

TJHSST Computer Systems Lab Senior
Research Project
Generating Puns
2009-2010

Vivaek Shivakumar

January 26, 2010

Abstract

Computational humor is a subfield of artificial intelligence focusing on computer recognition and generation of humorous texts. This paper investigates methods for generating various types of simple puns and jokes using a regular lexicon from the Natural Language Toolkit and the semantic web WordNet as well as phonetic information, which is the basis for most puns. Although no formal model or theory for humor or puns exists, pun-generation has been implemented using simple, constrained models and this paper attempts to recreate such implementations with possible expansions to more types of puns and even languages other than English.

Keywords: computational humor, joke, pun

1 Introduction

One of the main goals of artificial intelligence is natural language generation, meaning that computers should eventually be able to generate meaningful text that would require human intelligence and cognition to produce. One aspect of human language that is of interest to AI is humor. Not only is humor perfect for the goals of computer creativity, but humor generation has

applications for language development and teaching, and is essential for the future of human-computer interaction [1].

The purpose of this project is to investigate and implement methods for generating humor, specifically various types of short puns. Puns include spoonerisms, humorous acronyms, and a variety of other forms. However, there is no strict definition for a pun [2]. Furthermore, puns and humor in general have not been studied to the point of developing a formal model of humor that could be used in AI applicaitons [3]. Nevertheless, work has been done in the area of pun generation since 1994 [4] culminating in advanced user-interface programs such as STANDUP [5].

2 Background

Humor has been studied extensively in a social or otherwise non-AI context. Although much literature on humor and human cognition with respect to humor has indicated patterns and similarities yielding some informal accounts of humor such as the incongruity theory, no formal models or theories yet exist to facilitate the application of humor to AI [3]. Nevertheless, the possibility of computer generation or recognition of humor is by no means impossible. Several computer programs have been created to generate small puns, including JAPE, which was based on certain classes of puns modeled by *schema* and *templates* and utilizing the large lexicon WordNet. Recently, a successful effort called STANDUP was put through to improve JAPE and create a full interactive module for generating puns geared towards “children with complex communication needs (CCN)”. [5]

Other applications of computational humor that have been implemented include a humorous acronym-builder, a “What do you get when you cross” generator, and several joke-recognition applications. [5][6] Finally, not much literature is available for computational humor in other languages. However, the method of modeling types of puns for lexical semantic and syntactic relationships can definitely be applied to languages besides English.

2.1 Puns

Most popular types of puns that could potentially be implemented for generation, e.g. question-answer riddle puns, have some element of combining seemingly unrelated or random elements in a way that plays not necessarily

on the semantics of the words themselves but on the phonetics. Some techniques employed by pronunciation-based puns include rhyming, homonyms, spoonerisms (trading initial sounds in sets of words) and syllable/word substitution based on phonetic similarity. Some examples:

- What do you get when you cross a murderer with a breakfast food? A cereal killer.
- What is the difference between leaves and a car? One you brush and rake, the other you rush and brake.
- Pasteurise: too fast to see. (An example of redefinition wordplay) [7]

Other types of puns and jokes include knock-knock jokes and Tom Swifty puns [7]. Jokes such as “yo mama” jokes or “Chuck Norris facts” on the other hand are based on complex semantic and pragmatic specifications and relationships that are outside the scope of phonetic-pun-based computational humor.

3 Methodology

The first step is to outline and model the various types and subclasses of puns that are to be generated. I have started using the method of *schemata* described in [4] and apparently used in later applications including in [5]. Currently, due to the paucity of types of puns I have outlined, the schemata are generally specific and will probably be generalized later for more efficient implementation. Figure 1 shows an example schema. After a schema is planned, it is implemented in a program that finds a corresponding combination of words, possibly by receiving input for one word to start. The steps after that include applying a proper template and correcting for surface features that may need adjusting due to linguistic structures, e.g. the presence of indefinite articles.

The software used for implementing the pun-generator will be the Natural Language Toolkit, which includes the Carnegie Mellon pronunciation dictionary and the semantic web WordNet. Coding will be done in Python.

4 Results and Analysis

Currently the program succeeds in, given a starting word, finding a set of four “words” according to the schema given in Figure 1 (although currently only homophones are supported and not any similar-sounding words). However, several problems need to be addressed. The WordNet lexicon and the CMU pronouncing dictionary it uses employs both British and American English words and spellings, and for example in the case of homophones such distinctions can lead to false selections of variations of the same word. Proper names, which for the most part are not usable in the types of jokes analyzed here, are included in WordNet as well. Finally, a slew of uncommon nouns, not suitable for simple puns, are also present, giving rise to nonsensical or hard-to-understand combinations.

4.1 Expected Results

The goal is that the program, for each type of pun implemented, will output something that is recognizably humorous, even if any particular human does not find it to be funny. A clear failure for a specific trial occurs if the output is nonsensical, and clear success is indicated by audible human laughter (unless the laughter is directed towards the failure itself). If possible, the English pun-generator will be able to be recreated with an equivalent implementation in another language with an available semantic web.

References

- [1] Binsted, K., Bergen, B., Coulson, S., Nijholt, A., Stock, O., Strapparava, C., ... Manurung, R. (2006). Computational Humor. *Intelligent Systems*, 21(2):59-69.
- [2] Ritchie, G. (2005). Computational mechanisms for pun generation. In *Proceedings of the 10th European Natural Language Generation Workshop*, pages 125-132, Aberdeen.
- [3] Ritchie, G. (2001). Current directions in computational humour. *Artificial Intelligence Review*, 16(2):119-135.

- [4] Binsted, K. and Ritchie, G. (1994). An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 633-638, Seattle, USA.
- [5] Ritchie, G., Manurung, R., Pain, H., Waller, A., Black, R. and O'Mara, D. (2007). A practical application of computational humour. In *Proceedings of the 4th International Joint Conference on Computational Creativity*, pages 91-98, London.
- [6] Johnson, J. (2008, March 1). How to write original jokes (or have a computer do it for you) [Web log post]. Retrieved from <http://grok-code.com/12/how-to-write-original-jokes-or-have-a-computer-do-it-for-you/>
- [7] Puns and other word play. (2001, August 1). *BBC - h2g2*. <http://www.bbc.co.uk/dna/h2g2/A592643>

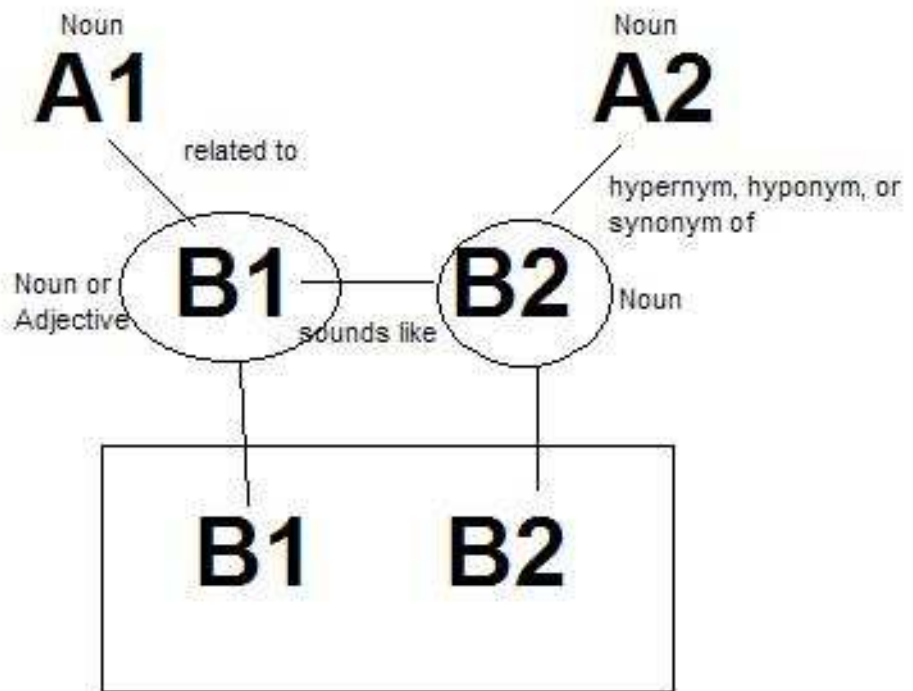


Figure 1: Schema for the “What do you get when you cross” joke