Algoritmo para generar frases homófonas de calambures desde un enfoque fonético

Roberto Villarejo-Martínez, Noé Alejandro Castro-Sánchez

Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México

{rvillarejo, ncastro}@cenidet.edu.mx

Resumen. Los juegos de palabras son un tipo de humor verbal ampliamente utilizado en todo el mundo. Comúnmente se valen de la homofonía o de frases muy similares en sonido, además de la ambigüedad en los significados de las palabras. En el área de Procesamiento de Lenguaje Natural normalmente esto representa una deficiencia. Sin embargo, en el humor computacional puede verse como una ventaja que puede ser aprovechada en la generación e identificación de estos textos. En este artículo se describen los fenómenos lingüísticos más utilizados en los juegos de palabras. Además se propone un algoritmo para la generación de frases homófonas a partir de una frase, utilizando un enfoque fonético y un lexicón no humorístico ni entrenado para tal propósito. El algoritmo fue capaz de generar 22 de 45 calambures extraídos principalmente de adivinanzas populares.

Palabras clave: calambur, humor computacional, homofonía.

Algorithm for Generate Homophone Phrases of Puns from a Phonetic Approach

Abstract. Wordplays are a type of verbal humor widely used around the world. Usually, they use homophony or phrases very similar in sound, furthermore ambiguity in word senses. In Natural Processing Language area this usually represents a failure. However, in Computational Humor, it can be seen as an advantage that can be taken as advantage in generation and identification of this kind of texts. In this article the most used linguistic phenomena are described. Additionally an algorithm for homophones phrases generation based on a phrase is described, using a phonetic approach and a non-humorous lexicon, neither trained. This algorithm was able to generate 22 of 45 puns taken, mainly, from popular guessing games.

Keywords: pun, computational humor, homophony.

1. Introducción

Los chistes, normalmente, tienen una preparación y un remate. La preparación crea ciertas expectativas y el remate las rompe, conduciendo así a diferentes interpretaciones de la preparación [4]. Cuando se trata de chistes que involucran juegos de palabras sucede lo mismo. Los chistes de juegos de palabras, o chistes que involucran juegos verbales, son una clase de chistes que dependen de palabras que son similares en sonido, pero se usan con diferentes significados. La diferencia entre los significados crea un conflicto que rompe la expectativa y es humorística. El juego de palabras puede ser creado entre: dos palabras con la misma pronunciación y ortografía, dos palabras con diferente ortografía pero misma pronunciación y, con dos palabras con diferente ortografía y pronunciación similar [5].

Como ejemplo tenemos el chiste "le pedí un café y me dijo 'sólo queda té', fue hermoso". La primera frase "le pedí un café" establece cierta expectativa; la segunda frase "me dijo 'sólo queda té'" es una respuesta aceptable y encaja con la primera; sin embargo, la última frase "fue hermoso" fuerza a interpretar la segunda de manera diferente, se sitúa ahora en un contexto romántico. La segunda frase, entonces, debe leerse como "sólo quédate" que es homófona a la original. Se ha logrado pues romper la expectativa y se produce así un efecto cómico o divertido.

2. Estado del arte

Investigadores de los tiempos antiguos de Aristóteles y Platón hasta nuestros días se han esforzado por descubrir y definir el origen del humor. Hay casi tantas definiciones del humor como teorías [1]. En el área computacional se han realizado apenas algunos trabajos y pocos han sido en español.

Para el idioma inglés una de las investigaciones más destacadas es el de [5] quienes crearon un programa que reconoce chistes *knock, knock* utilizando técnicas de n-gramas, un generador/reconocedor de juegos de palabras y un reconocedor/generador de remate de chistes. La comicidad de este tipo de chistes se basa en el juego verbal, sobre todo en el sonido de las palabras, principalmente en la paronimia, homonimia y homofonía.

En cuanto a juegos de palabras, se reporta que el programa fue capaz de encontrar el juego de palabras (contenido en la última línea de un tipo específico de chiste *knock, knock*) en 85 de 122 chistes que pudo haber reconocido potencialmente y, en la mayoría de los casos, el juego de palabras encontrado coincidió con el esperado. En cuanto al reconocimiento de chistes, el programa reconoció exitosamente 62 de los no chistes usando n-gramas. De los 130 chistes nuevos (desconocidos para el sistema) el programa no pudo reconocer ocho.

También realizaron algunos experimentos [6] en los que el objetivo era encontrar las palabras que producen el juego de palabras en chistes. Los autores parten de la hipótesis de que todo chiste de juego de palabras contiene un par de palabras *source* y *target*. Indican que *source* es una frase en el chiste que entra en conflicto con la intepretación inicial del perceptor; *target* es una frase que suena similar a *source* y, la similitud entre

ambas, además del conflicto de la interpretación inicial, conduce al descubrimiento de una segunda interpretación del texto. El experimento consistió en buscar la palabra *source* de final a principio del chiste y, una vez que se encontraba, se generaron varias palabras *target*. En resumen, la investigación toma ventaja de la estructura del chiste en relación a su familiaridad y valores de frecuencia KF para aplicar los reconocedores de recursos en la manera más eficiente.

JAPE [2] es un software capaz de generar *punning riddles* a partir de un léxico no humorístico. En ese trabajo se desarrolló un modelo formal de estos acertijos el cual se compone de cuatro partes: esquemas, que especifican las relaciones entre las unidades léxicas utilizadas para construir un chiste; el generador SAD (*Small Adequate Description*) que contiene descripciones cortas del mundo o de unidades léxicas construidas; las plantillas, que convierten unidades léxicas y sus descripciones en un formato de pregunta-respuesta y; recursos léxicos, que proveen toda la información léxica requerida para que las otras partes funcionen.

Los *punning riddles* son un subtipo de humor basado en juegos de palabras con un formato de acertijo. Cuentan con ciertos mecanismos de construcción y estructuras regulares. Un ejemplo de *punning riddle* es "What kind of tree is nauseated? – A sick-amore". En el idioma inglés existe cierta homofonía entre *sick* y *sycamore*. La primera palabra se usa para decir que una persona sufre de nauseas. La segunda se refiere a un árbol. Se puede decir que *sick* se obtiene de aplicar aféresis a *sycamore.* La similitud fonética entre ambas y el rompimiento de la expectativa creada por la pregunta inicial producen la comicidad del acertijo.

3. Fenómenos lingüísticos involucrados

Algunos juegos de palabras están basados en la homofonía entre dos palabras diferentes en significado y/o diferente ortografía, o bien, entre dos frases que cumplan estas condiciones. De esta manera, los juegos de palabras se prestan a diferentes interpretaciones e incluso a construcciones léxicas diferentes dependiendo de cómo suena la frase. En la mayoría de los casos esta homofonía es producida por los fenómenos lingüísticos que se describen a continuación.

Uno de los fenómenos más comunes en los juegos de palabras (o juegos de frases) es el calambur, el cual consiste en realizar un agrupamiento de sílabas diferente al original para formar una nueva frase. Por ejemplo, la oración "yo lo quito" puede reestructurarse para formar la oración "yo loquito" que adquiere un significado totalmente diferente. La comicidad de los juegos de palabras recae en exponer la frase que inicialmente se encontraba oculta debido a la homofonía y al significado de dicha frase inicial.

Otros de los fenómenos más frecuentes son la sinalefa, la sinéresis y la contracción. Éstos consisten en pronunciar como uno solo dos sonidos contiguos dentro de una palabra (sinéresis) o, del final de una y el principio de otra (sinalefa). La contracción se presenta cuando dos sonidos contiguos iguales se pronuncian como uno solo. Por ejemplo /t//e//ch//o/ en lugar de "te echo" que, en el plano acústico, son difíciles de

desambiguar. Incluso algunos sistemas de reconocimiento de voz tienen problemas para hacerlo.

En los juegos de palabras también deben considerarse los metaplasmos como aféresis y apócope. Y es que no siempre la frase latente coincide completamente con la frase original, ya sea en su forma escrita o a nivel fonético. La aféresis consiste en la desaparición de uno o más fonemas al principio de una palabra mientras que la apócope se realiza al final de una palabra. Como ejemplo tenemos un chiste en el que un panqué de chocolate hincado frente a una fresa le dice "mi corazón chocolate por ti". El juego de palabras está en usar "chocolate" por "late" siendo la primera el sabor del panqué. Este juego de palabras es lo que hace divertida a la imagen.

Por tanto, para comprender los juegos de palabras hay que realizar una construcción léxica diferente. La nueva frase debe ser homófona a la inicial o muy similar. También puede ser que la representación fonética de esta nueva frase coincida solo en parte a la de la original. Esta última condición correspondería al apócope y aféresis. En el ejemplo de la oración "El lado oscuro" generaríamos frases como "Helado oscuro" o "Helados curo".

4. Algoritmo para generar frases homófonas

En este trabajo se propone el siguiente algoritmo para generar frases homófonas a partir de una frase de entrada utilizando un lexicón no humorístico. Este proceso consta de tres fases: generación de la cadena fonética, búsqueda fonética a partir de la cadena fonética utilizando el lexicón y, construcción de nuevas frases.

Se debe considerar que las frases resultantes no son validadas a nivel sintáctico después de generarlas y que, entre mayor número de palabras posea el lexicón más palabras serán encontradas, mismas que producirán un número mayor de frases. En la figura 1 se muestra el proceso general para generar frases homófonas a partir de una oración de entrada.

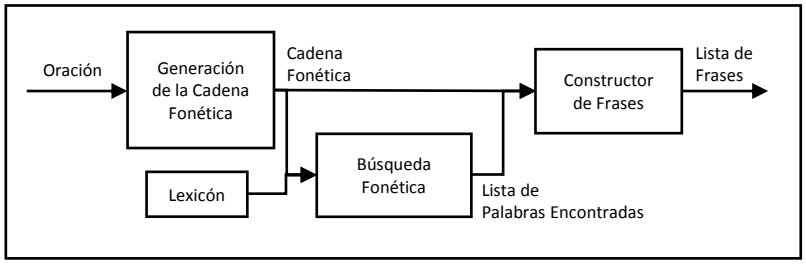

Fig. 1. Proceso general de generación de frases homófonas.

4.1. Generación de la cadena fonética

En esta fase se genera una cadena fonética que representa el sonido de una frase mediante símbolos de un alfabeto fonético. En este caso la oración es una lista ordenada

de palabras. A su vez, una palabra posee los siguientes atributos: forma de palabra, forma fonética, etiqueta gramatical, forma semántica.

Las letras de una oración son símbolos gráficos que corresponden a sonidos del plano acústico. Para transcribir fonéticamente una palabra se aplica una serie de reglas fonológicas que las cambian una o más letras por un símbolo de un alfabeto fonético, éste puede ser tan específico como se desee. En este caso se utiliza una modificación propia de las reglas fonológicas de SAMPA [7]. El motivo de esta modificación es generalizar algunos sonidos (fonemas y alófonos) para producir ambigüedad fonética, y a su vez provocar homofonía entre una palabra y otra o, entre una palabra y una frase o, viceversa.

Una regla fonológica está compuesta de tres elementos: letra(s), reemplazo y contexto. El reemplazo es un símbolo del alfabeto fonético que corresponde al sonido de una o más letras (primer elemento). El contexto es la condición que debe cumplir el primer elemento para ser reemplazado por el segundo.

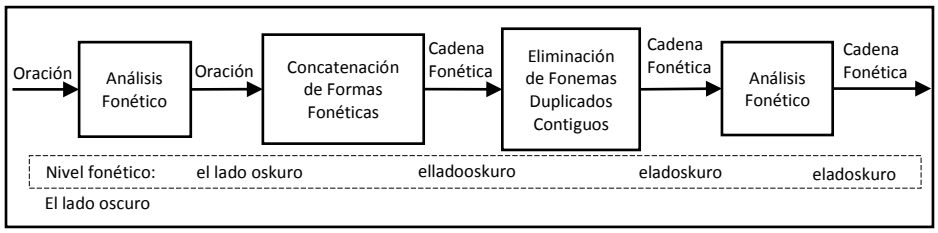

Fig. 2. Generación de la cadena fonética.

Tabla 1. Reglas fonológicas de SAMPA modificadas.

Categorías		
W=ei (vocales débiles)		
F=aou (vocales fuertes)		
B=pb (bilabiales)		
Reglas fonológicas[1]		
rr/R/_	q/k/_	z/s/_
r/R/^_	c/s/_W	h//_
ch/X/_	c/k/_	v/b/_
x/X/_	g/j/_W	y/i/_
qu/k/_W	gu_g/_W	ñ/J/_
qu/ku/_F	gü/hu/_	n/m/_B
qü/ku/_	ll/i/_	n/m/_f

En la figura 2 se muestran los sub-procesos que conforman a esta fase y se describen enseguida. Esta sucesión de pasos ayuda a simular fenómenos fonéticos como la contracción, sinéresis y calambur puesto que con la cadena fonética generada no se

[1] El símbolo "^" indica que el primer elemento (la letra r) debe encontrarse al inicio de una palabra.

distinguen los límites de las palabras o *tokens*. Por lo tanto, a partir de ésta se pueden realizar construcciones léxicas diferentes a la oración original.

- Como primer paso se realiza un análisis fonético de la oración de entrada, el cual consiste en enriquecer con su respectiva forma fonética a cada una de las palabras de la oración. Se utilizan las reglas fonológicas de la tabla 1.
- En el segundo paso se genera la cadena fonética concatenando (sin espacios en blanco) cada una de las formas fonéticas de las palabras, respetando su orden de aparición.
- Como tercer paso se eliminan los fonemas duplicados contiguos. Esto con el propósito de simular los fenómenos de contracción, sinéresis y sinalefa. Además de que permite que los sonidos (o fonemas) que se encontraban separados puedan re-analizarse en el paso siguiente.
- Como último paso se realiza un análisis fonético sobre la cadena fonética ya generada. De esta manera, se recodifican los sonidos (o fonemas) que antes del segundo o tercer paso se encontraban separados.

En la frase de ejemplo “el lado oscuro” se generara la cadena fonética “eladoskuro”.

4.2. Búsqueda fonética

Algoritmo 1. Búsqueda fonética.

```
Entrada: cadena_fonética: es una cadena de caracteres previamente generada
Salida: palabras_encontradas: lista de palabras encontradas
Descripción: Esta función guarda en resultados las palabras del lexicón cuya forma
fonética se encuentre contenida en cadena_fonética. A cada elemento de resultados
se le llama palabra encontrada; se compone de una palabra y la posición de inicio
y fin de su cadena fonética respecto a cadena_fonética.
INICIO
   MIENTRAS (lexicón tenga palabras) HACER
      palabra = actual palabra del lexicón
      forma_fonética = forma fonética de palabra
      SI (cadena_fonética contiene a forma_fonética) ENTONCES
         iniciaEn = posición de forma fonética en cadena_fonética
         long = tamaño de cadena_fonética
         terminaEn = iniciaEn + long
         crea palabra_encontrada con palabra, iniciaEn, terminaEn
         añade palabra_encontrada a palabras_encontradas
      FIN
   FIN
   retorna resultados
FIN
```

La cadena fonética representa el sonido de una frase por medio de símbolos de un alfabeto fonético. En una oración escrita los elementos que la conforman (*tokens*) se encuentran bien definidos y son fáciles de identificar porque se encuentran separados por espacios. En el plano acústico no siempre sucede así, debido a que las personas tendemos a formar vínculos entre el final de una palabra y el inicio de otra. Los límites entre éstas no suelen estar bien definidos. Esto propicia la ambigüedad fonética y esta, a su vez, provoca que estos sonidos puedan interpretarse de una manera diferente a la deseada, es decir, crear nuevas oraciones.

A continuación se genera una lista de palabras encontradas[2] a partir de la cadena fonética obtenida de la fase anterior. El objetivo es seleccionar aquellas palabras del lexicón cuya forma fonética se encuentra contenida en la cadena fonética. Por ejemplo, una cadena fonética "deskritos" puede interpretarse como "de escritos" o "descritos".

4.3. Construcción de nuevas frases

Cuando una persona escucha, sin previa contextualización, los sonidos */eladoskuro/* podría entender frases como "el lado oscuro" o "helados curo" o "helado oscuro". La idea es generar nuevas frases a partir de las palabras encontradas en la fase anterior y la cadena fonética de la primera fase.

Algoritmo 2. Construcción de nuevas frases.

```
Entrada: palabras_encontradas: lista de palabras encontradas ordenadas
Salida: frases: lista de frases generadas
Descripción: genera una lista de frases a partir de una lista de palabras
encontradas y una cadena fonética (misma usada para generar la lista de
palabras encontradas).
INICIO
   MIENTRAS (palabras_encontradas tenga elementos)  HACER
      palabra_encontrada_1 = actual palabra de la lista
      crea lista_palabras
      añade palabra_encontrada a lista_palabras
      cadena_fonética_1 = remoción_fonética(cadena_fonética,
palabra_encontrada_1)
      MIENTRAS (palabras_encontradas tenga elementos) HACER
         palabra_encontrada_2 = remoción_onética(cadena_fonética_1,
palabra_encontrada_2)
         SI (cadena_fonética_1 == cadena_fonética_2) ENTONCES
            añade palabra_encontrada_2 a lista_palabras
            cadena_fonética_1 = cadena_fonética_2
         FIN
      FIN
      ordena lista_palabras por índice inicial, luego por índice final y por
índice inicial
      crea frase con lista_palabras, cadena_fonética
      añade frase a frases
   FIN
   RETORNA frases
FIN
```

El proceso consiste básicamente en generar combinaciones de las palabras encontradas, respetando su orden de aparición, verificando que éstas puedan coexistir fonéticamente. Para lograrlo se utiliza la función *remoción_fonética*. En el ejemplo de la cadena fonética "eladoskuro" las palabras encontradas "helado" y "lado" no podrían coexistir porque sus formas fonéticas se traslapan. Esto no significa que una de las dos no pueda usarse en ningún caso sino que deben construirse oraciones diferentes: con la

[2] Una palabra encontrada se compone de una palabra y posiciones inicio y fin de su forma fonética respecto a una cadena fonética. Por ejemplo la cadena fonética "*eladoskuro"* contiene la cadena "*elado*" que es la forma fonética de "helado". Sus posiciones de inicio y sin son 0 y 5 respectivamente.

palabra "helado" se construye la frase "helado oscuro", con la palabra "lado" se construye la frase "el lado oscuro".

Las frases generadas[3] en esta fase no son validadas a nivel sintáctico, por lo tanto podrían carecer de sentido. La lista de palabras encontradas que se introducen al algoritmo deben ser previamente ordenadas por índice inicial, luego por índice final y nuevamente por índice inicial.

Al final se tiene una lista de frases nuevas de las que se calcula su cobertura fonética. Esta métrica ayuda a seleccionar, en un análisis posterior, la mejor frase candidata para analizarla o sustituirla por la original, dependiendo de las necesidades. Por ejemplo, si la longitud de una cadena fonética de una frase es 9 y la cadena fonética original es 10 entonces se tiene una cobertura fonética de 0.9.

A continuación se detalla la función *remoción_fonética* que se utiliza en el algoritmo anterior. Ésta función se asegura de remover la cadena fonética la forma fonética de una palabra a excepción de su primer y último carácter. De esta manera se permite que estos fonemas puedan ser reutilizados por otras palabras y así simular fenómenos como: contracción, sinéresis.

Algoritmo 3. Remoción fonética.

```
Entrada: palabra_encontrada, cadena_fonética
Salida: resultado: es una cadena de caracteres
Descripción: retorna cadena_fonética si la misma no contiene a la forma
fonética de palabra_encontrada. En caso contrario reemplaza en cadena_fonética
la forma fonética de palabra_encontrada por el primero y último carácter de la
forma fonética separados por un espacio.
INICIO
   forma_fonética = forma fonética de palabra_encontrada
   reemplazo = forma_fonética
   long = longitud de forma_fonética
   SI (cadena_fonética contiene a forma_fonética) ENTONCES
      SI (long >= 2) ENTONCES
         primero = primer caracter de forma_fonética
         último = último caracter de forma_fonética
         reemplazo = concatenar(primero, " ", último)
      FIN
      resultado = sustitución de forma_fonética por reemplazo en cadena_fonética
      RETORNA resultado
   EN CASO CONTRARIO
      RETORNA cadena_fonética
   FIN
FIN
```

Esta función verifica si una palabra puede añadirse o no a una oración dependiendo de la cadena fonética que se introduce. De ser así se modifica la cadena fonética a manera de preparación para verificar la siguiente palabra. A continuación se muestra un ejemplo de utilización de la función *remoción_fonética*.

[3] Una frase se compone de los siguientes atributos: una oración, que es la nueva construcción léxica; una cadena fonética propia, que corresponde a la oración; una cadena fonética de origen, que es la cadena fonética original y; una cobertura fonética, que es una proporción entre la longitud de la cadena fonética propia y la cadena fonética original.

Tabla 2. Ejemplo de uso de la función *remoción_fonética.*

Entradas			**Salida**
Cadena fonética	**Forma fonética**	**Reemplazo**	
eladoskuro	*elado* (helado)	e_o	e_oskuro
e_oskuro	*lado* (lado)		e_oskuro
e_oskuro	*oskuro* (oscuro)	o_o	e_o_o

A diferencia de los casos primero y tercero (renglones), note que en el segundo la salida de la función es igual a la cadena fonética de entrada. Esto significa que la palabra a la que corresponde esta forma fonética no puede coexistir en la oración y se descarta.

5. Pruebas y resultados

Para probar el algoritmo propuesto se utilizaron calambures de adivinanzas populares. Usualmente, son los calambures los que producen la comicidad en las adivinanzas. Por ejemplo, la adivinanza "oro no es, plata no es" contiene un calambur que, a su vez, es la respuesta de la misma. El calambur "plata no es" debe ser reinterpretada como "plátano es" para así responder a la adivinanza.

Tabla 3. Algunas pruebas del algoritmo en calambures de adivinanzas.

Calambur	**Resultado esperado**	**Grupo seleccionado**
Espera	Es pera	eh es pera he es pera **es pera** espera es pera ah es pera ha
Escapa	Es capa	escapa **es capa** es cap ah es cap ha
Lana baja	La navaja	lana va ja la ana va ja la han ah va ja la han ha va ja **la** han **navaja**

Como resultados se esperan obtener varias frases de las que se mide la cobertura fonética. A continuación se muestran algunas de las pruebas realizadas. Las frases que aparecen en la tercera columna fueron elegidas de entre todas las frases candidatas por tener el 100% de cobertura fonética.

En la implementación del algoritmo se utilizó como lexicón el diccionario de Freeling [3] el cual está formado por más de 556'000 entradas o formas de palabra

diferentes. Al analizar las pruebas se observa que cuanto más extenso es el lexicón mayor será el número de palabras encontradas con las que se forman las frases. El tamaño del lexicón también tiene efecto en las oraciones generadas: se producen frases "segmentadas". Por ejemplo, la frase "el la ado hoz cu uro" es, en efecto, homófona a la cadena fonética "eladoskuro". Sin embargo no es sintácticamente válida y, por lo tanto, carece de sentido.

Para medir la cobertura se verificó que el resultado esperado se encontrara en el grupo seleccionado mediante el criterio de cobertura fonética. En las pruebas, la frase esperada se encontró en el grupo seleccionado en 22 de 45 calambures. Es decir, se obtuvo un cobertura del 48.88%.

6. Conclusiones

Con el algoritmo presentado en este artículo es posible producir frases homófonas a una frase dada. Esto permite reinterpretar una oración a partir de su representación acústica por medio de un alfabeto fonética. Las frases generadas permiten reanalizar la frase en cuestión para comprender los significados alternos que se manejan sobre todo en textos humorísticos.

Como trabajo futuro se pretende desarrollar un módulo sintáctico como auxiliar en el proceso de construcción de frases. Se contempla la posibilidad de implementar un modelo de n-gramas para calcular la probabilidad de usar una palabra después de otra y así descartar algunas combinaciones indeseables. Otra posible solución es utilizar reglas gramaticales que ayuden a determinar si una palabra pudiera usarse o no después de otra.

En muchos de los casos la incongruencia a nivel semántico es la que provoca que se rompan expectativas en los textos humorísticos. Por lo tanto, no se planea implementar ningún tipo de validación en ese nivel. Este trabajo está enfocado al humor computacional y por tanto se desea mantener esta ambigüedad.

Los calambures se encuentran usualmente en adivinanzas, chistes u otro tipo de textos humorísticos. Por esa razón se considera que es importante poseer la capacidad computacional para resolverlos, sobre todo si se desea alcanzar una comprensión computacional del humor.

En este artículo no se reportan resultados de precisión porque se espera medirla una vez que se haya desarrollado e implementado un módulo sintáctico que complemente el proceso de construcción de frases.

Referencias

1. Latta, R. L.: The Basic Humor Process: A Cognitive-Shift Theory and the Case against Incongruity. Berlin-Nueva York: Mouton de Gruyter (1999)
2. Binsted, K.: Machine humour: An implemented model of puns. The University of Edinburgh (1996)

3. Carreras, X., Chao, I., Padró, L., Padró, M.: Freeling: An Open-Source Suite of Language Analyzers. Proceedings of the 4th Language Resources and Evaluation Conference (LREC 2004), 4, 239–242 (2004)
4. Ritchie, G.: Developing the Incongruity-Resolution Theory (1976)
5. Taylor, J. M., Mazlack, L. J.: Humorous Wordplay Recognition. In International Conference on Systems, Man and Cybernetics Proceedings (Vol. 4, pp. 3305–3311). The Hague, The Netherlands: IEEE (2004)
6. Taylor, J. M., Mazlack, L. J.: Reverse engineering Humor, 45221 (2008)
7. Wells, J.C.: SAMPA computer readable phonetic alphabet. In Gibbon, D., Moore, R. and Winski, R. (eds.), Handbook of Standards and Resources for Spoken Language Systems.
8. Berlin and New York: Mouton de Gruyter, Part IV, section B (1997)