret, remedy, rhyme, right, rite, ritual, road, root, rotation, route, running, rush, sack, sad, sail, sale, sand, scent, sea, see, seed, sentiment, servant, shelter, shoot, shower, simple, smart, so, sole, son, sore, soul, sound, sour, sow, spank, spare, speck, spot, square, squeak, squeak, stale, stew, story, straight, stranger, student, stupid, style, sunburn, sunshine, sweat, sword, tail, tale, tax, tender, term, terrible, thaw, thought, tie, time, tractor, tree, trick, true, tush, ugly, up, vegetable, vulgar, wagon, wail, wares, waste, water, weak, wear, weather, well, whale, whirl, wolf, wool, world.

Appendix I

Allowable sentence structures

The following are the allowable sentence structures for all types of text used in the confirmatory evaluation. Some were not used at all. Here, “a” could be replaced by an appropriate determiner.

- What is the difference between a ___ and a ___? You ___ a ___, but you ___ a ___.
- What is the difference between a ___ and a ___? A ___ ___, while a ___ ___.
- What is the difference between a ___ and a ___? One ___ and ___, but the other ___ and ___.
- How is a ___ like a ___? They are both ___.
- What is the difference between a ___ and a ___? One is a ___, the other is a ___.
- What do you get when you cross a ___ and a ___? A ___.
- What do you call a ___? A ___.
- What do you call ___ you can ___? A ___.
- What do you call a ___ that can ___? A ___.
- What kind of ___ can ___? A ___.
- What kind of ___ can you ___? A ___.
- What kind of ___ can ___ a ___? A ___.
- What kind of ___ can you ___ at/in [location]? A ___.
- What do you call a ___ that can ___ a ___? A ___.
- What do you call a ___ that you can ___ at/in [location]? A ___.
- What do you use to ___ a ___? A ___.

Appendix J

JAPE-2's schemata

These are the schemata implemented in JAPE-2. The starred ones were used in the evaluation.

*** Lotus:**

```
{Lexical preconditions:
        noun_phrase(NPLex)
        component_lexemes(NPLex, LexA, LexB)
        written_form([LexA], WordA)
        homophone(WordA, HomWord)
        written_form([HomLex], HomWord)
        written_form([HomLex, LexB], NPWF)
 SAD constraints:
        described_by([HomLex, NPLex], Desc)
 Relationships:
        describes(NPWF, Desc)}
```

What kind of murderer has fibre? *A cereal killer.* [JAPE]

*** Bazaar:**

```
{Lexical preconditions:
  homophone(WordA, WordB)
  written_form([LexA], WordA)
  written_form([LexB], WordB)
  noun(LexB)
  written_form([LexA,LexB], WFAB)
 SAD constraints:
  described_by([LexA,LexB], SAD)
 Relationships:
  describes(WFAB, SAD)}
```

What do you call a strange market? *A bizarre bazaar.* [JAPE]

***Negcomp:**

```
{Lexical preconditions:
  rhyme(WordA, WordB)
  rhyme(WordC, WordD)
  alliterate(WordA, WordC)
  alliterate(WordB, WordD)
  written_form([LexC], WordC)
  written_form([LexD], WordD)
  noun(LexC)
  noun(LexD)
  written_form([LexA], WordA)
  written_form([LexB], WordB)
  written_form([LexA, LexD], WFAD)
  written_form([LexB, LexC], WFBC)
SAD constraints:
  described_by([LexA, LexD], SAD1)
  described_by([LexB, LexC], SAD2)
Relations:
  describes(WFAD, SAD1)
  describes(WFBC, SAD2)}
```

What's the difference between a pretty glove and a silent cat? *One's a cute mitten, the other's a mute kitten.* [Ertner, 1993] and JAPE.

***Phonsub:**

```
{Lexical preconditions:
  phon_form(Lex, PhonForm)
  subphons(PhonForm, SubPhonForm, Remainder)
  phon_form(SubLex, SubPhonForm)
  written_form([SubLex], SubWord)
  spelt_form(Remainder, RemSpell)
  subspell(NewWord, SubWord, RemSpell)
  written_form([Lex], WritForm)
  notequal(WritForm, NewWord)
SAD constraints:
  described_by([SubLex, Lex], Desc)
Relationships:
  describes(NewWord, Desc)}
```

What do you call a depressed train? A low-comotive. [JAPE]

***Hopchew:** `{Lexical preconditions:`

```
  rhyme(WordA, WordB)
  written_form([LexA], WordA)
  written_form([LexB], WordB)
  verb(LexA)
  verb(LexB)
```

```
  rhyme(WordC, WordD)
  written_form([LexC], WordC)
  written_form([LexD], WordD)
  verb(LexC)
  verb(LexD)
  alliterate(WordA, WordC)
  alliterate(WordB, WordD)
SAD constraints:
  described_by([LexA, LexD], SAD1)
  described_by([LexB, LexC], SAD2)
Relations:
  describes_same(
        {inact_verb(Lex1, LexA), inact_verb(Lex1, LexD)}, SAD1)
  describes_same(
        {inact_verb(Lex2, LexB), inact_verb(Lex2, LexC)}, SAD2)}
```

What's the difference between leaves and a car? One you brush and rake, the other you rush and brake. [JAPE]

***Poscomp:**

```
{Lexical preconditions:
        alternate_meaning(LexA, LexB)
        alternate_meaning(LexC, LexD)
        noun(LexB)
        noun(LexD)
        written_form([LexA, LexC], WFAC)
 SAD constraints:
        described_by([WordA, WordC], SAD1)
        described_by([WordB, WordD], SAD2)
 Relationships:
        describes(WFAC, SAD1)
        describes(WFAC, SAD2)}
```

How's a nice girl like a sugary bird? *They're both sweet chicks.* [JAPE]

***Rhyming:** `{Lexical preconditions:`

```
  np(NPLex)
  comp_lex(NPLex, LexA, LexB)
  written_form([LexA], WordA)
  rhymes(WordA, RhymWord)
  written_form([RhymLex], RhymWord)
  written_form([RhymLex, LexB], WFRB)
 SAD constraints:
  described_by([RhymLex,NPLex], SAD)
 Relations:
  describes(WFRB, SAD)}
```

What do you call a bath tour? *a tub crawl.* [JAPE]

***Elan:**
```
{Lexical preconditions:
  np(NPLex)
  comp_lex(NPLex, LexA, LexB)
  written_form([LexB], WordB)
  homonym(WordB, HomWord)
  written_form([HomLex], HomWord)
  noun(HomLex)
  written_form([LexA, HomLex], WFAH)
 SAD constraints:
  described_by([HomLex, NP], SAD)
 Relationships:
  describes(WFAH, SAD)}
```

What do you call a naked bruin? *A grizzly bare.* [JAPE]

***VN:**
```
{Lexical preconditions:
  inact_verb(LexA, LexB)
  written_form([LexA], WordA)
  written_form([LexB], WordB)
  not(inact_verb(LexD, LexC))
  written_form([LexC], WordC)
  homonym(WordA, WordC)
  written_form([LexD], WordD)
  homonym(WordB, WordD)
SAD constraints:
Relations:
  describes_same(
        {synonym(LexA, LexA)},
        {inact_verb(LexA, LexB)})
  describes_same(
        {synonym(LexC, LexC)},
        {not(inact_verb(LexC, LexD))})}
```

What's the difference between a sea and a sale? *You can sail a sea, but you can't see a sale.* [JAPE]

Coatshed:
```
{Lexical preconditions:
  alternate_meaning(LexA, LexB)
  verb(LexA)
  noun(LexB)
  alternate_meaning(LexC, LexD)
  verb(LexC)
  noun(LexD)
SAD constraints:
  described_by([LexA, LexD], SAD1)
  described_by([LexB, LexC], SAD2)
Relations:
```

```
    describes_same({act_verb(Lex1, LexA), act_obj(Lex1, LexD)}, SAD1)
    describes_same({act_verb(Lex2, LexB), act_obj(Lex2, LexC)}, SAD2)}
```

What's the difference between a hairy dog and a painter? *One sheds his coat and one coats his shed.*[Ertner, 1993][1]

Jumper:

```
{Lexical preconditions:
   np(NPLex)
   comp_lex(NPLex, LexA, LexB)
   written_form([LexB], WordB)
   homonym(WordB, HomWord)
   written_form([HomLex], HomWord)
   noun(HomLex)
   written_form([LexA, HomLex], WFAH)
  SAD constraints:
   described_by([LexA, HomLex], SAD)
  Relationships:
   describes(WFAH, SAD)}
```

What do you call a grumpy nude? *A grizzly bare.* [JAPE]

Woolly:

```
{Lexical preconditions:
   np(NPLex)
   comp_lex(NPLex, LexA, LexB)
   written_form([LexA], WordA)
   homonym(WordA, HomWord)
   written_form([HomLex], HomWord)
   written_form([HomLex,LexB], WFHB)
  SAD constraints:
   described_by([HomLex,LexB], SAD)
  Relationships:
   describes(WFHB, SAD)}
```

What do you get when you cross a murderer with a breakfast food? *A cereal killer.* [JAPE]

Double pun:

```
{Lexical preconditions:
   np(NPLex)
   comp_lex(NPLex, LexA, LexB)
   written_form([LexA], WordA)
   written_form([LexB], WordB)
```

[1] JAPE did not have the lexical data to produce jokes of this type.

```
 homonym(WordA, HomWordA)
 homonym(WordB, HomWordB)
 written_form([HomLexA], HomWordA)
 written_form([HomLexB], HomWordB)
 noun(HomLexB)
 written_form([HomLexA,HomLexB], WFHH)
SAD constraints:
 described_by([HomLexA,HomLexB], SAD)
Relationships:
 describes([WFHH, SAD)}
```

What do you call a gory nude? *A grisly bare.* [JAPE]

Appendix K

Questionnaire used in confirmatory evaluation

PRACTICE SHEET

This is a sheet for you to practice on.

Please listen to #1 on the tape. Then fill in the questions in part #1, below.
Then listen to #2 on the tape, and fill in the questions in part #2, below.

---

1. What is a horse's favourite sport?
Stable tennis.

Is that a joke? YES NO

How funny is it?

not funny at all | not very funny | not sure | funny | very funny

Have you heard it before? YES NO

---

2. What do you get when you cross a horse and a donkey?
A mule.

Is that a joke? YES NO

How funny is it?

not funny at all | not very funny | not sure | funny | very funny

Have you heard it before? YES NO

Figure K.1: Practice sheet used in the confirmatory evaluation.

Response Sheet

Please help us by filling in these sheets. Fill in Part I before listening to the tape. Fill in Part II while listening to the tape. Raise your hand if you have any problems.

PART I

Age:

Year:

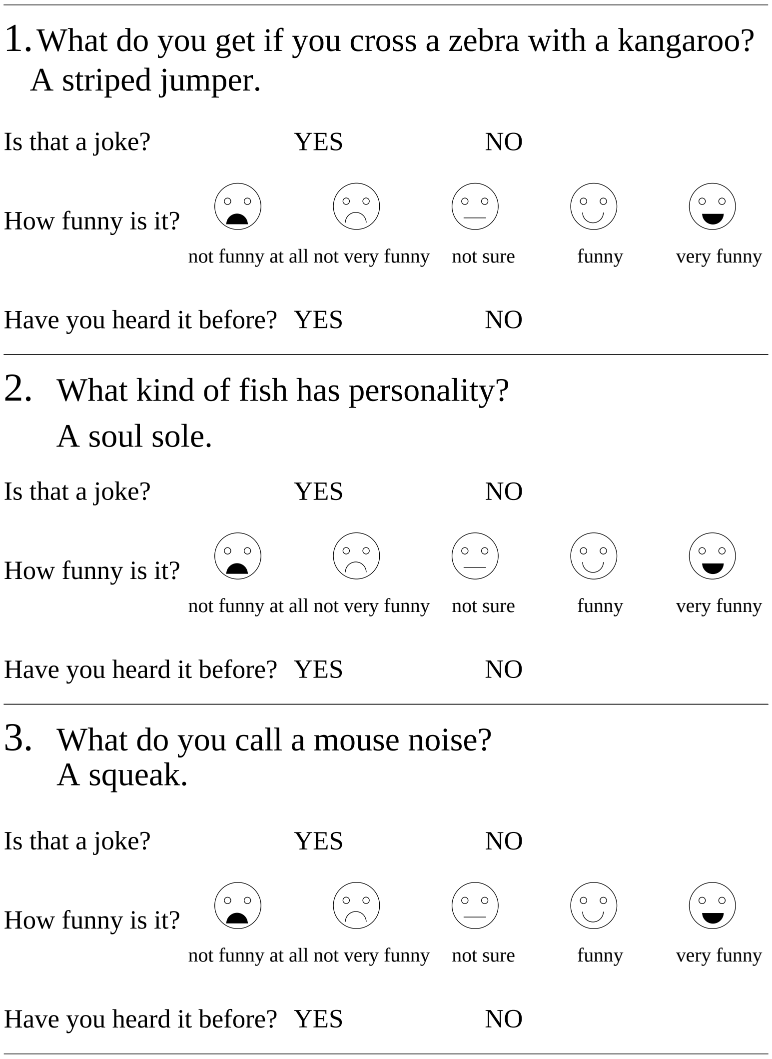
Do you like jokes? YES NO

PART II

Listen to the tape. It will tell you what to do.
Go to #1 on the next sheet. Listen to the tape. Answer the questions on the sheet. Then go to #2. Keep going like this until you get to #20.
Raise your hand if you have any problems.

COMMENTS:

Figure K.2: Cover sheet for questionnaire used in the confirmatory evaluation.



1. What do you get if you cross a zebra with a kangaroo?
A striped jumper.

Is that a joke? YES NO

How funny is it? not funny at all / not very funny / not sure / funny / very funny

Have you heard it before? YES NO

2. What kind of fish has personality?
A soul sole.

Is that a joke? YES NO

How funny is it? not funny at all / not very funny / not sure / funny / very funny

Have you heard it before? YES NO

3. What do you call a mouse noise?
A squeak.

Is that a joke? YES NO

How funny is it? not funny at all / not very funny / not sure / funny / very funny

Have you heard it before? YES NO

Figure K.3: Typical questionnaire sheet. Each questionnaire contained twenty texts to be judged.

Appendix L

Scores for each text

These are the average 'jokiness', 'funniness' and 'heard before' scores for each text, with their set number and source (H = human, J = JAPE, N = nonsensical, S = sensible), ordered by 'jokiness'. Scores for 'jokiness' range from 0 (none of the children who were asked to rate the text thought it was a joke) to 1 (all of the children who were asked to rate the text thought it was a joke). Scores for 'funniness' range from 1 to 5, with 1 meaning "not funny at all" and 5 meaning "very funny". Scores for 'heard before' range from 0 (none of the children who were asked to rate the text had heard it before) to 1 (all of the children who were asked to rate the text had heard it before).

Jokiness	Funniness	Heard	Source	Set	Text
1	4.33	0	J	4a	20. What's the difference between leaves and a car? One you brush and rake, the other you rush and brake.
1	4.08	0.33	H	4a	16. What do you get when you cross cars and sandwiches? Traffic jam.
1	3.92	0.54	H	2a	8. What kind of vegetable can jump? A spring onion.
1	3.92	0.46	H	2a	6. What do you call a cat with eight legs? An octopus.
1	3.92	0.08	H	2a	10. What do you get when you cross a house with a pancake? A flat.
1	3.83	0.18	H	6a	16. What do you call a bad dream with teeth? A bite-mare.
1	3.77	0.62	H	2a	9. What kind of food do octopuses eat? Fish and ships.
1	3.73	0.17	H	10a	15. What nuts can you use to build a house? Walnuts.
1	3.73	0.08	H	7a	3. What do you call a fun cow? A-moo-sement.
1	3.67	0.25	H	4a	18. What kind of clothing can a spider wear? A coat of arms.
1	3.67	0.18	H	4a	2. What kinds of babies do winds have? Chill-dren.
1	3.67	0.08	H	6a	8. What do you call a boy who eats six bowls of raspberries? Berry greedy.

Jokiness	Funniness	Heard	Source	Set	Text
1	3.5	0.17	H	8a	19. What do you use to talk to a skunk? A smelly-phone.
1	3.36	0.08	J	10a	14. What kind of boy burns? A son-burn.
1	3.31	0.23	J	2a	13. What do you call a beloved mammal? A dear deer.
0.92	3.62	0.31	H	1a	12. What do you call a deer with no eyes? No eye-deer.
0.92	3.92	0.33	H	9a	7. What kind of fruit fixes taps? A plum-ber.
0.92	3.91	0.18	H	3a	14. How is a window like a headache? They are both panes.
0.92	3.75	0.25	H	5a	18. What kind of tent has hair? A wig-wam.
0.92	3.75	0.08	H	6a	20. What do you call a cold aunt? Aunty-freeze.
0.92	3.73	0.18	H	10a	17. What do you use to talk to an elephant? An elly-phone.
0.92	3.64	0.10	J	3a	6. What do you get when you cross a bag and a human? A purse-on.
0.92	3.58	0.17	H	8a	4. What do ghosts eat for pudding? Scream cakes.
0.92	3.58	0.17	H	6a	10. What do you call a lizard on the wall? A rep-tile.
0.92	3.58	0.08	J	5a	20. What's the difference between money and and a bottom? One you spare and bank, the other you bare and spank.
0.92	3.55	0.10	H	10a	20. What kind of Christmas tree does a hedgehog have? A porcu-pine.
0.92	3.5	0.5	H	5a	1. What do you get if you cross a zebra with a kangaroo? A striped jumper.
0.92	3.5	0.36	H	4a	19. How is a car like an elephant? They both have trunks.
0.92	3.33	0.17	H	5a	19. What's the difference between a piece of cotton and a tattered toy? One is a bare thread and the other is threadbare.
0.92	3.33	0	J	4a	5. What do you call true dancing? A real reel.
0.92	3.18	0.33	H	7a	7. What do you call a dead author? A ghost writer.
0.92	3.10	0	J	10a	18. What do you call a bright night? Light time.
0.92	3.08	0.45	H	3a	9. What's the difference between a pony and a sore throat? One is a horse, and the other is hoarse.
0.92	3.08	0.17	H	8a	11. What does a vegetable earn? Celery.
0.92	2.83	0.17	J	6a	11. What kind of beast has a fleece? A wool-f.
0.92	2.82	0	J	3a	19. What's the difference between a straight bill and a nude noise? One's a square beak and the other's a bare squeak.
0.85	3.38	0.33	J	2a	1. What do you get when you cross a monkey and a peach? An ape-ricot.

Jokiness	Funniness	Heard	Source	Set	Text
0.85	3.08	0.33	H	1a	6. What kind of animal plays cricket? A bat.
0.85	3.08	0.31	H	1a	2. What do you call a monkey bed? An ape-ricot.
0.83	3.83	0.33	J	9a	13. What kind of pupil has sweat? A stew-dent.
0.83	3.58	0.17	H	8a	9. What do you call a clever skunk? A fast stinker.
0.83	3.58	0.17	H	4a	3. What kind of food do cannibals eat? Human beans.
0.83	3.55	0.17	J	7a	18. What do you call an Earth rotation? A whirl-d.
0.83	3.45	0.10	H	9a	3. What do you call a ghost summer race? A dead heat.
0.83	3.42	0	H	3a	5. How is mathematics like a headache? They are both sum trouble.
0.83	3.36	0.83	J	7a	4. What do you call a tender blade? A sore-d.
0.83	3.33	0.17	J	4a	10. What do you get when you cross a penny and an odour? A cent scent.
0.83	3.27	0.08	J	7a	16. What do you call a pleasant period? The nice age.
0.83	3.25	0.08	J	8a	7. What do you call an adult moan? A grown groan.
0.83	3.18	0.33	H	7a	1. What do plumbers have for Christmas dinner? Plumbed pudding.
0.83	3.17	0.10	J	4a	13. What do you get when you cross a remedy and a rarity? A cure-iosity.
0.83	3.10	0.25	J	10a	5. What kind of hall has a doorway? A door-mitory.
0.83	3.10	0	J	10a	9. What do you call a corn labyrinth? A maize maze.
0.83	2.75	0	J	6a	19. What kind of term has clowns? A fool term.
0.83	2.67	0.18	H	3a	11. What do you get if you cross a car with a vile substance? Crude oil.
0.77	3.38	0.17	H	2a	15. What kind of dog has no tail? A hot dog.
0.77	3	0.15	H	1a	17. What do you get if you cross a flower with a monkey? A chim-pansy.
0.77	2.69	0.15	J	2a	4. What's the difference between a terrible pouch and a desolate rear? One's a bad sack and the other's a sad back.
0.77	2.54	0.08	J	1a	5. What's the difference between a sea and a sale? You can sail a sea, but you can't see a sale.
0.75	3.75	0.10	H	8a	13. What do you call a fight between an apple and an orange? Fruit punch.
0.75	3.58	0.25	H	9a	9. What do you call a bird that lives under ground? A miner bird.
0.75	3.58	0.08	H	5a	10. What kind of being drinks beer? An ale-ien.

Jokiness	Funniness	Heard	Source	Set	Text
0.75	3.5	0	H	4a	12. Wht do you get when you cross ghosts with trees? Cemetries.
0.75	3.45	0	J	10a	11. What's the difference between a seed and a so? You can sow a seed, but you can't cede a so.
0.75	3.42	0	H	6a	13. What do you get when you cross a chicken and a power pack? A battery hen.
0.75	3.36	0.25	H	7a	10. What kind of apple is bad-tempered? A crab-apple.
0.75	3.33	0.10	J	9a	17. What do you call an old bottom? A stale end.
0.75	3.18	0.17	H	10a	16. What kind of ghost rings the door bell? A dead ringer.
0.75	3.17	0.08	J	5a	5. What kind of boy has the post? A mail child.
0.75	3	0.25	H	7a	14. What do you call a line of ghosts? A dead line.
0.75	3	0.08	J	3a	4. What's the difference between a horse and a wagon? One bolts and jumps, the other jolts and bumps.
0.75	2.92	0.25	H	6a	7. What bird is red and steals? A robin.
0.75	2.92	0.08	J	5a	12. What do you call a rancid shower? A sour bath.
0.75	2.75	0	J	9a	14. What kind of iron has a position? Caste iron.
0.69	2.85	0.10	H	1a	8. Why is a gold coin like a criminal? They are both gilty.
0.69	2.62	0.08	J	1a	13. What do you call a depressed engine? A low-comotive.
0.67	3.33	0.18	J	5a	15. What's the difference between a pane and a brake? You can break a pane, but you can't pain a brake.
0.67	3.27	0.36	H	7a	19. What do you call a hot sheep? A woolly sweater.
0.67	3.08	0.33	J	8a	14. What do you use to colour an animal? Hare dye.
0.67	2.92	0.17	H	3a	16. What's the difference between a fish and a fly? A fish can fly but a fly cannot fish.
0.67	2.91	0.08	J	7a	20. What do you call a poetry pause? A rhyme out.
0.67	2.83	0	J	8a	5. What kind of girl has an error? A Miss take.
0.67	2.83	0	J	5a	8. What kind of penny has an opinion? A cent-iment.
0.67	2.42	0.25	J	9a	20. What kind of idea melts? A thaw-t.
0.64	3.10	0.10	J	9a	4. What do you call bare garbage? Nude waste.
0.62	3.08	0.08	J	2a	19. What's the difference between a tractor and a servant? One hoes and bales, the other bows and hails.
0.62	2.77	0.08	J	2a	14. What do you get when you cross a road and a basement? A route cellar.

Jokiness	Funniness	Heard	Source	Set	Text
0.62	2.46	0	J	1a	10. What's the difference between a potato and an egg? One you broil and bake, the other you boil and break.
0.62	2.38	0.15	J	2a	20. What do you get when you cross a bird and a blunder? A fowl up.
0.58	3.36	0.17	H	10a	1. What do you call a sick bird? An ill eagle.
0.58	3	0	J	6a	14. What kind of dark has horses? Knight time.
0.58	2.92	0.10	H	5a	6. What kind of dairy product has muscles? Hard cheese.
0.58	2.82	0	J	10a	2. What kind of leg can shoot? A bow leg.
0.58	2.73	0.10	J	10a	12. What do you get when you cross style and flash? A flair flare.
0.58	2.67	0.27	J	3a	3. What do you call a nude animal? A bare bear.
0.58	2.67	0.17	H	3a	17. What is the difference between a butcher and a fish? One has scales and sees meat and the other has scales and meets seas.
0.58	2.5	0.08	J	6a	18. What do you call a fun spot? A play-ce.
0.58	2.5	0.08	J	5a	4. What's the difference between a just bond and nude failure? One's fair bail and the other's a bare fail.
0.58	2.5	0.08	J	5a	2. What kind of fish has personality? A soul sole.
0.58	2.42	0.08	J	4a	14. How is simple vegetable like a level sound? They're both plain beats.
0.58	2.33	0	J	9a	2. What do you call better water? Well water.
0.58	2.25	0	J	3a	18. What kind of bird is dirty? A foul fowl.
0.55	2.64	0.10	H	9a	5. What's the difference between a game and a romance novel? One is exciting, the other is yecch writing.
0.55	2.33	0.17	J	8a	3. What do you call a genuine bill? A frank note.
0.54	3.08	0.15	H	1a	4. What is the difference between an enormous hen and a huge coward? One is a giant chicken and the other is a chicken giant.
0.54	2.54	0.17	J	1a	20. What kind of tree has sand? A beach beech.
0.54	2.08	0	H	2a	11. What's the difference between a picture and a church? One is a centre-piece and the other is a peace centre.
0.5	3.17	0	H	9a	15. How is a boy scout like a tin of raspberries? They're both prepared.
0.5	2.73	0.08	N	10a	8. What is the difference between a ghost and a spot? You can miss hay, but you cannot help a dormitory.
0.5	2.33	0	J	3a	13. What's the difference between clothes and wares? You can wear clothes, but you can't close wares.
0.5	2.08	0.17	J	8a	20. What do you call stupid bread? A dough-lt.

Jokiness	Funniness	Heard	Source	Set	Text
0.5	2.08	0	J	3a	10. What kind of manor has style? A manner house.
0.46	2.38	0	N	2a	12. What do you call a gilt explanation? A true principal.
0.42	2.83	0.08	H	5a	7. What do you call an Aboriginal in the blood? A foreign body.
0.42	2.58	0.08	H	8a	1. What paper does a hang man read? A noose-paper.
0.42	2.55	0.08	H	10a	3. What ribbon do lawyers use? Red tape.
0.42	2.33	0.18	N	9a	11. How is an ocean like a saucer? They are both sensitive.
0.42	2.33	0	J	4a	1. What kind of curve has cheek? A nerve ball.
0.42	2.27	0.10	J	7a	15. What kind of tree has a pelt? A fur tree.
0.42	2.25	0.17	H	9a	19. What has four legs and one arm? A pit bull.
0.42	2.17	0.08	J	6a	4. What do you call a smart ritual? Rite smart.
0.42	2.17	0.08	J	6a	1. What kind of speck has power? A mighty mite.
0.42	2.08	0.17	J	4a	17. What kind of tush has a story? A tale end.
0.42	2.08	0	J	4a	9. What do you get when you cross a bitter and a stranger? An ale-ien.
0.42	1.91	0.27	S	3a	8. What's the difference between a lemon and an orange? One is yellow and the other is orange.
0.42	1.91	0.08	J	7a	13. What do you call an ugly instrument? A base bass.
0.42	1.91	0	J	7a	9. What do you get when you cross sunshine with a menu? Fare weather.
0.42	1.75	0.08	J	3a	12. What kind of draw has a lobby? An entrance haul.
0.42	1.73	0	J	7a	8. What do you call a mammal's lament? A whale wail.
0.38	2.31	0	N	2a	18. What do you get when you cross a remedy with a mall? A coarse line.
0.36	2.45	0.10	S	9a	6. What kind of entrance can you open? A door.
0.33	2.5	0.08	N	6a	3. What do you get when you cross an amusement and an eagle? A bad bowl.
0.33	2.45	0.10	S	10a	4. How is a robin like an eagle? They are both birds.
0.33	2.42	0.08	S	9a	8. What is the difference between a sheep and a wolf? One eats grass, the other eats meat.
0.33	2.42	0	N	5a	16. What kind of monkey can remedy? A spare peach.
0.33	2.33	0.08	N	9a	1. What kind of enigma can burn? A cold raspberry.

Jokiness	Funniness	Heard	Source	Set	Text
0.33	2.33	0	J	9a	18. What do you call golden regret? Gilt guilt.
0.33	2.18	0.08	S	10a	13. What is the difference between a raspberry and a walnut? One is a berry, the other is a nut.
0.33	2.17	0	J	6a	9. What do you call jam time? Crush hour.
0.33	2	0.08	S	10a	6. How is celery like broccoli? They are both vegetables.
0.33	1.92	0.17	S	5a	9. What is the difference between a scent and an odour? One is a good smell and the other is a bad smell.
0.33	1.67	0	N	3a	20. What kind of dancing has a principle? A cunning personality.
0.31	2.15	0	J	2a	3. What do you get when you cross a link and a lodge? A tie inn.
0.31	2.15	0	J	1a	18. How is an ugly insect like a deep kinswoman? They're both bass aunts.
0.31	1.92	0	J	1a	7. What do you call a clever feat? Cunning away.
0.31	1.77	0	J	1a	14. What do you get when you cross a bargain and a hobby? A buy line.
0.31	1.62	0	N	1a	11. What do you get when you cross a regret with an adult? A primary utterance.
0.27	2.18	0.10	S	3a	2. What kind of craft has sails? A ship.
0.25	2.36	0	J	10a	19. What do you call a weak tail? A frail end.
0.25	2.25	0.33	S	9a	16. What kind of reptile has legs? A lizard.
0.25	2.18	0.10	N	10a	7. What kind of load can you hate? A baron.
0.25	2.17	0	S	4a	7. How is a remedy like a cure? They both relieve pain.
0.25	2.08	0.08	N	5a	17. What's the difference between a beloved deer and a dim feint? One is a fair bird and the other is dirty gold.
0.25	1.92	0.08	N	4a	11. What kind of feat can squeak? Cunning clothes.
0.25	1.92	0	J	8a	6. What do you call an ugly fish? A base bass.
0.25	1.91	0.08	S	7a	6. What do you call a tender spot? A sore.
0.25	1.83	0.27	S	8a	8. What kind of animal can fly? A bird.
0.25	1.83	0	J	9a	12. What do you call a vulgar education? A coarse course.
0.25	1.82	0.33	S	7a	11. What kind of dish do you use to eat stew? A bowl.
0.25	1.82	0.08	N	7a	12. What do you call a greedy tape? A fast crab.
0.25	1.75	0.25	S	8a	2. What do you call a smelly animal? A skunk.

Jokiness	Funniness	Heard	Source	Set	Text
0.25	1.75	0.17	N	3a	7. What's the difference between a bass and sand? A penny dims and faints, and an odour costs and regrets.
0.25	1.75	0	N	3a	1. What do you call a grown bargain? A gilt hobby.
0.25	1.5	0.17	S	3a	15. What do you call a green mineral? Jade.
0.25	1.45	0.08	N	7a	5. What do you get when you cross a scout with a clever celery? A man.
0.23	2	0	J	2a	7. What do you call a final trick? A last one.
0.23	1.77	0.17	S	1a	19. What do you call a stick with leaves? A tree.
0.23	1.62	0.15	J	1a	16. What kind of squeak has clothing? A clothes call.
0.23	1.62	0	S	2a	16. How is an ape like a chimpanzee? They are both monkeys.
0.23	1.54	0	N	1a	9. What kind of call can you blunder? An alien fowl.
0.18	1.67	0	N	8a	17. What is the difference between an aunt and teeth? One taps and whirls, the other writes and punches.
0.17	2	0.17	N	5a	11. What kind of sole is low? Bitter clothing.
0.17	2	0.10	S	8a	10. What is the difference between straw and tin? You can burn straw, but you can't burn tin.
0.17	2	0.08	N	10a	10. What kind of mare has a walnut? A plumber.
0.17	1.83	0.17	S	5a	14. What's the difference between a tree and a tractor? One has leaves and the other has wheels.
0.17	1.83	0.08	N	6a	15. What is the difference between earth and a beast? One is an idea, the other is steadfast.
0.17	1.75	0.08	J	6a	2. What do you call a reasonable menu? A fair fare.
0.17	1.75	0.08	N	6a	17. How is a red thaw like a sword? They are both ribbons.
0.17	1.67	0.10	S	5a	3. What do you call a mouse noise? A squeak.
0.15	1.69	0	N	2a	17. What's the difference between a bear and a scent? An instrument can course but a tree cannot move.
0.15	1.46	0.08	S	1a	3. What's the difference between a bird and a horse? One has wings and the other has legs.
0.10	1.83	0.17	N	8a	16. What kind of berry has a bell? A sick newspaper.
0.10	1.67	0	J	8a	12. What do you call a lenient shelter? A lax deduction.
0.08	1.92	0	N	9a	10. What is the difference between an entrance and antifreeze? One is straw, the other is a sinful mammal.
0.08	1.73	0.08	N	7a	2. What is the difference between a company and an idea? A boy can melt, but a sore cannot twist.
0.08	1.64	0.08	N	4a	6. What kind of engine has a lodge? An apricot purse.

Jokiness	Funniness	Heard	Source	Set	Text
0.08	1.55	1	S	7a	17. What kind of place can you sleep in? A dormitory.
0.08	1.5	0	S	4a	8. What do you get if you cross an occupation and pleasure? A hobby.
0.08	1.42	0	S	4a	15. What kind of animal can swing through trees? A monkey.
0.08	1.46	0	N	1a	15. What's the difference between a nude and an animal? You can bare a fish but you cannot scent a base.
0.08	1.38	0.08	S	2a	5. How is a cod like a bass? They are both fish.
0.08	1.31	0.10	S	1a	1. What kind of yellow fruit can you eat? A banana.
0	1.64	0.10	N	4a	4. How is a cent like an education? They are both impartial mammals.
0	1.58	0	N	8a	15. What do you call a fast place? New dough.
0	1.5	0.17	S	6a	6. What kind of man fixes taps? A plumber.
0	1.5	0.08	S	6a	5. What kind of food grows on trees? Fruit.
0	1.42	0.08	S	6a	12. What do you call a person who studies? A student.
0	1.23	0.08	S	2a	2. What do you call a broken nose? Damage.