Homophonic Pun Generation with Lexically Constrained Rewriting

Zhiwei Yu[*1,2], **Hongyu Zang**[*1] **and Xiaojun Wan**[1]
[1]Wangxuan Institute of Computer Technology, Peking University
[1]The MOE Key Laboratory of Computational Linguistics, Peking University
[2]Microsoft Research Asia
yuzw, zanghy, wanxiaojun@pku.edu.cn

Abstract

Punning is a creative way to make conversation enjoyable and literary writing elegant. In this paper, we focus on the task of generating a pun sentence given a pair of homophones. We first find the constraint words supporting the semantic incongruity for a sentence. Then we rewrite the sentence with explicit positive and negative constraints. Our model achieves the state-of-the-art results in both automatic and human evaluations. We further make an error analysis and discuss the challenges for the computational pun models.

1 Introduction

In this work, we mainly study the homophonic puns where two meanings relying on the same (or similar) sounding signs. As Figure 1 shows, the pun exploits the sound similarity between "tuna" and "tune". The word ("tuna") appearing in the sentence that triggers humor is a pun word, while its homophonic word ("tune") is the alternative word. The semantics of the homophones are expressed by two words independently ("whistling", "fisherman"). On the surface, there is one interpretation: "the fisherman had no tuna". Implied by "whistling" and the pronunciation of "tuna", there is another interpretation:"the whistling fisherman sang out of tune".

Early models for pun generation are mainly relying on templates (Lessard and Levison, 1992; Binsted and Ritchie, 1994, 1997; Lessard, 1992; McKay, 2002; Ritchie et al., 2007; Hong and Ong, 2009). To improve the diversity of generated puns, Yu et al. (2018) propose a neural approach to generate puns conditioning on two meanings of the target word. Based on it, Luo et al. (2019) introduce a word sense classifier as the discriminator to generate homographic puns by adversarial generative network. However, neural generative models usually mimic the norm and the generated sentences are lack of novelty. To make the generated puns more creative, He et al. (2019) first sample a sentence containing the alternative word from the corpus. They then replace the alternative word with the pun word and insert a topic word. This retrieve-and-edit approach is brilliant. However, the part-of-speech (POS) tags of the pun words and its alternative words are different in 46.08% puns in the gold pun dataset (Miller et al., 2017). Directly replacing the alternative word with the pun word usually leads to grammar errors in the generated sentence. As there are tons of sentences containing the alternative word, it is also necessary to rank the sentences purposefully.

Figure 1: An example of a homophonic pun.

To address the issues above, we propose a constraint selection algorithm to extract the candidate sentence containing the alternative word and its corresponding constraints. Then our lexically constrained rewriting model (LCR) generates puns by rewriting the normal sentences with constraints. In this way, the different semantic meanings of two homophones are expressed naturally in a generated sentence. Both automatic and human evaluation results demonstrate the efficacy of our model[1].

2 Our Method

The framework of our model is shown in Figure 2. Given a pair of homophones, we retrieve sentences containing the alternative word ("tune") and pun word ("tuna"). Next, we extract the most suitable

[*] The two authors contributed equally to this paper. Contribution was done at Peking University.

[1]https://github.com/ArleneYuZhiwei/LCR.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2870–2876,
November 16–20, 2020.

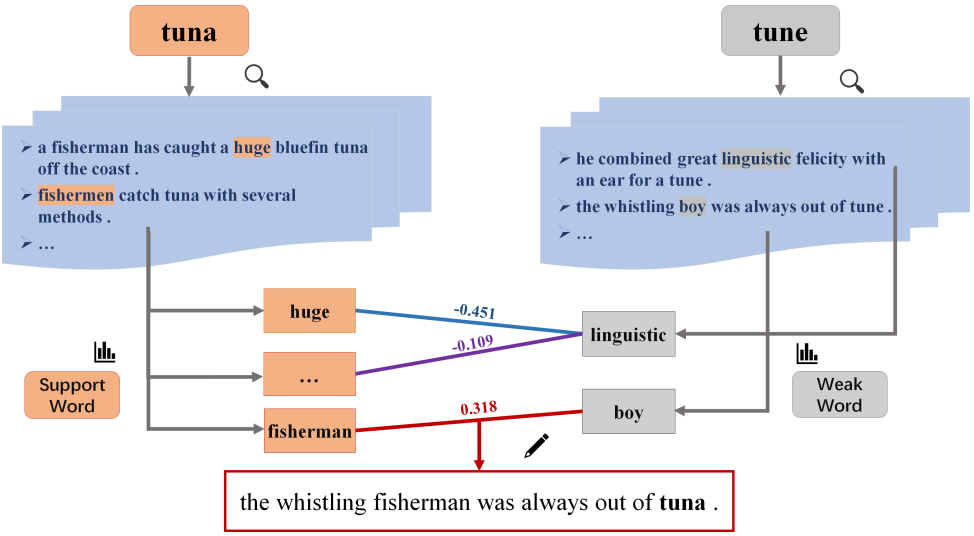

Figure 2: Overview of our pun generation process.

weak word ("boy") and the corresponding support word ("fisherman") to form constraints. Positive constraints contain the pun word and its support word. Negative constraints contain the alternative word and its weak word. Finally, we rewrite the candidate sentence with lexical constraints to generate a homophonic pun.

2.1 Constraints Extraction

Incongruity is a leading theory in computational humor. We achieve it by rewriting the sentences with lexical constraints. For each pair of homophones, we obtain a set of candidate sentences C. For the i-th candidate sentence, we extract words weakly related to the alternative word to compose the weak word vocabulary W_i. The words in the whole corpus which have the same POS tag with each word w in W_i and are strongly related to the pun word compose the support word vocabulary $S_{i,w}$. We use Point-wise Mutual Information (PMI) to evaluate the relatedness between two words (e.g. x, y) (Church and Hanks, 1990) :

$$PMI(x,y) = \log_2 \frac{p(x,y)}{p(x) \cdot p(y)}. \quad (1)$$

To make the pun words more plausible, we hope to replace the weak word with a support word. Keep $||W_i|| = nw$ words with lowest PMI scores in W_i and $||S_{i,w}|| = ns$ words with highest PMI scores in $S_{i,w}$. For each sentence, there are $nw * ns$ pairs of possible constraints. To keep the edited sentences grammatical, we need to select one pair which results in the most reasonable modifications. As Algorithm 1 shows, for each homophone pair, we go through the corpus and calculate scores for their candidate sentences. We train a Continuous Bag of Words model (CBOW) to obtain the word embeddings (Mikolov et al., 2013). Inspired by Mao et al. (2018), we use OUT-IN vectors to measure the similarity between a word and its given

Algorithm 1 Constraint Selection Algorithm.

Require: C: a set of candidate sentences containing the alternative word a
Require: M: a trained CBOW model. v_a^i denotes the input vector of word a. v_a^o denotes the output vector of word a. $v_{context}^i$ denotes the average input vectors of the words in the $context$.
Require: W: a list of the weak word vocabulary sets.
Require: S : a list of the support word vocabulary sets.
Require: p : corresponding pun word
$Sim(x,y)$ calculates the cosine similarity between two vectors x, y.
$R = \emptyset$
For any sentence $C_i \in C$:
$context$ = the set consisting of the words in C_i excluding a.
  For any weak word w in W_i:
    For any support word s in $S_{i,n}$:
      $q \leftarrow$ the set consists of the words in $context$ excluding w and including p.
      $Score_{i,w,s} \leftarrow Sim(v_w^i, v_s^i) + Sim(v_s^o, v_q^i) - Sim(v_p^i, v_s^i)$
      $R \leftarrow R \cup Score_{i,w,s}$
Return C_i, w, s where $Score_{i,w,s} = maxR$

contexts, and use IN-IN vectors to measure the similarity between two words.[2] The support word should play the same role as the weak word and fits the contexts well. For a specific support word, its contexts are likely to contain all the words in the candidate sentence except for the alternative word and its weak word. So we remove these two words and add the pun word to form the potential contexts q. We calculate the similarity between the support word and its potential contexts q, to evaluate the fitness of the support word. In practice, $S_{i,w}$ always contains inflections of the pun word, which can make the generated sentences redundant and less likely to be a pun. So the similarity between the pun word and its support word is negatively correlated to the final score. In our case, we do not consider verbs as weak words.[3]

2.2 Lexically Constrained Rewriting

With the extracted constraints, we further generate homophonic puns by rewriting the candidate sentence. Following Hu et al. (2019a), we train a generator in an end-to-end way. Given a source sentence x, the generator aims to rewrite it as $\tilde{y}$. In the end-to-end model, $\tilde{y}$ maximizes the conditional probability given by a model θ and an input sequence x:

$$\tilde{y} = argmax_y p_\theta(y|x), y \in Y \quad (2)$$

where Y is the space of possible outputs. Traditional beam search can not guarantee the outputs conformed to the constraints.

Researchers propose algorithms to place constraints in natural and meaningful ways (Hokamp and Liu, 2017; Post and Vilar, 2018). However, previous works ignore the situation that some constraints may share a prefix. To avoid repeating, the constraints that have not been generated are organized into a trie. For positive constraints, there is a counter to indicate that how many times the constraint must be generated. When a constraint is generated, its counter is decremented. For negative constraints, the trie does not need any counters. At each time step, the generation of an active phrase is blocked by setting the costs of all word IDs marked in the current node to infinity. Hypotheses are ranked by sentence number, number of unmet constraints, and sequence scores. We keep the top-k hypotheses. In this way, both positive constraints and negative constraints can be satisfied without repeating. We call our model Lexically Constrained Rewriting Model (**LCR**).

[2]IN vectors are input vectors of a trained CBOW model. OUT vectors are output vectors of a trained CBOW model.

[3]Replacing verbs can result in the transformations of other words, which changes the candidate sentence to a great extent.

3 Experiments

3.1 Data Set

We use PARABANK (Hu et al., 2019b), a large-scale English paraphrase dataset to train the rewriting model. Following previous work (He et al., 2019), we use BookCorpus (Zhu et al., 2015) as retrieval corpus. And we use the homophone pairs in 2017 SemEval task 7 (Doogan et al., 2017) for testing.

3.2 Experimental Setting

We train a CBOW model on BookCorpus with a context window ($width = 5$) to learn 300-dimensional word vectors. In our constraint selection algorithm, we keep $w = 3$ weak words for each sentence and $s = 5$ support words for each weak word. In the decoding phrase, we set the beam size $k = 5$.

3.3 Baseline Models

RE: Retrieve a sentence containing the pun word.

RE+S: Sample one sentence containing the alternative word and replace it with the corresponding pun word.

NJD (Yu et al., 2018): Retrained the model on the BookCorpus. Use two homophones as the inputs for decoding.

SURGEN (He et al., 2019): Given a pair of homophones, their retrieve-and-edit approach generates a homophonic pun.

3.4 Evaluation Metrics

3.4.1 Automatic Evaluation

Following Yu et al. (2018), the diversity is measured by the ratio of distinct unigrams (d.-1%) and bigrams (d.-2%). *w.-num* denotes the number of distinct words in the outputs. As Table 1 shows, the neural model **NJD** tends to generate normal sentences and obtains the lowest scores. It generates sentences using very limited vocabulary. Gold puns sometimes share the similar structure, for example : "old storekeepers never die , they just sale away" and "old school superintendents never die , they just lose their principals", which causes a lower diversity and fewer *w.-num*.

Models	*w.-num*	*d.-1(%)*	*d.-2(%)*
RE	**7,628**	46.76	88.75
RE+S	7,129	42.88	87.07
SURGEN	6,650	45.40	88.18
NJD	4,435	23.86	49.36
LCR	7,062	**56.00**	**93.22**
GOLD	6,432	48.10	87.18

Table 1: Results of automatic evaluation.

Models	*Gram.*	*Flue.*	*Pun.*	*Fun.*	*Overall*
RE	**4.51**	**4.14**	1.44	1.27	2.87
RE+S	4.32	3.85	2.77	1.43	3.11
NJD	3.75	3.41	1.74	1.31	2.49
SURGEN	4.13	3.72	2.71	1.84	2.91
LCR	4.50	4.11	**2.82**	**2.15**	**3.35**
GOLD	4.69	4.52	4.01	3.57	4.29

Table 2: Results of human evaluation.

RE outputs human-written sentences retrieved in the corpus and shows good diversity. By swapping the alternative words with corresponding pun words, **RE+S** obtains equivalent diversity to **RE**. Our model (**LCR**) rewrites retrieved sentences with constraints and generates sentences with creativity and highest diversity.

3.4.2 Human Evaluation

Pun is a creative form of language which is hard to evaluate automatically. For a comprehensive evaluation, we ask annotators to do the human evaluation. We randomly sampled 100 outputs of different models including gold puns and ask native speakers to score the sentences from 1 to 5 on five aspects: (1) *Grammar (Gram.)* Is the sentence grammatically correct? (2) *Fluency (Flue.)* Is the sentence fluent and easy to understand? (3) *Pun (Pun.)* Is this sentence a pun? (4) *Funniness (Fun.)* How funny is the sentence? (5) *Overall* How is the overall quality of this pun? Table 2 shows the results of human evaluation. Our model (**LCR**) outperforms other models in terms of *Pun, Funniness* and *Overall* quality as a pun. As **RE** outputs human-written sentences in BookCorpus, it obtains highest scores in two terms: *Grammar* and *Fluency*. The sentences from **RE** are grammatical and fluent but lack of ambiguity. Thus, it obtains the lowest scores in *Pun* and *Funniness*. With a swap of pun word and alternative word, **RE+S** introduces incongruity to the retrieved sentences and makes them funnier. However, when the usages of pun word and alternative word are different, a swap makes the sentences not readable and decreases the *Grammar* and *Fluency*. **NJD** is a neural language model. Given two homophones, it generates sentences according to the limited input knowledge, which always leads to sentences of low quality. **SURGEN** edits the sentences generated by **RE+S**. Topic words are inserted in the sentences to benefit pun generation. However, due to inconsistent types, the topic words sometimes do not fit in its contexts. The similar issue appears when directly swapping the alternative word with pun word. **LCR** carefully chooses support words which fit the contexts and place them naturally by rewriting the sentences with constraints. The generated sentences are both interesting and grammatical.

3.5 Case Study

Pun/Alternative Word: board-bored	
RE	but now there is no time left... let 's send her on **board**!
RE+S	he brought his other hand to cup my cheek as his eyes went dark and **board** into mine.
NJD	go to the board, i'm not going to be able to do so.
SURGEN	the gulls went on, "but i am pretty sure you won't be **board** for long."
LCR	your bedding is **board**.
GOLD	do hotel managers get **board** with their jobs ?
Pun/Alternative Word: rune-ruin	
RE	i'd once had a necklace like hers, except mine had been shaped like a spider **rune**.
RE+S	when that time comes, i will guide your hand so that you do not **rune** the moment.
NJD	"however, it is **rune**, " he said.
SURGEN	when the millennia pass, i will guide your hand so that you do not **rune** the moment.
LCR	there was nothing but desolation and **rune**.
GOLD	the study of ancient symbols will lead you to **rune**.

Figure 3: Examples of model outputs.

Examples generated by different models are in Figure 3. The sentence generated by **RE** can only be interpreted in one way. **RE+S** and **SURGEN** introduce grammar errors by directly replacing alternative words with pun words. The sentence generated by **NJD** is fragmentary and cannot inspire people to think about another interpretation. **LCR** can generate fluent puns. **GOLD** pun triggers humor in a similar way.

3.6 Error Analysis

We analyze our model outputs one by one and list the overall situation in Figure 4. We find there are mainly five reasons for unsuccessful pun generation:

(1) *Imperfect Constraints*: Pun is a creative language full of imagination. Extracting strongly related words as support words sometimes does not

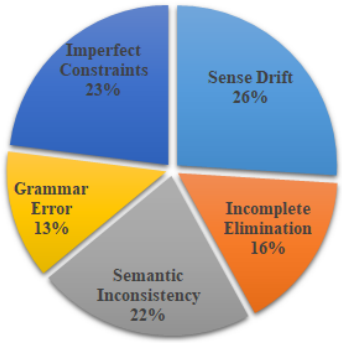

Figure 4: Overview of error types.

work. For example, given "maid-made", "a china doll was marked in tiny letters : maid in usa" is a good pun. However, "doll" is not so related to "maid" and we cannot extract it as a support word.

(2) *Sense Drift*: When rewriting the sentences with constraints, the word collocations can be modified and the sense of the alternative word will disappear, e.g. when "air" means "a distinctive but intangible quality surrounding a person or thing", it is always used as "... an air of", "have a ... air". We rewrite the candidate sentence ("... and with an air of freedom however specious.") with constraints (negative constraints: "air", "freedom"; positive constraints: "err", "principle"). The model outputs "principles err ... and with an atmosphere of liberty, however specious". Instead of the expected collocation "an err of ...", it turns out to be "an atmosphere of liberty". The expected sense of "air" disappears and "principles err" cannot trigger people to think of another interpretation.

(3) *Incomplete Elimination*: When the usages of the given homophone pair are very different, an inflection of the alternative word will be decoded in the result to keep the original semantic meaning of the input sentence. For example, given "sum-sun", the candidate is "add one more likeness ... let me and my sun beget a man". Our model rewrite it as "sum up one more likeness, ... let me and my sunshine beep a man".

(4) *Semantic Inconsistency*: The generated sentence is grammatical but the semantic of the sentence is full of contradictions, e.g. "brilliant mourning lights in the old town", where "mourning" is conflicted with "brilliant" emotionally.

(5) *Grammar Error*: When the usages of two homophones are similar, pun word will replace the alternative word in the rewriting process. However, the alternative word has its own collocations and causes the grammar error in the output texts, e.g., given "zinc-sink", the model outputs "his enemy's solution could **zinc (sink)** him **into** deep dejection". There is much room for our model to improve the semantic consistency by introducing related modules.

4 Conclusion

In this work, we propose to generate homophonic puns with lexically constrained rewriting. We retrieve sentences containing the alternative word as candidate sentences. And then we use constraint selection algorithm to rank candidate sentences and choose support word to imply the semantics of the pun word. Finally we rewrite the sentence with constraints. Our model outperforms previous works on the task of homophonic pun generation. However,as discussion in error analysis, there are several challenges to solve in the future.

5 Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial IntelligenceBAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, pages 633–638.

Kim Binsted and Graeme Ritchie. 1997. Computational rules for generating punning riddles. *HUMOR-International Journal of Humor Research*, 10(1):25–76.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 103–108.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 1734–1744.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 1535–1546.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 24–31, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019a. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 839–850.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019b. PARABANK: monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6521–6528. AAAI Press.

Greg Lessard. 1992. Computational modelling of linguistic humour: Tom swifties. In *Selected Papers from the 1992 Association for Literary and Linguistic Computing (ALLC) and the Association for Computers and the Humanities (ACH) Joint Annual Conference*, pages 175–178. Oxford University Press.

Greg Lessard and Michael Levison. 1992. Computational modelling of linguistic humour: Tom swifties. In *In ALLC/ACH Joint Annual Conference, Oxford*, pages 175–178.

Fuli Luo, Shunyao Li, Pengcheng Yang, Lei Li, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Pun-gan: Generative adversarial network for pun generation. In *EMNLP-IJCNLP 2019*, pages 3386–3391.

Rui Mao, Chenghua Lin, and Frank Guerin. 2018. Word embedding and WordNet based metaphor identification and interpretation. In *ACL 2018 (Volume 1: Long Papers)*, pages 1222–1231, Melbourne, Australia. Association for Computational Linguistics.

Justin McKay. 2002. Generation of idiom-based witticisms to aid second language learning. *Stock et al.(2002)*, pages 77–87.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. SemEval-2017 task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68, Vancouver, Canada. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *NAACL-HLT 2018*, pages 1314–1324.

Graeme Ritchie, Ruli Manurung, Helen Pain, Annalu Waller, Rolf Black, and Dave O'Mara. 2007. A practical application of computational humour. In *Proceedings of the 4th International Joint Conference on Computational Creativity*, pages 91–98.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1660, Melbourne, Australia. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015,*

Santiago, Chile, December 7-13, 2015, pages 19–27.