An implemented model of punning riddles

Kim Binsted*and Graeme Ritchie
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland EH1 1HN
kimb@aisb.ed.ac.uk graeme@aisb.ed.ac.uk

Abstract

In this paper, we discuss a model of simple question–answer punning, implemented in a program, JAPE-1, which generates riddles from humour–independent lexical entries. The model uses two main types of structure: *schemata*, which determine the relationships between key words in a joke, and *templates*, which produce the surface form of the joke. JAPE-1 succeeds in generating pieces of text that are recognizably jokes, but some of them are not very good jokes. We mention some potential improvements and extensions, including post–production heuristics for ordering the jokes according to quality.

Humour and artificial intelligence

If 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), then the production of humorous texts, including jokes and riddles, is a fit topic for AI research. As well as probing some intriguing aspects of the notion of "intelligence", it has the methodological advantage (unlike, say, computer art) of leading to more directly falsifiable theories: the resulting humorous artefacts can be tested on human subjects.

Although no computationally tractable model of humour as a whole has yet been developed (see (Attardo & Raskin 1991) for a general theory of verbal humour, and (Attardo 1994) for a comprehensive survey), we believe that by tackling a very limited and linguistically-based set of phenomena, it is realistic to start developing a formal symbolic account.

One very common form of humour is the question-answer joke, or riddle. Most of these jokes (e.g. almost a third of the riddles in the Crack-a-Joke Book (Webb 1978)) are based on some form of pun. For example:

What do you use to flatten a ghost? *A spirit level.* (Webb 1978)

*Thanks are due to Canada Student Loans, the Overseas Research Students Scheme, and the St Andrew's Society of Washington, DC, for their financial support.

This riddle is of a general sort which is of particular interest for a number of reasons. The linguistics of riddles has been investigated before (e.g. (Pepicello & Green 1984)). Also, there is a large corpus of riddles to examine: books such as (Webb 1978) record them by the thousand. Finally, riddles exhibit more regular structures and mechanisms than some other forms of humour.

We have devised a formal model of the punning mechanisms underlying some subclasses of riddle, and have implemented a computer program which uses these symbolic rules and structures to construct punning riddles from a humour-independent (i.e. linguistically general) lexicon. An informal evaluation of the performance of this program suggests that its output is not significantly worse than that produced by human composers of such riddles.

Punning riddles

Pepicello and Green (Pepicello & Green 1984) describe the various strategies incorporated in riddles. They hold the common view that humour is closely related to ambiguity, whether it be linguistic (such as the phonological ambiguity in a punning riddle) or contextual (such as riddles that manipulate social conventions to confuse the listener). What the linguistic 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 (cf. "iconism" (Attardo 1994)). Riddles of this type are known as *puns*.

We decided to select a subset of riddles which displayed regularities at the level of semantic, or logical, structure, and whose structures could be described in fairly conventional linguistic terms (simple lexical relations). As a sample of existing riddles, we studied "The Crack-a-Joke Book" (Webb 1978), a collection of jokes chosen by British children. These riddles are simple, and their humour generally arises from their punning nature, rather than their subject matter. This sample does not represent sophisticated adult humour, but it suffices for an initial exploration.

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

Syllable substitution: Puns using this strategy confuse a syllable (or syllables) in a word with a similar- or identical-sounding word. For example:

> What do short-sighted ghosts wear? *Spooktacles.* (Webb 1978)

Word substitution: Word substitution is very similar to syllable substitution. In this strategy, an entire word is confused with another similar- or identical-sounding word. For example:

> How do you make gold soup? *Put fourteen carrots in it.* (Webb 1978)

Metathesis: Metathesis is quite different from syllable or word substitution. Also known as *spoonerism*, it uses a reversal of sounds and words to suggest (wrongly) a similarity in meaning between two semantically-distinct phrases. For example:

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

All three of the above-described types of pun are potentially tractable for detailed formalisation and hence computer generation. We chose to generate only word-substitution puns, simply because lists of phonologically identical words (*homonyms*) are readily available, whereas the other two types require some kind of sub-word comparison. In particular, the class of jokes which we chose to generate all: use word substitution; have the substituted word in the *punchline* of the joke, rather than the question; and substitute a homonym for a word in a *common noun phrase* (cf. the "spirit level" riddle cited earlier). These restrictions are simply to reduce the scope of the research even further, so that the chosen subset of jokes can be covered in a comprehensive, rigorous manner. We believe that our basic model, with some straightforward extensions, is general enough to cover other forms.

Symbolic descriptions

Our analysis of word-substitution riddles is based (semi-formally) on the following essential items, related as shown in Figure 1:

- a valid English word/phrase
- the meaning of the word/phrase
- a shorter word, phonologically similar to part of the word/phrase
- the meaning of the shorter word
- a fake word/phrase, made by substituting the shorter word into the word/phrase
- the meaning of the fake word/phrase, made by combining the meanings of the original word/phrase and the shorter word.

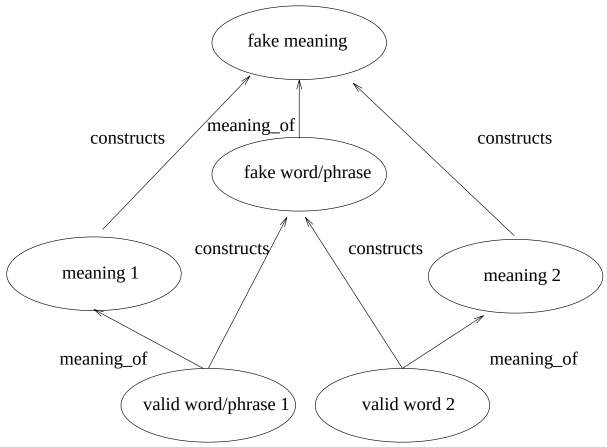

Figure 1: The relationships between parts of a pun

At this point, it is important to distinguish between the mechanism for building the *meaning* of the fake word/phrase, and the mechanism that uses that meaning to build a question with the word/phrase as an answer. Consider the joke:

> What do you give an elephant that's exhausted? *Trunkquillizers.* (Webb 1978)

In this joke, the word "trunk", which is phonologically similar to the syllable "tranq", is substituted into the valid English word "tranquillizer". The resulting fake 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" (in this case, a tranquillizer for elephants). 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?

We have adopted the term *schema* for the symbolic description of the underlying configuration of meanings and words, and *template* for the textual patterns used to construct a question-answer pair.

Lexicon

Our minimal assumptions about the structure of the lexicon are as follows. There is a (finite) set of *lexemes*.

A lexeme is an abstract entity, roughly corresponding to a meaning of a word or phrase. Each lexeme has exactly one entry in the lexicon, so if a word has two meanings, it will have two corresponding lexemes. Each lexeme may have some *properties* which are true of it (e.g. being a noun), and there are a number of possible *relations* which may hold between lexemes (e.g. synonym, homonym, subclass). Each lexeme is also associated with a *near-surface form* which indicates (roughly) the written form of the word or phrase.

Schemata

A *schema* stipulates a set of relationships which must hold between the lexemes used to build a joke. More specifically, a schema determines how real words/phrases are glued together to make a fake word/phrase, and which parts of the lexical entries for real words/phrases are used to construct the meaning of the fake word/phrase.

There are many different possible schemata (with obscure symbolic labels which the reader can ignore). For example, the schema in Figure 2 constructs a fake phrase by substituting a homonym for the first word in a real phrase, then builds its meaning from the meaning of the homonym and the real phrase.

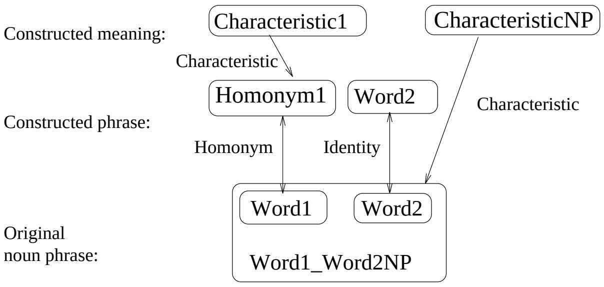

Figure 2: The *lotus* schema

The schema shown in Figure 2 is *uninstantiated*; that is, the actual lexemes to use have not yet been specified. Moreover, some of the relationships are still quite general — the *characteristic* link merely indicates that *some* lexical relationship must be present, and the *homonym* link allows either a homophone or the same word with an alternative meaning. Instantiating a schema means inserting lexemes in the schema, and specifying the exact relationships between those lexemes (i.e. making exact the *characteristic* links). For example, in the lexicon, the lexeme **spring_cabbage** might participate in relations as follows:

```
class(spring_cabbage, vegetable)
location(spring_cabbage, garden)
action(spring_cabbage, grows)
adjective(spring_cabbage, green)
....
```

If **spring_cabbage** were to be included in a schema, at one end of a *characteristic* link, the other end of the link could be associated with any one, or any combination of, these values (vegetable, garden, etc), depending on the exact label (class, location, etc.) chosen for the characteristic link.

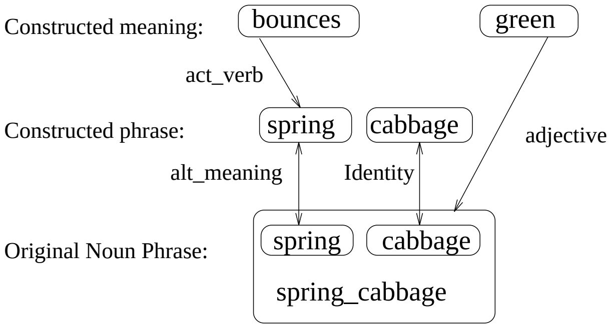

Figure 3: A completely instantiated *lotus* schema

The completely instantiated *lotus* schema in Figure 3 could (with an appropriate template — see below) be used to construct the joke:

What's green and bounces? *A spring cabbage.* (Webb 1978)

Templates

A template is used to produce the surface form of a joke from the lexemes and relationships specified in an instantiated schema. Templates are not inherently humour-related. Given a (real or nonsense) noun phrase, and a meaning for that noun phrase (genuine or constructed), a template builds a suitable question-answer pair. Because of the need to provide a suitable amount of information in the riddle question, every schema has to be associated with a set of appropriate templates. Notice that the precise choice of relations for the under-specified "characteristic" links will also affect the appropriateness of a template. (Conversely, one could say that the choice of template influences the choice of lexical relation for the characteristic link, and this is in fact how we have implemented it.) Abstractly, a template is a mechanism which maps a set of lexemes (from the instantiated schema) to the surface form of a joke.

The JAPE-1 computer program

Introduction

We have implemented the model described earlier in a computer program called JAPE-1, which produces the chosen subtype of jokes — riddles that use homonym substitution and have a noun phrase punchline. Such riddles are representative of punning riddles in general, and include approximately one quarter of the punning riddles in (Webb 1978).

JAPE-1 is significantly different from other attempts to computationally generate humour in various ways: its lexicon is humour-independent (i.e. the structures that generate the riddles are distinct from the semantic and syntactic data they manipulate), and it generates riddles that are similar on a strategic and structural level, rather than in surface form.

JAPE-1's main mechanism attempts to construct a punning riddle based on a common noun phrase. It has several distinct knowledge bases with which to accomplish this task: the lexicon (including the homonym base), a set of schemata, a set of templates, and a post-production checker.

Lexicon

The lexicon contains humour–independent semantic and syntactic information about the words and noun phrases entered in it, in the form of "slots" which can contain other lexemes or may contain other symbols. A typical entry might be:

```
lexeme = jumper_1                 countable = yes
category = noun                   class = clothing
written_form = ‘‘jumper’’         specifying_adj = warm
vowel_start = no                  synonym =  sweater
```

Although the lexicon stores syntactic information, the amount of syntax used by the rest of the program is minimal. Because the templates are based on certain fixed forms, the only necessary syntactic information has to do with the syntactic category, verb person, and determiner agreement. Also, the lexicon need only contain entries for nouns, verbs, adjectives, and common noun phrases — other types of word (conjunctions, determiners, etc) are built into the templates. Moreover, because the model implemented in JAPE-1 is restricted to covering riddles with noun phrase punchlines, the schemata require *semantic* information only for nouns and adjectives.

The "homonym" relation between lexemes was implemented as a separate *homonym base* derived from a list (Townsend & Antworth 1993) of homophones in American English, shortened considerably for our purposes. The list now contains only common, concrete nouns and adjectives. The homonym base also includes words with two distinct meanings (e.g. "lemon", the fruit, and "lemon", slang for a low-quality car).

Schemata

JAPE-1 has a set of six schemata, one of which is the *jumper* schema, shown in Figure 4. The same schema, instantiated in two different ways, is shown in Figure 5 and Figure 6.

Templates

Since riddles often use certain fixed forms (for example, "What do you get when you cross ___ with ___ ?"), JAPE-1's templates embody such standard forms. A JAPE-1 template consists of some fragments of canned text with "slots" where generated words or phrases can be inserted, derived from the lexemes in an instantiated schema. For example, the *syn_syn* template:

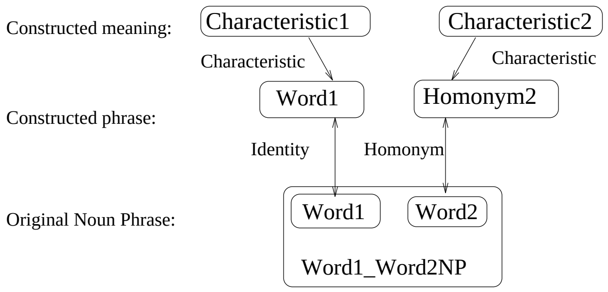

Figure 4: The uninstantiated *jumper* schema

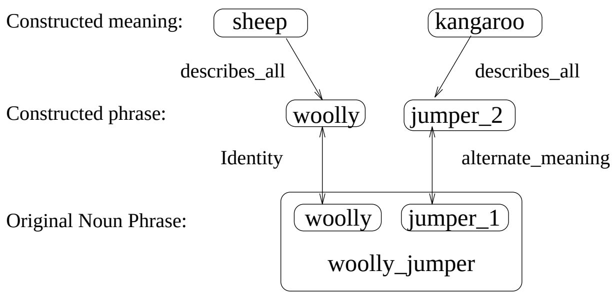

Figure 5: The instantiated *jumper* schema, with links suitable for the *syn_syn* template. Gives the riddle: What do you get when you cross a sheep and a kangaroo? *A woolly jumper.*

> What do you get when you cross [**text fragment generated from the first characteristic lexeme(s)**] with [**text fragment generated from the second characteristic lexeme(s)**]? *[the constructed noun phrase].*

A template also specifies the values it requires to be used for "characteristic" links in the schema; the *describes_all* labels in Figure 5 are derived from the *syn_syn* template. When the schema has been fully instantiated, JAPE-1 selects one of the associated templates, generates text fragments from the lexemes, and

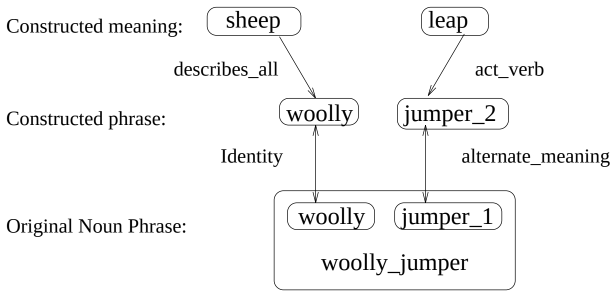

Figure 6: The instantiated *jumper* schema, with links suitable for the *syn_verb* template. Gives the riddle: What do you call a sheep that can leap? *A woolly jumper.*

slots those fragments into the template.

Another template which can be used with the *jumper* schema (see Figure 6) is the *syn_verb* template:

> What do you call **[text fragment generated from the first characteristic lexeme(s)]** that **[text fragment generated from the second characteristic lexeme(s)]**? *[the constructed noun phrase.]*

Post-production checking

To improve the standard of the jokes slightly, some simple checks are made on the final form. The first is that none of the lexemes used to build the question and punchline are accidentally identical; the second is that the lexemes used to build the nonsense noun phrase and its meaning, do not build a *genuine* common noun phrase.

The evaluation procedure

An informal evaluation of JAPE-1 was carried out, with three stages: *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" (simply to check for errors and excessively obscure suggestions), entered into JAPE-1's lexicon, and a substantial set of jokes were produced. A different group of volunteers then gave verdicts, both quantitative and qualititative, on these jokes. The use of volunteers to write lexical entries was a way of making the testing slightly more rigorous. We did not have access to a suitable large lexicon, but if we had handcrafted the entries ourselves there would have been the risk of bias (i.e. humour-oriented information) creeping in.

JAPE-1 produced a set of 188 jokes in near-surface form, which were distributed in batches to 14 judges, who gave the jokes scores on a scale from 0 ("Not a joke. Doesn't make any sense.") to 5 ("Really good"). They were also asked for qualitative information, such as how the jokes might be improved, and if they had heard any of the jokes before.

This testing was *not* meant to be statistically rigorous. However, when it comes to analyzing the data, this lack of rigour causes some problems. Because there were so few jokes and joke judges, the scores are not statistically significant. Moreover, there was no control group of jokes. We suspect that jokes of this genre are not very funny even when they are produced by humans; however, we do not know how human-produced jokes would fare if judged in the same way JAPE-1's jokes were, so it is difficult to make the comparison. Ideally, with hindsight, JAPE-1's jokes would then have been mixed with similar jokes (from (Webb 1978), for example), and then all the jokes would have been judged by a group of schoolchildren, who would be less likely to have heard the jokes before and more likely to appreciate them.

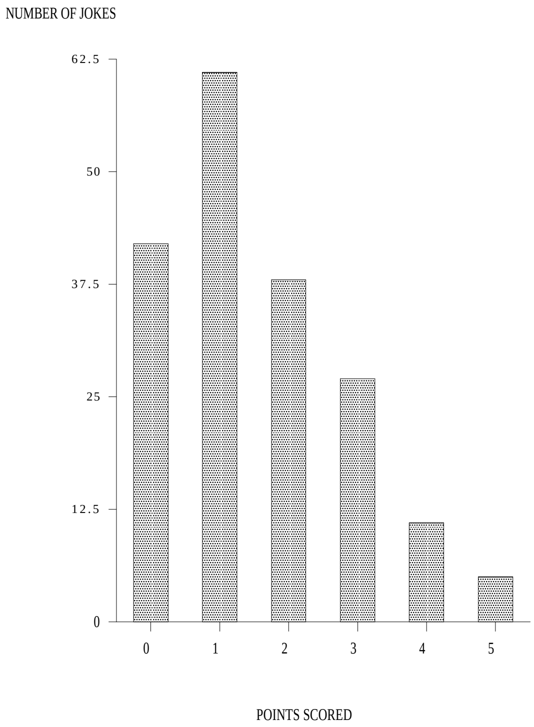

Figure 7: The point distribution over all the output

The results of the testing are summarised in Figure 7. The average point score for all the jokes JAPE-1 produced from the lexical data provided by volunteers is 1.5 points, over a total of 188 jokes. Most of the jokes were given a score of 1. Interestingly, all of the nine jokes that were given the maximum score of five by one judge, were given low scores by the other judge — three got zeroes, three got ones, and three got twos. Overall, the current version of JAPE-1 produced, according to the scores the judges gave, "jokes, but pathetic ones". The top end of the output are definitely of Crack-a-Joke book quality, and some (according to the judges) existed already as jokes, including:

> What do you call a murderer that has fibre? *A cereal killer.*
> What kind of tree can you wear? *A fir coat.*
> What kind of rain brings presents? *A bridal shower.*
> What do you call a good-looking taxi? *A handsome cab.*
> What do you call a perforated relic? *A holey grail.*
> What kind of pig can you ignore at a party? *A wild bore.*
> What kind of emotion has bits? *A love byte.*

It was clear from the evaluation that some schemata and templates tended to produce better jokes than others. For example, the *use_syn* template produced several texts that were judged to be non-jokes, such as:

What do you use to hit a waiting line? *A pool queue.*

The problem with this template is probably that it uses the definition constructed by the schema inappropriately. The schema-generated definition is 'nonsense', in that it describes something that doesn't exist; nonetheless, the word order of the punchline does contain some semantic information (i.e. which of its words is the object and which word describes that object), and it is important for the question to reflect that information. A more appropriate template, *class_has_rev*, produced this joke:

What kind of line has sixteen balls? *A pool queue.*

which the judges gave an average of two points.

Another problem was that the definitions provided by the volunteers were often too general for our purposes. For example, the entry for the word "hanger" gave its class as **device**, producing jokes like:

What kind of device has wings? *An aeroplane hanger.*

which scored half a point.

Conclusions

This evaluation has accomplished two things. It has shown that JAPE-1 can produce pieces of text that are recognizably jokes (if not very good ones) from a relatively unbiased lexicon. More importantly, it has suggested some ways that JAPE-1 could be improved:

- The description of the lexicon could be made more precise, so that it is easier for people unfamiliar with JAPE-1 to make appropriate entries. Moreover, multiple versions of an entry could be compared for 'common knowledge', and that common knowledge entered in the lexicon.
- More slots could be added to the lexicon, allowing the person entering words to specify what a thing is made of, what it uses, and/or what it is part of.
- New, more detailed templates could be added, such as ones which would allow more complex punchlines.
- Templates and schemata that give consistently poor results could be removed.
- The remaining templates could be adjusted so that they use the lexical data more gracefully, by providing the right amount of information in the question part of the riddle.
- Schema-template links that give consistently poor results could be removed.
- JAPE-1 could be extended to handle other joke types, such as simple spoonerisms and sub-word puns.

If even the simplest of the trimming and ordering heuristics described above were implemented, JAPE-1's output would be restricted to good–quality punning riddles. Although there is certainly room for improvement in JAPE-1's performance, it does produce recognizable jokes in accordance with a model of punning riddles, which has not been done successfully by any other program we know of. In that, it is a success.

Acknowledgments

We would like to thank Salvatore Attardo for letting us have access to his unpublished work, and for his comments on the research reported here.

References

Attardo, S., and Raskin, V. 1991. Script theory revis(it)ed: joke similarity and joke representation model. *Humor* 4(3):293–347.

Attardo, S. 1994. *Linguistic Theories of Humour.* Berlin: Mouton de Gruyter.

Binsted, K., and Ritchie, G. 1994. A symbolic description of punning riddles and its computer implementation. Research Paper 688, University of Edinburgh, Edinburgh, Scotland.

Ephratt, M. 1990. What's in a joke. In Golumbic, M., ed., *Advances in AI: Natural Language and Knowledge Based Systems.* Springer Verlag. 43–74.

Minsky, M. 1963. Steps towards artificial intelligence. In Feigenbaum, E., and Feldman, J., eds., *Computers and Thought.* McGraw-Hill. 406–450.

Minsky, M. 1980. Jokes and the logic of the cognitive unconscious. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Palma, P. D., and Weiner, E. J. 1992. Riddles: accessibility and knowledge representation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-92)*, volume 4. 1121–1125.

Pepicello, and Green. 1984. *The Language of Riddles.* Ohio State University.

Townsend, W., and Antworth, E. 1993. *Handbook of Homophones (online version).*

Webb, K., ed. 1978. *The Crack-a-Joke Book.* Puffin.