Computational Humor: Automated Pun Generation

Bradley Tyler
Computer Science Department
Idaho State University
tylebrad@isu.com

Katherine Wilsdon
Computer Science Department
Idaho State University
wilskat7@isu.edu

Abstract

Humor is incorporated into our daily interactions, but conceiving jokes ideas can be inherently difficult to produce spontaneously. The PAUL BOT system, "**p**uns **a**re **u**sually **l**ame **b**ut **o**ccasionally **t**errific", includes features of the JAPE system by incorporating aspects of the SAD generator, templates, and sentence forms to automatically generate a pun (Ritchie 2003). A two-word database is composed of adjective-noun phrases that contain a homophone. The punchline to the pun is replacing a word in the two-word phrase with a homophone. A synonym is derived from the homophone, and a hypernym is chosen from the non-homophone word in the original phrase. The synonym and the hypernym are incorporated into a predetermined sentence structure to construct the question. Surveys were conducted to evaluate our artefacts produced by PAUL BOT in order to identify future improvements of the system. The creativity in the system is attributed to the novelty of the unique artefacts, the level of surprise, the typicality of the artefacts as classic puns, and the intentionality by providing the connection between the setup and punchline. This paper proposes the PAUL BOT system that incorporates the JAPE system to create puns in order to produce a humorous, creative system utilized for entertainment.

Introduction

The most successful joke-generation systems have concentrated on pun generation. The JAPE system, "**J**oke **A**nalysis and **P**roduction **E**ngine", exemplifies the a humorous system that generates a wide range of puns that are consistently evaluated as novel and valuable (Ritchie 2003). An improved version of JAPE has since been developed known as the STANDUP system, "**S**ystem **T**o **A**ugment **N**on-speakers' **D**ialog **U**sing **P**uns" that teaches children with communication impairments to tell novel jokes (Waller et al. 2009).

The purpose of JAPE is to produce short texts that are intended to be punning riddles. The fundamental aspects of JAPE are the schemata, sentence forms, templates, and the SAD generation rules (Ritchie 2003). JAPE uses lexemes from riddles, which are a phrases that contain linguistic information about a verb, adjective, or noun. *Schemata* identify the configuration of the lexemes in various riddles. The *SAD generator*, **s**mall **a**dequate **d**escription, uses the information provided by the schema to construct abstract linguistic structures called *SADs* from the lexemes. The SAD generator follows various *SAD rules* that satisfy preconditions for generating linguistic data according to the information provided by the schema. The relations between the lexemes and the derived constituents, called *SAD relations*, are transferred to the template stage. The *template matcher* chooses sentence forms where the conditions are satisfied for insertion. Lastly, the *grammar rule generator* inserts the parameters into slots within the fixed text to produce the punning riddle.

JAPE incorporates so many schemata because there are different lexical preconditions that are possible for each variation of comparable strings. Jokes typically have the same sentence structure, but because of the grammatically diverse English language separate schemata are used.

We have designed a computational humorous system called PAUL BOT, "**p**uns **a**re **u**sually **l**ame **b**ut **o**ccasionally **t**errific". Our main interest of PAUL BOT is to create puns that users find humorous in some way. In this paper, we focus on the design of the PAUL BOT system and the analysis of the artefacts produced.

Much of the design of PAUL BOT relies on previous work in the joke generation field. Our system filters a two-word database for adjective-noun phrases which circumvents the need for identifying the linguistic information about a particular phrase in schema.

Similarly to JAPE's SAD generator, the system constructs abstract linguistic structures from the two-word phrase (Ritchie 2003). The punchline to the joke replaces a word in the original two-word phrase with a homophone via the 2-Gram Database from Corpus of Contemporary American English (COCA) (Weide 2005). The setup to the pun uses WordNet to identify relationships between words on the basis of synonyms, antonyms, and hypernyms (Fellbaum 2012).

Finally, PAUL BOT chooses a template depending on whether the synonym/antonym or hypernym is a verb and whether the sentence structure should be negated for an antonym. Like the template matcher in JAPE, these conditions need to be satisfied in order to choose the correct format of the question (Ritchie 2003). Then, the synonym/antonym and hypernym are inserted into the slots within the chosen template and the appropriate article is added to the noun, which is comparable to the grammar rule generator.

Methods

In this section we describe the design and operation of the PAUL BOT system. We first provide a high-level overview of the system. Then, the two-word database, homophone dictionary, synonyms, antonyms, hypernyms, and templates are explored further in detail. Finally, we define the metrics for our user evaluation and special package installations for Python.

PAUL BOT System Overview

Taking an input of a two-word phrase, the system transforms the phrase using its semantic relations with other words to output a humorous pun. The flow of information through PAUL BOT is as follows, Figure 1.

1. Choose a two-word phrase from a database of adjective noun phrases as the input into the system.
2. Identify a homophone within the two-word phrase.
3. Replace a word in the phrase with the homophone to form the answer.
4. Identify a synonym or antonym for the homophone.
5. Identify a hypernym for the non-homophone word in the original phrase.
6. Identify the appropriate template that satisfies the parts of speech and the negation.
7. Insert the synonym or antonym and the hypernym into the slots of the template.
8. Output the generated pun.

The following demonstrates an example of the flow of information through the system. The chosen two-word phrase input is electric motor. A homophone for motor is voter. A synonym for voter is elector. A hypernym for motor is car through a part-of relationship. The output is

What do you get when you cross a car with an elector?
electric voter

Two-word Database Synopsis

A 2-Gram database of two-word phrases was obtained from COCA, **C**orpus **O**f **C**ontemporary **A**merican English (Davies 2014). The database included the two words, their associated parts of speech, and the frequency of the phrase in the English language. The punchline to the joke is an adjective-noun combination of words, so we filtered the database for phrases that only contained an adjective followed by a noun.

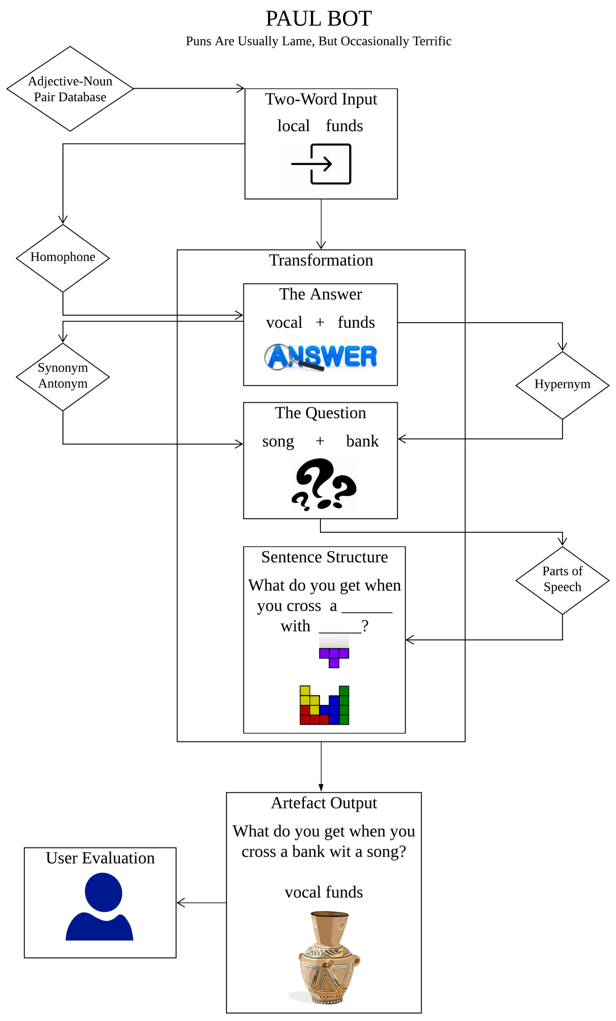

Figure 1: An overview of the PAUL BOT system. Given a two-word phrase, the system determines a homophone within the phrase, chooses a synonym for the homophone, finds a hypernym for the non-homophone word, and selects a template that satisfies all the conditions to produce a pun.

Homophone Examination

The CMUdict, a dictionary of homophones, was retrieved from Carnegie Mellon University (Weide 2005). The dictionary is composed of words and their phonetic spellings. The Levenshtein distance was used to measure the minimum number of single-character edits (insertions, deletions, or substitutions) to change one word into the other. Our algorithm used the Levenshtein distance to measure the difference between the phonetic spellings of two words. If a word is within one edit distance of the search word, this word is considered a homophone. The homophone with the highest frequency in the English language is chosen via the *wordfreq* package.

Synonyms, Antonyms, and Hypernyms Overview

WordNet was utilized for grouping words by semantic relations for synonyms, antonyms, and hypernyms (Fellbaum 2012). The synonyms and antonyms are found in the lemmas of synsets which are data elements considered to be semantically equivalent. Our algorithm uses the *wordnet* module of the *nltk* package to decipher synonyms and antonyms of a word. WordNet was used for obtaining type-of relationships for hypernyms. The system supports PartOf, IsA, HasA, UsedFor, FormOf, RelatedTo, CapableOf, AtLocation, Causes, and HasSubevent relationships. The hypernym and synonym or antonym with the highest frequency in the English language is chosen. Credit to Andrew Christiansen and and Andres Sewell for the hypernym section within our system.

Template Identification

The template is chosen depending on two factors: the parts of speech and the negation. The hypernyms, synonyms, or antonyms must be either an adjective, noun, or verb. If one of these is a verb, the ”What do you _ that is _?” template is used. Otherwise, the chosen template is ”What do you get when you cross _ with _?”. The negation of these templates are included to accommodate antonyms.

Evaluation Metrics

The PAUL BOT system was run consecutively to produce 150 artefacts. From these artefacts, the top ten jokes were chosen to be evaluated by users. Ten users evaluated the puns on a scale of 1 to 5 on their funniness, surprise, cleverness, did the user laugh, wit, ingenuity, timelessness, and accessibility.

Installation of Packages

The Levenshtein, nltk, and wordfreq packages were add to our local Python installations. The *Levenshtein* package measures the edit distance between the phonetic spellings of homophones. The *nltk* package provides access to WordNet to determine synonyms and antonyms. Lastly, the *wordfreq* package provides the frequencies of words in the English language.

Results

A survey of jokes was administered to ten participants to assess the creativity of the PAUL BOT system. Ten jokes were rated on a scale of 1 to 5 on their *funniness*, *surprise*, *cleverness*, *if the user laughed*, *wit*, *ingenuity*, *timelessness*, and *accessibility*.

The average evaluation of the artefacts across all the features is shown in Table 1. The overall evaluations of the puns range from 2.21 to 3.30. The highest rated pun was ”What do you get when you cross a training with a predator? corporate vultures” with an average rating of 3.3 / 5.0. The lowest evaluated pun with an average evaluation of 2.2 / 5.0 was ”What do you get when you cross a woman with full? wide women.”

Average Rating of Jokes	
What do you get when you cross a people with a fire? human burning	2.73
What do you get when you cross a time with a self-will? temporary possession	2.55
What do you get when you cross a mother with a sign? uterine signing	3.19
What do you get when you cross a living with a jerk? professional yanks	2.83
What do you get when you cross large with a joke? muscular jest	2.73
What do you get when you cross a training with a predator? corporate vultures	3.30
What do you get when you cross a butter with a cycle? buttered biking	2.84
What do you have that is bad? tough sex	2.54
What do you create that is big? large shit	3.10
What do you get when you cross a woman with full? wide women	2.21

Table 1: The average evaluation of artefacts on a scale of 1 to 5.

The median evaluation of the funniness, surprise, cleverness, if the user laughed, wit, ingenuity, timelessness, and accessibility of the jokes is shown in Figure 2. Out of all the features of a joke, the artefacts were the most surprising with a 3.35/5.00 evaluation. Unfortunately, the artefacts performed the worst in making the user laugh with a median of 2.0/5.0 and an average of 2.6/5.0. The rest of the features of a joke performed marginally above the midpoint with a rating of 3.0/5.0.

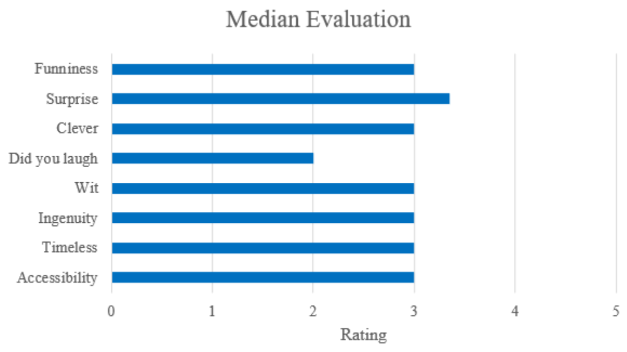

Figure 2: The median evaluation of the ten jokes according to funniness, surprise, cleverness, if the user laughed, wit, ingenuity, timelessness, and accessibility on a scale of 1 to 5.

According to Table 1, the highest rated pun was, ”What do you get when you cross a training with a predator? corporate vultures.” Shown in Figure 3, the user attributed this pun to be clever, witty, and accessible with a median evaluation of 4.0 / 5.0. The average user thought this pun was funny with a 3.5/5.0 evaluation. However, this pun did not make the user laugh with an overall 2.0/5.0 rating.

The lowest evaluated pun was ”What do you get when you

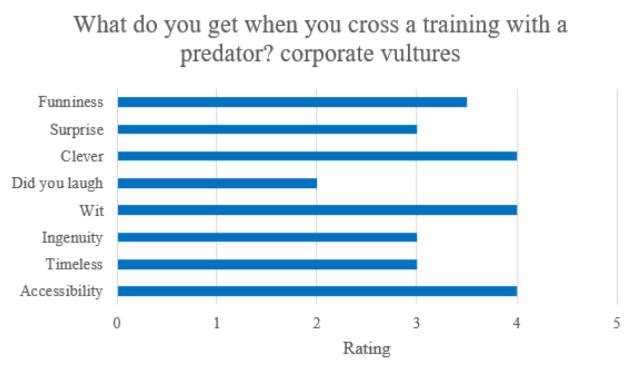

Figure 3: The best joke evaluation according to funniness, surprise, cleverness, if the user laughed, wit, ingenuity, timelessness, and accessibility on a scale of 1 to 5.

cross a woman with full? wide women" as shown in Table 1. Figure 4 shows the underwhelming performance of the joke. The users thought this joke was not clever with a 1.0/5.0 evaluation. Most of the users thought the pun lacked wit, ingenuity, and timelessness and did not make them laugh with a score of 1.5/5.0. Overall, the funniness and surprise of the joke were marginally better with a 2.0/5.0 rating. This artefact performed above the midpoint on accessibility with a median score of 3.0/5.0.

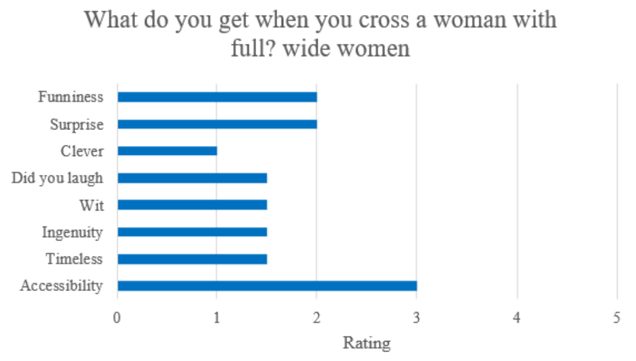

Figure 4: The worst joke evaluation according to funniness, surprise, cleverness, if the user laughed, wit, ingenuity, timelessness, and accessibility on a scale of 1 to 5.

Discussion and Conclusion

In this paper, we explored the design of the PAUL BOT system and the analysis of the artefacts produced. We discovered that our puns usually did not make the user laugh. However, the users were usually surprised by the joke. The funniness, cleverness, wit, ingenuity, timelessness, and accessibility of the pun were marginally better than midpoint.

We believe that PAUL BOT is a computational creative system that exemplifies novelty, value, typicality, and surprise with limited intentionality. PAUL BOT exhibits P-creativity through its creation of puns without a prior knowledge base of existing puns (Wiggins 2006). The artefacts produced are unique due to the transformation of words and therefore novel. The worth or value of the artefacts is moderate because users found the puns to be marginally funny. Our system has high typicality because the artefacts produced by our system represent a ordinary, classic pun. Our system has limited intentionality because the artefact is framed by showing the connection between the setup and the punchline shown in Figure 1. Our puns have a high level of surprise as shown in Figure 2. Because of these points, we believe the PAUL BOT system meets the criteria for generalization (Ventura 2016). Our system does have a fitness function for generating a pun. The word with the highest frequency is chosen from various word lists including homophones, hypernyms, and synonyms/antonyms. However, there is no fitness function for choosing the best puns that are above a certain threshold. Therefore, we argue that PAUL BOT is in between the generalization and filtration stages on a scale of being merely generative to being creative as shown in Figure 5.

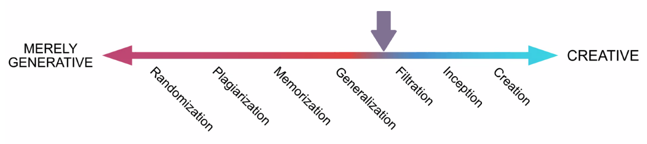

Figure 5: On a scale from merely generative to creative, PAUL BOT is in between generalization and filtration.

Some of the limitations of PAUL BOT are the diversity of the templates, when to add an article to a noun, contains adult content, and does not incorporate homonyms. Our system is limited to only having four templates for the question where the hypernym and synonym/antonym can be inserted into the slots. The worst joke could be improved by adding a template in the following format, "What do you call a woman that is full? wide women." Our system also has loose constraints on adding an article to a noun by misplacing an article on a plural noun, e.g. a people. The breath of topics of our puns narrows the audience to only adults. Children are not advised to use PAUL BOT. Our system currently does not support the use of homonyms, words that have the same spelling or pronunciation but different meanings, only homophones. As seen in Figure 2, our system cannot produce a pun that is liked by every user.

Despite these limitations, PAUL BOT will be developed further to provide humor for adults. Our next steps are to accommodate more templates, refine the algorithm for adding articles to nouns, and incorporate homonym choices. Our goal is to create a computational humorous system that is capable of making users laugh.

References

Davies, M. 2014. N-grams data from the corpus of contemporary american english (coca).

Fellbaum, C. 2012. Wordnet. *The encyclopedia of applied linguistics*.

Ritchie, G. 2003. The jape riddle generator: technical specification. *Institute for Communicating and Collaborative Systems*.

Ventura, D. 2016. Mere generation: Essential barometer or dated concept. In *Proceedings of the Seventh International Conference on Computational Creativity*, 17–24. Sony CSL, Paris.

Waller, A.; Black, R.; O'Mara, D. A.; Pain, H.; Ritchie, G.; and Manurung, R. 2009. Evaluating the standup pun generating software with children with cerebral palsy. *ACM Transactions on Accessible Computing (TACCESS)* 1(3):1–27.

Weide, R. 2005. The carnegie mellon pronouncing dictionary [cmudict. 0.6]. *Pittsburgh, PA: Carnegie Mellon University*.

Wiggins, G. A. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19(7):449–458.