Automatically Extracting Word Relationships as Templates for Pun Generation

Bryan Anthony Hong and **Ethel Ong**
College of Computer Studies
De La Salle University
Manila, 1004 Philippines
bashx5@yahoo.com, ethel.ong@delasalle.ph

Abstract

Computational models can be built to capture the syntactic structures and semantic patterns of human punning riddles. This model is then used as rules by a computer to generate its own puns. This paper presents T-PEG, a system that utilizes phonetic and semantic linguistic resources to automatically extract word relationships in puns and store the knowledge in template form. Given a set of training examples, it is able to extract 69.2% usable templates, resulting in computer-generated puns that received an average score of 2.13 as compared to 2.70 for human-generated puns from user feedback.

1 Introduction

Previous works in computational humor have shown that by analyzing the syntax and semantics of how humans combine words to produce puns, computational models can be built to capture the linguistic aspects involved in this creative wordplay. The model is then used in the design of computer systems that can generate puns which are almost at par with those of human-generated puns, as the case of the Joke Analysis and Production Engine or JAPE (Binsted et al, 1997) system.

The computational model used by the JAPE (Binsted, 1996) system is in the form of schemas and templates with rules describing the linguistic structures of human puns. The use of templates in NLP tasks is not new. Information extraction systems (Muslea, 1999) have used templates as rules for extracting relevant information from large, unstructured text. Text generation systems use templates as linguistic patterns with variables (or slots) that can be filled in to generate syntactically correct and coherent text for their human readers.

One common characteristic among these NLP systems was that the templates were constructed manually. This is a tedious and time-consuming task. Because of this, several researches in example-based machine translation systems, such as those in (Cicekli and Güvenir, 2003) and in (Go et al, 2007), have worked on automatically extracting templates from training examples. The learned templates are bilingual pairs of patterns with corresponding words and phrases replaced with variables. Each template is a complete sentence to preserve the syntax and word order in the source text, regardless of the variance in the sentence structures of the source and target languages (Nunez et al, 2008).

The motivation for T-PEG (Template-Based Pun Extractor and Generator) is to build a model of human-generated puns through the automatic identification, extraction and representation of the word relationships in a template, and then using these templates as patterns for the computer to generate its own puns. T-PEG does not maintain its own lexical resources, but instead relies on publicly available lexicons, in order to perform these tasks. The linguistic aspects of puns and the resources utilized by T-PEG are presented in Section 2.

Sections 3 and 4 discuss the algorithms for extracting templates and generating puns, respectively. The tests conducted and the analysis of the results on the learned templates and generated puns follow in Section 5, to show the limitations of T-PEG's approach and the level of humor in the generated puns. The paper concludes with a summary of what T-PEG has been able to accomplish.

Proceedings of the NAACL HLT Workshop on Computational Approaches to Linguistic Creativity, pages 24–31,
Boulder, Colorado, June 2009.

2 Linguistic Resources

Ritchie (2005) defines a pun as "a humorous written or spoken text which relies crucially on phonetic similarity for its humorous effect". Puns can be based on inexact matches between words (Binsted and Ritchie, 2001), where tactics include metathesis (e.g., *throw stones* and *stow thrones*) and substitution of a phonetically similar segment (e.g., *glass* and *grass*).

In T-PEG, punning riddles are considered to be a class of jokes that use wordplay, specifically pronunciation, spelling, and possible semantic similarities and differences between words (Hong and Ong, 2008). Only puns using the question - answer format as shown in example (1) from (Binsted, 1996) are considered. Compound words are also included, underlined in example (2) from (Webb, 1978).

(1) What do you call a beloved mammal?
A dear deer.
(2) What do barbers study? Short-cuts.

The automatic tasks of analyzing human-generated puns in order to build a formal model of the word relationships present in the puns require the use of a number of linguistic resources. These same set of resources are used for later generation. STANDUP (Manurung et al, 2008), for example, uses "a database of word definitions, sounds and syntax to generate simple play-on-words jokes, or puns, on a chosen subject". Aside from using WordNet (2006) as its lexical resource, STANDUP maintains its own lexical database of phonetic similarity ratings for pairs of words and phrases.

Various works have already emphasized that puns can be generated by distorting a word in the source pun into a similar-sounding pun, e.g., (Ritchie, 2005 and Manurung et al, 2008). This notion of phonetic similarity can be extended further by allowing puns containing words that sound similar to be generated, as shown in example (3), which was generated by T-PEG following the structure of (1).

(3) What do you call an overall absence?
A whole hole.

The Unisyn English Pronunciation lexicon (Fitt, 2002) was utilized for this purpose. The dictionary contains about 70,000 entries with phonetic transcriptions and is used by T-PEG to find the pronunciation of individual words and to locate similar sounding words for a given word. Because Unisyn also provides support in checking for spelling regularity, it is also used by T-PEG to check if a given word does exist, particularly when a compound word is split into its constituent syllables and determining if these individual syllables are valid words, such as the constituents "*short*" and "*cuts*" for the compound word "*shortcuts*" in (2).

The wordplay in punning riddles is not based on phonetic similarity alone, but may also involve the semantic links among words that make up the pun. These semantic relationships must also be identified and captured in the template, such that the generated puns are not only syntactically well-formed (due to the nature of templates) but also have consistent semantics with the source human pun, as shown in example (4) from (Binsted, 1996) and T-PEG's counterpart in example (5).

(4) How is a car like an elephant?
They both have trunks.
(5) How is a person like an elephant?
They both have memory.

Two resources are utilized for this purpose. WordNet (2006) is used to find the synonym of a given word, while ConceptNet (Liu and Singh, 2004) is used to determine the semantic relationships of words.

ConceptNet is a large-scale common sense knowledge base with about 1.6 million assertions. It focuses on contextual common sense reasoning, which can be used by a computer to understand concepts and situating these concepts on previous knowledge.

Relationship Types	Examples
IsA	IsA *headache pain* IsA *deer mammal*
PartOf	PartOf *window pane* PartOf *car trunk*
PropertyOf	PropertyOf *pancake flat* PropertyOf *ghost dead*
MadeOf	MadeOf *snowman snow*
CapableOf	CapableOf *sun burn* CapableOf *animal eat*
LocationOf	LocationOf *money bank*
CanDo	CanDo *ball bounce*
ConceptuallyRelatedTo	ConceptuallyRelatedTo *wedding bride* *forest animal*

Table 1. Some Semantic Relationships of ConceptNet (Liu and Singh, 2004)

The concepts can be classified into three general classes – noun phrases, attributes, and activity phrases, and are connected by edges to form an ontology. Binary relationship types defined by the Open Mind Commonsense (OMCS) Project (Liu and Singh, 2004) are used to relate two concepts together, examples of which are shown in Table 1.

3 Extracting Punning Templates

The structural regularities of puns are captured in T-PEG with the use of templates. A template is the combined notion of schemas and templates in (Binsted, 2006), and it contains the relationship between the words (lexemes) in a pun as well as its syntactical structure. The template constrains the set of words that can be used to fill-in the slots during the generation phase; it also preserves the syntactical structure of the source pun, to enable the generated puns to follow the same syntax.

3.1 Templates in T-Peg

A template in T-PEG is composed of multiple parts. The first component is the source punning riddle, where variables replaced the keywords in the pun and also serve as slots that can be filled during the pun generation phase.

Variables can be one of three types. A *regular variable* is a basic keyword in the source pun whose part-of-speech tag is a noun, a verb, or an adjective. Variables in the question-part of the pun are represented with *Xn* while *Yn* represent variables in the answer-part (where *n* denotes the lexical sequence of the word in the sentence starting at index 0).

A *similar-sound variable* represents a word that has the same pronunciation as the regular variable, for example, *deer* and *dear*. A *compound-word variable* contains two regular or similar-sound variables that combine to form a word, for example *sun* and *burn* combine to form the word *sunburn*. A colon (:) is used to connect the variables comprising a compound variable, for example, *X1:X2*.

Word relationships may exist among the variables in a pun. These word relationships comprise the second component of a template and are represented **<var1> <relationship type> <var2>**.

There are four types of binary word relationships captured by T-PEG. *SynonymOf relationships* specify that two variables are synonymous with each other, as derived from WordNet (2006). *Compound-word (or IsAWord) relationships* specify that one variable combined with a second variable should form a word. Unisyn (Fiit, 2002) is used to check that the individual constituents as well as the combined word are valid. *SoundsLike relationships* specify that two variables have the same pronunciation as derived from Unisyn. *Semantic relationships* show the relationships of two variables derived from ConceptNet (Liu and Singh, 2004), and can be any one of the relationship types presented in Table 1.

3.2 Learning Algorithm

Template learning begins from a given corpus of training examples that is preprocessed by the tagger and the stemmer. The tagged puns undergo valid word selection to identify keywords (noun, verb, or adjective) as candidate variables. The candidate variables are then paired with each other to identify any word relationships that may exist between them. The word relationships are determined by the phonetic checker, the synonym checker, and the semantic analyzer. Only those candidate variables with at least one word relationship with another candidate variable will be retained as final variables in the learned template.

Table 2 presents the template for "*Which bird can lift the heaviest weights? The crane.*" (Webb, 1978). Keywords are underlined. All of the extracted word relationships in Table 2 were derived from ConceptNet. Notice that i) some word pairs may have one or more word relationships, for example, "*crane*" and "*lift*"; while ii) some candidate keywords may not have any relationships, i.e, the adjective "*heaviest*", thus it is not replaced with a variable in the resulting template. This second condition will be explored further in Section 5.

Source Pun	Which bird can lift the heaviest weights? The crane.
Template	Which <X1> can <X3> the heaviest <X6>? The <Y1>.
Word Relationships	X1 ConceptuallyRelatedTo X6 X6 ConceptuallyRelatedTo X1 Y1 IsA X1 X6 CapableOfReceivingAction X3 Y1 CapableOf X3 Y1 UsedFor X3

Table 2. Template with *Semantic* Relationships identified through ConceptNet

Table 3 presents another template from the pun "*What do you call a beloved mammal? A dear deer.*" (Binsted, 1996), with the *SynonymOf* relationship derived from WordNet, the *IsA* relationship from ConceptNet, and the *SoundsLike* relationship from Unisyn. Notice the "-0" suffix in variables *Y1* and *Y2*. "<*var*>-*0*" is used to represent a word that is phonetically similar to <*var*>.

Source Pun	What do you call a beloved mammal? A dear deer.
Template	What do you call a <X5> <X6>? A <Y1> <Y2>.
Word Relationships	X5 **SynonymOf** Y1 X5 **SynonymOf** Y2-0 Y1-0 IsA X6 Y2 IsA X6 Y1 **SoundsLike** Y2 Y1-0 **SoundsLike** Y1 Y2-0 **SoundsLike** Y2

Table 3. Template with *Synonym* Relationships and *Sounds-Like* Relationships

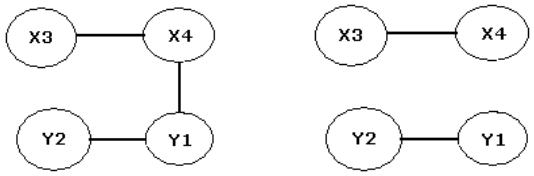
A constituent word in a compound word (identified through the presence of a dash "-") may also contain additional word relationships. Thus, in "*What kind of fruit fixes taps? A plum-ber.*" (Binsted, 1996), T-PEG learns the template shown in Table 4. The compound word relationship extracted is *Y1 IsAWord Y2* (*plum* IsAWord *ber*). *Y1* (*plum*), which is a constituent of the compound word, has a relationship with another word in the pun, *X3* (*fruit*).

Source Pun	What kind of fruit fixes taps? A plum-ber.
Template	What kind of <X3> <X4> taps? A <Y1>:<Y2>.
Word Relationships	Y1 IsA X3 Y1 **IsAWord** Y2 **Y1:Y2** CapableOf X4

Table 4. Template with Compound Word

The last phase of the learning algorithm involves template usability check to determine if the extracted template has any missing link. A template is usable if all of the word relationships form a connected graph. If the graph contains unreachable node/s (that is, it has missing edges), the template cannot be used in the pun generation phase since not all of the variables will be filled with possible words.

Consider a template with four variables named *X3*, *X4*, *Y1* and *Y2*. The word relationships *X3-X4*, *X4-Y1* and *Y1-Y2* form a connected graph as shown in Figure 1(a). However, if only *X3-X4* and *Y1-Y2* relationships are available as shown in Figure 1(b), there is a missing edge such that if variable *X3* has an initial possible word and is the starting point for generation, a corresponding word for variable *X4* can be derived through the *X3-X4* edge, but no words can be derived for variables *Y1* and *Y2*.



(a) Connected Graph (b) Graph with Missing Edge

Figure 1. Graphs for Word Relationships

This condition is exemplified in Table 5, where two disjoint subgraphs are created as a result of the missing "*house-wall*" and "*wall-wal*" relationships. Further discussion on this is found in Section 5.

Source Pun	What nuts can you use to build a house? Wal-nuts. (Binsted, 1996)
Template	What <X1> can you use to <X6> a <X8>? <Y0>-<Y1>.
Word Relationships	X8 CapableOfReceivingAction X6 X1 SoundsLike Y1 Y0 IsAWord Y1 Y0:Y1 IsA X1
Missing Relations	Y0-0 PartOf X8 Y0-0 SoundsLike Y0

Table 5. Template with Missing Word Relationships where *Y0-0* is the word "*wall*"

4 Generating Puns from Templates

The pun generation phase, having access to the library of learned templates and utilizing the same set of linguistic resources as the template learning algorithm, begins with a keyword input from the user. For each of the usable templates in the library, the keyword is tested on each variable with the same POS tag, except for *SoundsLike* and *IsA-Word* relationships where tags are ignored. When a variable has a word, it is used to populate other variables with words that satisfy the word relationships in the template.

T-PEG uses two approaches of populating the variables – forward recursion and backward recursion. *Forward recursion* involves traversing the graph by moving from one node (variable in a template) to the next and following the edges of relationships. Consider the template in Table 6.

Human Joke	How is a window like a headache? They are both panes. (Binsted, 1996)
Template	How is a <X3> like a <X6>? They are both <Y3>.
Word Relationships	Y3-0 SoundsLike Y3 X3 ConceptuallyRelatedTo Y3 Y3 ConceptuallyRelatedTo X3 Y3 PartOf X3 X6 ConceptallyRelatedTo Y3-0 X6 IsA Y3-0 Y3-0 ConceptuallyRelatedTo X6

Table 6. Sample Template for Pun Generation

Given the keyword "*garbage*", one possible sequence of activities to locate words and populate the variables in this template is as follows:

a. "*garbage*" is tried on variable *X6*.
b. *X6* has three word relationships all of which are with *Y3-0*, so it is used to find possible words for *Y3-0*. ConceptNet returns an "*IsA*" relationship with the word "*waste*".
c. *Y3-0* has only one word relationship and this is with *Y3*. Unisyn returns the phonetically similar word "*waist*".
d. *Y3* has two possible relationships with *X3*, and ConceptNet satisfies the "*PartOf"* relationship with the word "*trunk*".

Since two variables may have more than one word relationships connecting them, relationship grouping is also performed. A word relationship group is said to be *satisfied* if at least one of the word relationships in the group is satisfied. Table 7 shows the relationship grouping and the word relationship that was satisfied in each group for the template in Table 6.

Word Relationship	Filled Template
X6 ConceptallyRelatedTo Y3-0 X6 IsA Y3-0 Y3-0 ConceptuallyRelatedTo X6	*garbage* IsA *waste*
Y3-0 SoundsLike Y3	*waste* SoundsLike *waist*
X3 ConceptuallyRelatedTo Y3 Y3 ConceptuallyRelatedTo X3 Y3 PartOf X3	*waist* PartOf *trunk*

Table 7. Relationship Groups and Filled Template

The filled template is passed to the surface realizer, LanguageTool (Naber, 2007), to fix grammatical errors, before displaying the resulting pun "*How is a trunk like a garbage? They are both waists*." to the user.

The forward recursion approach may lead to a situation in which a variable has been filled with two different sets of words. This usually occurs when the graph contains a cycle, as shown in Figure 2.

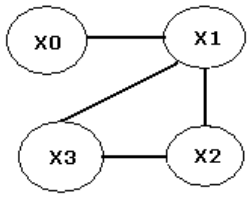


Figure 2. Graph with Cycle

Assume the process of populating the template begins at *X0*. The following edges and resulting set of possible words are retrieved in sequence:

a. *X0-X1* (Words retrieved for X1 ➔ A, B)
b. *X1-X2* (Words retrieved for X2 ➔ D, E, F)
c. *X2-X3* (Words retrieved for X3 ➔ G, H)
d. *X3-X1* (Words retrieved for X1 ➔ B, C)

When the forward recursion algorithm reaches *X3* in step (d), a second set of possible words for *X1* is generated. Since the two sets of words for *X1* do not match, the algorithm gets the intersection of (*A*, *B*) and (*B*, *C*) and assigns this to *X1* (in this case, the word "*B"* is assigned to *X1*). Backward recursion has to be performed starting from step (b) using the new set of words so that other variables with relationships to *X1* will also be checked for possible changes in their values.

5 Test Results

Various tests were conducted to validate the completeness of the word relationships in the learned template, the correctness of the generation algorithm, and the quality of the generated puns.

5.1 Evaluating the Learned Templates

The corpus used in training T-PEG contained 39 punning riddles derived from JAPE (Binsted, 1996) and The Crack-a-Joke Book (Webb, 1978). Since one template is learned from each source

pun, the size of the corpus is not a factor in determining the quality of the generated jokes.

Of the 39 resulting templates, only 27 (69.2%) are usable. The unusable templates contain missing word relationships that are caused by two factors. Unisyn contains entries only for valid words and not for syllables. Thus, in (6), the relationship between "*house*" and "*wall*" is missing in the learned template shown in Table 5 because "*wal*" is not found in Unisyn to produce "*wall*". In (7), ConceptNet is unable to determine the relationship between "*infantry*" and "*army*".

(6) What nuts can you use to build a house?
Wal-nuts. (Binsted, 1996)

(7) What part of the army could a baby join?
The infant-ry. (Webb, 1978)

The generation algorithm relies heavily on the presence of correct word relationships. 10 of the 27 usable templates were selected for manual evaluation by a linguist to determine the completeness of the extracted word relationships. A template is said to be *complete* if it is able to capture the essential word relationships in a pun. The evaluation criteria are based on the number of incorrect relationships as identified by the linguist, and includes missing relationship, extra relationship, or incorrect word pairing. A scoring system from 1 to 5 is used, where 5 means there are no incorrect relationship, 4 means there is one incorrect relationship, and so on.

The learning algorithm received an average score of 4.0 out of 5, due to missing word relationships in some of the templates. Again, these were caused by limitations of the resources. For example, in (8), the linguist noted that no relationship between "*heaviest*" and "*weight*" (i.e., PropertyOf *heavy weight*) is included in the learned template presented in Table 2.

(8) What bird can lift the heaviest weights?
The crane. (Webb, 1978)

(9) What kind of fruit fixes taps?
The plum-ber. (Binsted, 1996)

In (9), the linguist identified a missing relationship between "*tap*" and "*plumber*", which is not extracted by the template shown in Table 4.

The linguist also noted that the constituents of a compound word do not always form valid words, such as "*ber*" in *plum-ber* of pun (9), and "*wal*" in *wal-nuts* of pun (6). This type of templates were considered to contain incorrect relationships, and they may cause problems during generation because similar sounding words could not be found for the constituent of the compound word that is not a valid word.

5.2 Evaluating the Generation Algorithm

The generation algorithm was evaluated on two aspects. In the first test, a keyword from each of the source puns was used as input to T-PEG to determine if it can generate back the training corpus. From the 27 usable templates, 20 (74.07%) of the source puns were generated back. Regeneration failed in cases where a word in the source pun has multiple POS tags, as the case in (10), where "*cut*" is tagged as a noun during learning, but verb during generation. In the learning phase, tagging is done at the sentence level, as opposed to a single-word tagging in the generation phase.

(10) What do barbers study? Short-cuts.
(Webb, 1978)

Since a keyword is tried on each variable with the same POS tag in the template, the linguistic resources provided the generation algorithm with a large set of possible words. Consider again the pun in (10), using its template and given the keyword "*farmer*" as an example, the system generated 122 possible puns, some of which are listed in Table 8. Notice that only a couple of these seemed plausible puns, i.e., #3 and #7.

1.	What do *farmers* study? Egg - plant.
2.	What do *farmers* study? Power - plant.
3.	What do *farmers* study? Trans - plant.
4.	What do *farmers* study? Battle - ground.
5.	What do *farmers* study? Play - ground.
6.	What do *farmers* study? Battle - field.
7.	What do *farmers* study? Gar - field.

Table 8. Excerpt of the Generated Puns Using "*farmer*" as Keyword

In order to find out how this affects the overall performance of the system, the execution times in locating words for the different types of word relationships were measured for the set of 20 regenerated human puns. Table 9 shows the summary for the running time and the number of word relationships extracted for each relationship type.

Another test was also conducted to validate the previous finding. A threshold for the maximum

number of possible words to be generated was set to 50, resulting in a shorter running time as depicted in Table 10. A negative outcome of using a threshold value is that only 16 (instead of 20) human puns were regenerated. The other four cases failed because the threshold became restrictive and filtered out the words that should be generated.

Relationship Type	Running Time	# Relationships
Synonym	2 seconds	2
IsAWord	875 seconds	5
Semantic	1,699 seconds	82
SoundsLike	979 seconds	8

Table 9. Running Time of the Generation Algorithm

Relationship Type	Running Time	# Relationships
Synonym	2 seconds	2
IsAWord	321 seconds	4
Semantic	315 seconds	57
SoundsLike	273 seconds	8

Table 10. Running Time of the Generation Algorithm with Threshold = 50 Possible Words

5.3 Evaluating the Generated Puns

Common words, such as *man*, *farmer*, *cow*, *garbage*, and *computer*, were fed to T-PEG so that the chances of these keywords being covered by the resources (specifically ConceptNet) are higher. An exception to this is the use of keywords with possible homonyms (i.e., *whole* and *hole*) to increase the possibility of generating puns with *SoundsLike* relationships.

As previously stated, the linguistic resources provided the generation algorithm with various words that generated a large set of puns. The proponents manually went through this set, identifying which of the output seemed humorous, resulting in the subjective selection of eight puns that were then forwarded for user feedback.

User feedback was gathered from 40 people to compare if the puns of T-PEG are as funny as their source human puns. 15 puns (7 pairs of human-T-PEG puns, with the last pair containing 1 human and 2 T-PEG puns) were rated from a scale of 0 to 5, with 5 being the funniest. This rating system was based on the joke judging process used in (Binsted, 1996), where 0 means it is not a joke, 1 is a pathetic joke, 2 is a "not-so-bad" joke, 3 means average, 4 is quite funny, and 5 is really funny.

T-PEG puns received an average score of 2.13 while the corresponding source puns received an average score of 2.70. Table 11 shows the scores of four pairs of punning riddles that were evaluated, with the input keyword used in generating the T-PEG puns enclosed in parentheses. Pun evaluation is very subjective and depends on the prior knowledge of the reader. Most of the users involved in the survey, for example, did not understand the relationship between *elephant* and *memory*[1], accounting for its low feedback score.

Training Pun	T-Peg Generated Pun
What keys are furry? Mon-keys. (Webb, 1978) (2.93)	What verses are endless? Uni-verses. (Keyword: verses) (2.73)
What part of a fish weighs the most? The scales. (Webb, 1978) (3.00)	What part of a man lengthens the most? The shadow. (Keyword: man) (2.43)
What do you call a lizard on the wall? A rep-tile. (Binsted, 1996) (2.33)	What do you call a fire on the floor? A fire-wood. (Keyword: fire) (1.90)
How is a car like an elephant? They both have trunks. (Binsted, 1996) (2.50)	How is a person like an elephant? They both have memory. (Keyword: elephant) (1.50)

Table 11. Sample Puns and User Feedback Scores

Although the generated puns of T-PEG did not receive scores that are as high as the puns in the training corpus, with an average difference rating of 0.57, this work is able to show that the available linguistic resources can be used to train computers to extract word relationships in human puns and to use these learned templates to automatically generate their own puns.

6 Conclusions

Puns have syntactic structures and semantic patterns that can be analyzed and represented in computational models. T-PEG has shown that these computational models or templates can be automatically extracted from training examples of human puns with the use of available linguistic resources. The word relationships extracted are

[1] Elephant characters in children's stories are usually portrayed to have good memories, with the common phrase "An elephant never forgets."

synonyms, is-a-word, sounds-like, and semantic relationships. User feedback further showed that the resulting puns are of a standard comparable to their source puns.

A template is learned for each new joke fed to the T-PEG system. However, the quantity of the learned templates does not necessarily improve the quality of the generated puns. Future work for T-PEG involves exploring template refinement or merging, where a newly learned template may update previously learned templates to improve their quality.

T-PEG is also heavily reliant on the presence of word relationships from linguistic resources. This limitation can be addressed by adding some form of manual intervention to address the missing word relationships caused by limitations of the external resources, thereby increasing the number of usable templates. A different tagger that returns multiple tags may also be explored to consider all possible tags in both the learning and the generation phases.

The manual process employed by the proponents in identifying which of the generated puns are indeed humorous is very time-consuming and subjective. Automatic humor recognition, similar to the works of Mihalcea and Pulman (2007), may be considered for future work.

The template-learning algorithm of T-PEG can be applied in other NLP systems where the extraction of word relationships can be explored further as a means of teaching vocabulary and related concepts to young readers.

References

Kim Binsted. 1996. *Machine Humour: An Implemented Model of Puns*. PhD Thesis, University of Edinburgh, Scotland.

Kim Binsted, Anton Nijholt, Oliviero Stock, and Carlo Strapparava. 2006. Computational Humor. *IEEE Intelligent Systems*, 21(2):59-69.

Kim Binsted and Graeme Ritchie. 1997. Computational Rules for Punning Riddles. *HUMOR, the International Journal of Humor Research*, 10(1):25-76.

Kim Binsted, Helen Pain, and Graeme Ritchie. 1997. Children's Evaluation of Computer-Generated Puns. *Pragmatics and Cognition*, 5(2):309-358.

Iyas Cicekli, and H. Atay Güvenir. 2003. Learning Translation Templates from Bilingual Translation Examples. *Recent Advances in Example-Based Machine Translation*, pp. 255-286, Kluwer Publishers.

Susan Fitt 2002, Unisyn Lexicon Release. Available: http://www.cstr.ed.ac.uk/projects/unisyn/.

Kathleen Go, Manimin Morga, Vince Andrew Nunez, Francis Veto, and Ethel Ong. 2007. Extracting and Using Translation Templates in an Example-Based Machine Translation System. *Journal of Research in Science, Computing, and Engineering*, 4(3):17-29.

Bryan Anthony Hong and Ethel Ong. 2008. Generating Punning Riddles from Examples. *Proceedings of the Second International Symposium on Universal Communication*, 347-352, Osaka, Japan.

Hugo Liu, and Push Singh, 2004. ConceptNet — A Practical Commonsense Reasoning Tool-Kit. *BT Technology Journal*, 22(4):211-226, Springer Netherlands.

Ruli Manurung, Graeme Ritchie, Helen Pain, and Annalu Waller. 2008. Adding Phonetic Similarity Data to a Lexical Database. *Applied Artificial Intelligence*, Kluwer Academic Publishers, Netherlands.

Rada Mihalcea and Stephen Pulman. 2007. Characterizing Humour: An Exploration of Features in Humorous Texts. *Computational Linguistics and Intelligent Text Processing,* Lecture Notes in Computer Science, Vol. 4394, 337-347, Springer Berlin.

Ion Muslea. 1999. Extraction Patterns for Information Extraction Tasks: A Survey. *Proceedings AAAI-99 Workshop on Machine Learning for Information Extraction*, American Association for Artificial Intelligence.

Daniel Naber. 2003. A Rule-Based Style and Grammar Checker.

Vince Andrew Nunez, Bryan Anthony Hong, and Ethel Ong. 2008. Automatically Extracting Templates from Examples for NLP Tasks. *Proceedings of the 22nd Pacific Asia Conference on Language, Information and Computation*, 452-459, Cebu, Philippines.

Graeme Ritchie. 2005. Computational Mechanisms for Pun Generation. *Proceedings of the 10th European Natural Language Generation Workshop*, 125-132. Aberdeen.

Graeme Ritchie, Ruli Manurung, Helen Pain, Annalu Waller, and D. O'Mara. 2006. The STANDUP Interactive Riddle Builder. *IEEE Intelligent Systems* 21(2):67-69.

K. Webb, *The Crack-a-Joke Book*, Puffin Books, London, England, 1978.

WordNet: A Lexical Database for the English Language. Princeton University, New Jersey, 2006.