Pun Generation with Surprise

He He[1]* and **Nanyun Peng**[2]* and **Percy Liang**[1]
[1]Computer Science Department, Stanford University
[2]Information Sciences Institute, University of Southern California
{hehe,pliang}@cs.stanford.edu, npeng@isi.edu

Abstract

We tackle the problem of generating a pun sentence given a pair of homophones (e.g., "*died*" and "*dyed*"). Supervised text generation is inappropriate due to the lack of a large corpus of puns, and even if such a corpus existed, mimicry is at odds with generating novel content. In this paper, we propose an unsupervised approach to pun generation using a corpus of unhumorous text and what we call the *local-global surprisal principle*: we posit that in a pun sentence, there is a strong association between the pun word (e.g., "*dyed*") and the distant context, as well as a strong association between the alternative word (e.g., "*died*") and the immediate context. This contrast creates surprise and thus humor. We instantiate this principle for pun generation in two ways: (i) as a measure based on the ratio of probabilities under a language model, and (ii) a retrieve-and-edit approach based on words suggested by a skip-gram model. Human evaluation shows that our retrieve-and-edit approach generates puns successfully 31% of the time, tripling the success rate of a neural generation baseline.

1 Introduction

Generating *creative* content is a key requirement in many natural language generation tasks such as poetry generation (Manurung et al., 2000; Ghazvininejad et al., 2016), story generation (Meehan, 1977; Peng et al., 2018; Fan et al., 2018; Yao et al., 2019), and social chatbots (Weizenbaum, 1966; Hao et al., 2018). In this paper, we explore creative generation with a focus on puns. We follow the definition of puns in Aarons (2017); Miller et al. (2017): "A pun is a form of wordplay in which one sign (e.g., a word or a phrase) suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another sign, for an intended humorous or rhetorical effect." We focus on a typical class of puns where the ambiguity comes from two (near) homophones. Consider the example in Figure 1: "*Yesterday I accidentally swallowed some food coloring. The doctor says I'm OK, but I feel like I've dyed (died) a little inside.*". The *pun word* shown in the sentence ("*dyed*") indicates one interpretation: the person is colored inside by food coloring. On the other hand, an *alternative word* ("*died*") is implied by the context for another interpretation: the person is sad due to the accident.

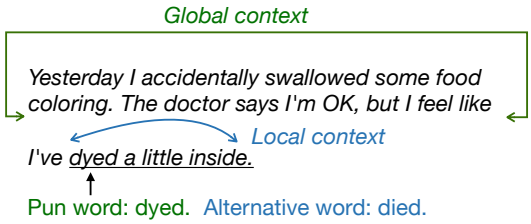

Figure 1: An illustration of a homophonic pun. The *pun word* appears in the sentence, while the *alternative word*, which has the same pronunciation but different meaning, is implicated. The *local context* refers to the immediate words around the pun word, whereas the *global context* refers to the whole sentence.

Current approaches to text generation require lots of training data, but there is no large corpus of puns. Even such a corpus existed, learning the distribution of existing data and sampling from it is unlikely to lead to truly novel, creative sentences. Creative composition requires deviating from the norm, whereas standard generation approaches seek to mimic the norm.

Recently, Yu et al. (2018) proposed an unsupervised approach that generates puns from a neural language model by jointly decoding conditioned on both the pun and the alternative words, thus injecting ambiguity to the output sentence. However, Kao et al. (2015) showed that ambiguity alone is insufficient to bring humor; the two mean-

*Equal contribution.

ings must also be supported by distinct sets of words in the sentence.

Inspired by Kao et al. (2015), we propose a general principle for puns which we call *local-global surprisal principle*. Our key observation is that the strength for the interpretation of the pun and the alternative words flips as one reads the sentence. For example, in Figure 1, "*died*" is favored by the immediate (local) context, whereas "*dyed*" is favored by the global context (i.e. "...*food coloring*..."). Our surprisal principle posits that the pun word is much more surprising in the local context than in the global context, while the opposite is true for the alternative word.

We instantiate our local-global surprisal principle in two ways. First, we develop a quantitative metric for surprise based on the conditional probabilities of the pun word and the alternative word given local and global contexts under a neural language model. However, we find that this metric is not sufficient for generation. We then develop an unsupervised approach to generate puns based on a retrieve-and-edit framework (Guu et al., 2018; Hashimoto et al., 2018) given an unhumorous corpus (Figure 2). We call our system SURGEN (SURprisal-based pun GENeration).

We test our approach on 150 pun-alternative word pairs.[1] First, we show a strong correlation between our surprisal metric and funniness ratings from crowdworkers. Second, human evaluation shows that our system generates puns successfully 31% of the time, compared to 9% of a neural generation baseline (Yu et al., 2018), and results in higher funniness scores.

2 Problem Statement

We assume access to a large corpus of raw (unhumorous) text. Given a pun word w^{p} (e.g., "*dyed*") and an alternative word w^{a} (e.g., "*died*") which are (near) homophones, we aim to generate a list of pun sentences. A pun sentence contains only the pun word w^{p}, but both w^{p} and w^{a} should be evoked by the sentence.

3 Approach

3.1 Surprise in Puns

What makes a good pun sentence? Our key observation is that as a reader processes a sentence, he or she expects to see the alternative word at the pun word position, and are tickled by the relation between the pun word and the rest of the sentence. Consider the following cloze test: "*Yesterday I accidentally swallowed some food coloring. The doctor says I'm OK, but I feel like I've* _____ *a little inside.*". Most people would expect the word in the blank to be "*died*" whereas the actual word is "*dyed*". Locally, "*died a little inside*" is much more likely than "*dyed a little inside*". However, globally when looking back at the whole sentence, "*dyed*" is evoked by "*food coloring*".

Formally, w^{p} is more surprising relative to w^{a} in the local context, but much less so in the global context. We hypothesize that this contrast between local and global surprisal creates humor.

3.2 A Local-Global Surprisal Measure

Let us try to formalize the local-global surprisal principle quantitatively. To measure the amount of surprise due to seeing the pun word instead of the alternative word in a certain context c, we define surprisal S as the log-likelihood ratio of the two events:

$$S(c) \stackrel{\text{def}}{=} -\log \frac{p(w^{\text{p}} \mid c)}{p(w^{\text{a}} \mid c)} = -\log \frac{p(w^{\text{p}}, c)}{p(w^{\text{a}}, c)}. \quad (1)$$

We define the local surprisal to only consider context of a span around the pun word, and the global surprisal to consider context of the whole sentence. Letting $x_1, \ldots, x_n$ be a sequence of tokens, and x_p be the pun word w^{p}, we have

$$S_{\text{local}} \stackrel{\text{def}}{=} S(x_{p-d:p-1}, x_{p+1:p+d}), \quad (2)$$

$$S_{\text{global}} \stackrel{\text{def}}{=} S(x_{1:p-1}, x_{p+1:n}), \quad (3)$$

where d is the local window size.

For puns, both the local and global surprisal should be positive because they are unusual sentences by nature. However, the global surprisal should be lower than the local surprisal due to topic words hinting at the pun word. We use the following unified metric, *local-global surprisal*, to quantify whether a sentence is a pun:

$$S_{\text{ratio}} \stackrel{\text{def}}{=} \begin{cases} -1 & S_{\text{local}} < 0 \text{ or } S_{\text{global}} < 0, \\ S_{\text{local}}/S_{\text{global}} & \text{otherwise.} \end{cases} \quad (4)$$

We hypothesize that larger S_{ratio} is indicative of a good pun. Note that this hypothesis is invalid when either S_{local} or S_{global} is negative, in which case we consider the sentences equally unfunny by setting S_{ratio} to -1.

[1] Our code and data are available at https://github.com/hhexiy/pungen.

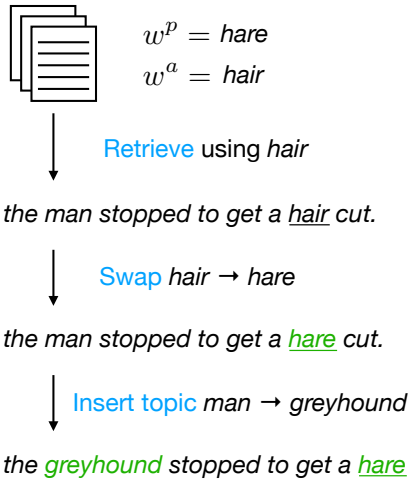

Figure 2: Overview of our pun generation approach. Given a pair of pun/alternative word, we first retrieve sentences containing w^{a} from a generic corpus. Next, w^{a} is replaced by w^{p} to increase local surprisal. Lastly, we insert a topic word at the beginning of the sentence to create global associations supporting w^{p} and decrease global surprisal.

3.3 Generating Puns

The surprisal metric above can be used to assess whether a sentence is a pun, but to generate puns, we need a procedure that can ensure grammaticality. Recall that the surprisal principle requires (1) a strong association between the alternative word and the local context; (2) a strong association between the pun word and the distant context; and (3) both words should be interpretable given local and global context to maintain ambiguity.

Our strategy is to model puns as deviations from normality. Specifically, we mine *seed sentences* (sentences with the potential to be transformed into puns) from a large, generic corpus, and edit them to satisfy the three requirements above.

Figure 2 gives an overview of our approach. Suppose we are generating a pun given w^{p} = "*hare*" and w^{a} = "*hair*". To reinforce w^{a} = "*hair*" in the local context despite the appearance of "*hare*", we retrieve sentences containing "*hair*" and replace occurrences of it with "*hare*". Here, the local context strongly favors the alternative word ("*hair cut*") relative to the pun word ("*hare cut*"). Next, to make the pun word "*hare*" more plausible, we insert a "*hare*"-related topic word ("*greyhound*") near the beginning of the sentence. In summary, we create local surprisal by putting w^{p} in common contexts for w^{a}, and connect w^{p} to a distant topic word by substitution. We describe each step in detail below.

Local surprisal. The first step is to retrieve sentences containing w^{a}. A typical pattern of pun sentences is that the pun word only occurs once towards the end of the sentence, which separates local context from pun-related topics at the beginning. Therefore, we retrieve sentences containing exactly one w^{a} and rank them by the position of w^{a} in the sentence (later is better). Next, we replace w^{a} in the retrieved sentence with w^{p}. The pun word usually fits in the context as it often has the same part-of-speech tag as the alternative word. Thus the swap creates local surprisal by putting the pun word in an unusual but acceptable context. We call this step RETRIEVE+SWAP, and use it as a baseline to generate puns.

Global surprisal. While the pun word is locally unexpected, we need to foreshadow it. This global association must not be too strong that it eliminates the ambiguity. Therefore, we include a single *topic word* related to the pun word by replacing one word at the beginning of the seed sentence. We see this simple structure in many human-written puns as well. For example, "*Old butchers never die, they only meat their fate.*", where pun words and their corresponding topic words are underlined.

We define relatedness between two words w_i and w_j based on a "distant" skip-gram model $p_\theta(w_j \mid w_i)$, where we train p_θ to maximize $p_\theta(w_j \mid w_i)$ for all w_i, w_j in the same sentence between d_1 to d_2 words apart. Formally:

$$\sum_{j=i-d_2}^{i-d_1} \log p_\theta(w_j \mid w_i) + \sum_{j=i+d_1}^{i+d_2} \log p_\theta(w_j \mid w_i). \tag{5}$$

We take the top-k predictions from $p_\theta(w \mid w^{\mathrm{p}})$, where w^{p} is the pun word, as candidate topic words w to be further filtered next.

Type consistent constraint. The replacement must maintain acceptability of the sentence. For example, changing "*person*" to "*ship*" in "*Each person must pay their fare share*" does not make sense even though "*ship*" and "*fare*" are related. Therefore, we restrict the deleted word in the seed sentence to nouns and pronouns, as verbs have more constraints on their arguments and replacing them is likely to result in unacceptable sentences.

In addition, we select candidate topic words that are type-consistent with the deleted word, e.g., replacing "*person*" with "*passenger*" as opposed to

"*ship*". We define type-consistency (for nouns) based on WordNet path similarity.[2] Given two words, we get their synsets from WordNet constrained by their POS tags.[3] If the path similarity between any pair of senses from the two respective synsets is larger than a threshold, we consider the two words type-consistent. In summary, the first noun or pronoun in the seed sentence is replaced by a type-consistent topic word. We call this baseline RETRIEVE+SWAP+TOPIC.

Improve grammaticality. Directly replacing a word with the topic word may result in ungrammatical sentences, e.g., replacing "*i*" with "*negotiator*" and getting "*negotiator am just a woman trying to peace her life back together.*". Therefore, we use a sequence-to-sequence model to smooth the edited sentence (RETRIEVE+SWAP+TOPIC+SMOOTHER).

We smooth the sentence by deleting words around the topic word and train a model to fill in the blank. The smoother is trained in a similar fashion to denoising autoencoders: we delete immediate neighbors of a word in a sentence, and ask the model to reconstruct the sentence by predicting missing neighbors. A training example is shown below:

Original:	`the man slowly walked towards the woods .`
Input:	`<i> man </i> walked towards the woods .`
Output:	`the man slowly`

During training, the word to delete is selected in the same way as selecting the word to replace in a seed sentence, i.e. nouns or pronouns at the beginning of a sentence. At test time, the smoother is expected to fill in words to connect the topic word with the seed sentence in a grammatical way, e.g., "*the negotiator is just a woman trying to peace her life back together.*" (the part rewritten by the smoother is underlined).

4 Experiments

We first evaluate how well our surprisal principle predicts the funniness of sentences perceived by humans (Section 4.2), and then compare our pun generation system and its variations with a simple retrieval baseline and a neural generation model (Yu et al., 2018) (Section 4.3). We show that the local-global surprisal scores strongly correlate with human ratings of funniness, and all of our systems outperform the baselines based on human evaluation. In particular, RETRIEVE+SWAP+TOPIC (henceforth SURGEN) achieves the highest success rate and average funniness score among all systems.

4.1 Datasets

We use the pun dataset from 2017 SemEval task7 (Doogan et al., 2017). The dataset contains 1099 human-written puns annotated with pun words and alternative words, from which we take 219 for development. We use BookCorpus (Zhu et al., 2015) as the generic corpus for retrieval and training various components of our system.

4.2 Analysis of the Surprisal Principle

We evaluate the surprisal principle by analyzing how well the local-global surprisal score (Equation (4)) predicts funniness rated by humans. We first give a brief overview of previous computational accounts of humor, and then analyze the correlation between each metric and human ratings.

Prior funniness metrics. Kao et al. (2015) proposed two information-theoretic metrics: *ambiguity* of meanings and *distinctiveness* of supporting words. Ambiguity says that the sentence should support both the pun meaning and the alternative meaning. Distinctiveness further requires that the two meanings be supported by distinct sets of words.

In contrast, our metric based on the surprisal principle imposes additional requirements. First, surprisal says that while both meanings are acceptable (indicating ambiguity), the pun meaning is unexpected based on the local context. Second, the local-global surprisal contrast requires the pun word to be well supported in the global context.

Given the anomalous nature of puns, we also consider a metric for *unusualness* based on normalized log-probabilities under a language model (Pauls and Klein, 2012):

$$\text{Unusualness} \stackrel{\text{def}}{=} -\frac{1}{n}\log\left(p(x_1,\ldots,x_n)/\prod_{i=1}^{n}p(x_i)\right). \quad (6)$$

Implementation details. Both ambiguity and distinctiveness are based on a generative model

[2] Path similarity is a score between 0 and 1 that is inversely proportional to the shortest distance between two word senses in WordNet.

[3] Pronouns are mapped to the synset `person.n.01`.

Type	Example	SEMEVAL		KAO	
		Count	Funniness	Count	Funniness
Pun	Yesterday a cow saved my life—it was bovine intervention.	33	1.13	141	1.09
Swap-pun	Yesterday a cow saved my life—it was divine intervention.	33	0.05	0	—
Non-pun	The workers are all involved in studying the spread of bovine TB.	64	-0.34	257	-0.53

Table 1: Dataset statistics and funniness ratings of SEMEVAL and KAO. Pun or alternative words are underlined in the example sentence. Each worker's ratings are standardized to z-scores. There is clear separation among the three types in terms of funniness, where pun > swap-pun > non-pun.

Metric	Pun and non-pun				Pun and swap-pun		Pun			
	SEMEVAL		KAO		SEMEVAL		SEMEVAL		KAO	
Surprisal (S_{ratio})	**0.46**	p=0.00	**0.58**	p=0.00	**0.48**	p=0.00	0.26	p=0.15	0.08	p=0.37
Ambiguity	**0.40**	p=0.00	**0.59**	p=0.00	0.18	p=0.15	0.00	p=0.98	0.00	p=0.95
Distinctiveness	-0.17	p=0.10	**0.29**	p=0.00	0.15	p=0.24	**0.41**	p=0.02	**0.27**	p=0.00
Unusualness	**0.37**	p=0.00	**0.36**	p=0.00	0.19	p=0.12	0.20	p=0.27	0.11	p=0.18

Table 2: Spearman correlation between different metrics and human ratings of funniness. Statistically significant correlations with p-value < 0.05 are bolded. Our surprisal principle successfully differentiates puns from non-puns and swap-puns. Distinctiveness is the only metric that correlates strongly with human ratings within puns. However, no single metric works well across different types of sentences.

of puns. Each sentence has a latent variable $z \in \{w^{\text{p}}, w^{\text{a}}\}$ corresponding to the pun meaning and the alternative meaning. Each word also has a latent meaning assignment variable f controlling whether it is generated from an unconditional unigram language model or a unigram model conditioned on z. Ambiguity is defined as the entropy of the posterior distribution over z given all the words, and distinctiveness is defined as the symmetrized KL-divergence between distributions of the assignment variables given the pun meaning and the alternative meaning respectively. The generative model relies on $p(x_i \mid z)$, which Kao et al. (2015) estimates using human ratings of word relatedness. We instead use the skip-gram model described in Section 3.3 as we are interested in a fully-automated system.

For local-global surprisal and unusualness, we estimate probabilities of text spans using a neural language model trained on WikiText-103 (Merity et al., 2016).[4] The local context window size (d in Equation (2)) is set to 2.

Human ratings of funniness. Similar to Kao et al. (2015), to test whether a metric can differentiate puns from normal sentences, we collected ratings for both puns from the SemEval dataset and *non-puns* retrieved from the generic corpus containing either w^{p} or w^{a}. To test the importance of surprisal, we also included *swap-puns* where w^{p} is replaced by w^{a}, which results in sentences that are ambiguous but not necessarily surprising.

We collected all of our human ratings on Amazon Mechanical Turk (AMT). Workers are asked to answer the question "*How funny is this sentence?*" on a scale from 1 (not at all) to 7 (extremely). We obtained funniness ratings on 130 sentences from the development set with 33 puns, 33 swap-puns, and 64 non-puns. 48 workers each read roughly 10–20 sentences in random order, counterbalanced for sentence types of non-puns, swap-puns, and puns. Each sentence is rated by 5 workers, and we removed 10 workers whose maximum Spearman correlation with other people rating the same sentence is lower than 0.2. The average Spearman correlation among all the remaining workers (which captures inter-annotator agreement) is 0.3. We z-scored the ratings of each worker for calibration and took the average z-scored ratings of a sentence as its funniness score.

Table 1 shows the statistics of our annotated dataset (SEMEVAL) and Kao et al. (2015)'s dataset (KAO). Note that the two datasets have different numbers and types of sentences, and the human ratings were collected separately. As expected, puns are funnier than both swap-puns and non-puns. Swap-puns are funnier than non-puns, possibly because they have inherit ambiguity brought by the RETRIEVE+SWAP operation.

[4] https://dl.fbaipublicfiles.com/fairseq/models/wiki103_fconv_lm.tar.bz2.

Automatic metrics of funniness. We analyze the following metrics: local-global surprisal (S_{ratio}), ambiguity, distinctiveness, and unusualness, with respect to their correlation with human ratings of funniness. For each metric, we standardized the scores and outliers beyond two standard deviations are set to $+2$ or -2 accordingly.[5] We then compute the metrics' Spearman correlation with human ratings. On KAO, we directly took the ambiguity scores and distinctiveness scores from the original implementation which requires human-annotated word relatedness.[6] On SEMEVAL, we used our reimplemention of Kao et al. (2015)'s algorithm but with the skip-gram model.

The results are shown in Table 2. For puns and non-puns, all metrics correlate strongly with human scores, indicating all of them are useful for pun detection. For puns and swap-puns, only local-global surprisal (S_{ratio}) has strong correlation, which shows that surprisal is important for characterizing puns. Ambiguity and distinctiveness do not differentiate pun word from the alternative word, and unusualness only considers probability of the sentence with the pun word, thus they do not correlate as significantly as S_{ratio}.

Within puns, only distinctiveness has significant correlation, whereas the other metrics are not fine-grained enough to differentiate good puns from mediocre ones. Overall, no single metric is robust enough to score funniness across all types of sentences, which makes it hard to generate puns by optimizing automatic metrics of funniness directly.

There is slight inconsistency between results on SEMEVAL and KAO. Specifically, for puns and non-puns, the distinctiveness metric shows a significant correlation with human ratings on KAO but not on SEMEVAL. We hypothesize that it is mainly due to differences in the two corpora and noise from the skip-gram approximation. For example, our dataset contains longer sentences with an average length of 20 words versus 11 words for KAO. Further, Kao et al. (2015) used human annotation of word relatedness while we used the skip-gram model to estimate $p(x_i \mid z)$.

[5] Since both S_{ratio} and distinctiveness are unbounded, bounding the values gives more reliable correlation results.

[6] https://github.com/amoudgl/pun-model

Method	Success	Funniness	Grammar
NJD	9.2%	1.4	2.6
R	4.6%	1.3	**3.9**
R+S	27.0%	1.6	3.5
R+S+T+M	28.8%	**1.7**	2.9
SURGEN	**31.4%**	**1.7**	3.0
Human	78.9%	3.0	3.8

Table 3: Human evaluation results of all systems. We show average scores of funniness and grammaticality on a 1-5 scale and success rate computed from yes/no responses. We compare with two baselines: NEURALJOINTDECODER (NJD) and RETRIEVE (R). R+S, SURGEN, and R+S+T+M are three variations of our method: RETRIEVE+SWAP, RETRIEVE+SWAP+TOPIC, and RETRIEVE+SWAP+TOPIC+SMOOTHER, respectively. Overall, SURGEN performs the best across the board.

4.3 Pun Generation Results

Systems. We compare with a recent neural pun generator (Yu et al., 2018). They proposed an unsupervised approach based on generic language models to generate homographic puns.[7] Their approach takes as input two senses of a target word (e.g., *bat.n01*, *bat.n02* from WordNet synsets), and decodes from both senses jointly by taking a product of the probabilities conditioned on the two senses respectively (e.g., *bat.n01* and *bat.n02*), so that both senses are reflected in the output. To ensure that the target word appears in the middle of a sentence, they decode backward from the target word towards the beginning and then decode forward to complete the sentence. We adapted their method to generate homophonic puns by considering w^{p} and w^{a} as two input senses and decoding from the pun word. We retrained their forward / backward language models on the same BookCorpus used for our system. For comparison, we chose their best model (NEURALJOINTDECODER), which mainly captures ambiguity in puns.

In addition, we include a retrieval baseline (RETRIEVE) which simply retrieves sentences containing the pun word.

For our systems, we include the entire progression of methods described in Section 3 (RETRIEVE+SWAP, RETRIEVE+SWAP+TOPIC, and RETRIEVE+SWAP+TOPIC+SMOOTHER).

Implementation details. The key components of our systems include a retriever, a skip-gram

[7] Sentences where the pun word and alternative word have the same written form (e.g., *bat*) but different senses.

Aspect	SURGEN v.s. NJD win %	SURGEN v.s. NJD lose %	SURGEN v.s. Human win %	SURGEN v.s. Human lose %
Success	48.0	5.3	6.0	78.7
Funniness	56.7	25.3	10.7	85.3
Grammar	60.7	30.0	8.0	82.0

Table 4: Pairwise comparison between SURGEN and NEURALJOINTDECODER (NJD), and between SURGEN and human written puns. Win % (lose %) is the percentage among the human-rated 150 sentences where SURGEN achieves a higher (lower) average score compared to the other method. The rest are ties.

model for topic word prediction, a type consistency checker, and a neural smoother. Given an alternative word, the retriever returned 500 candidates, among which we took the top 100 as seed sentences (Section 3.3 local surprisal). For topic words, we took the top 100 words predicted by the skip-gram model and filtered them to ensure type consistency with the deleted word (Section 3.3 global surprisal). The WordNet path similarity threshold for type consistency was set to 0.3.

The skip-gram model was trained on BookCorpus with $d_1 = 5$ and $d_2 = 10$ in Equation (5). We set the word embedding size to 300 and trained for 15 epochs using Adam (Kingma and Ba, 2014) with a learning rate of 0.0001. For the neural smoother, we trained a single-layer LSTM (512 hidden units) sequence-to-sequence model with attention on BookCorpus. The model was trained for 50 epochs using AdaGrad (Duchi et al., 2010) with a learning rate of 0.01 and a dropout rate of 0.1.

Human evaluation. We hired workers on AMT to rate outputs from all 5 systems together with expert-written puns from the SemEval pun dataset. Each worker was shown a group of sentences generated by all systems (randomly shuffled) given the same pun word and alternative word pair. Workers were asked to rate each sentence on three aspects: (1) success ("*Is the sentence a pun?*"),[8] (2) funniness ("*How funny is the sentence?*"), and (3) grammaticality ("*How grammatical is the sentence?*"). Success was rated as yes/no, and funniness and grammaticality were rated on a scale from 1 (not at all) to 5 (very). We also included a N/A choice (does not make sense) for funniness to exclude cases where the sentence are not understandable. Workers were explicitly instructed to try their best to give different scores for sentences in the same group.

[8] They were shown the definition from Miller et al. (2017).

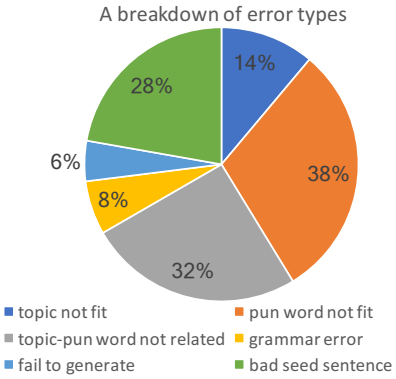

Figure 3: Error case breakdown shows that the main issues lie in finding seed sentences that accommodates both the pun word and the topic word (topic not fit + pun word not fit + bad seed sentence).

We evaluated 150 pun/alternative word pairs. Each generated sentence was rated by 5 workers and their scores were averaged. N/A ratings were excluded unless all ratings of a sentence were N/A, in which case we set its score to 0. We attracted 65, 93, 66 workers for the success, funniness, and grammaticality surveys respectively, and removed 3, 4, 4 workers because their maximum Spearman correlation with other workers was lower than 0.2. We measure inter-annotator agreement using average Spearman correlation among all workers, and the average inter-annotator Spearman correlation for success, funniness, and grammaticality are 0.57, 0.36, and 0.32, respectively.

Table 3 shows the overall results. All 3 of our systems outperform the baselines in terms of success rate and funniness. More edits (i.e. swapping, inserting topic words) made the sentence less grammatical, but also much more like puns (higher success rate). Interestingly, introducing the neural smoother did not improve grammaticality and hurt success rate slightly. Manual inspection shows that ungrammaticality is often caused by improper topic word, thus fixing its neighboring words does not truly solve the problem. For example, filling "*drum*" (related to "*lute*") in "*if that was _____ it was likely that another body would turn up soon, because someone probably wouldn't want to share the lute.*". In addition, when the neural model is given a rare topic word, it tends to rewrite it to a common phrase instead, again showing that supervised learning is against the spirit of generating novel content. For example, inserting "*gentlewoman*" to "*_____ not allow me to ...*" produces "*these people did not allow me to ...*". Overall, our

Method	Example	Rating
1. Pun/alternative word pair: butter – better		
NJD	He is going to come up with the butter a 'very good' approach to the world's economic crisis, the world's biggest economic climate.	1
SURGEN	Well, gourmet did it, he thought, it'd butter be right.	2
Human	Why did the dairy churn? The less said, the butter...	1.5
2. Pun/alternative word pair: peace – piece		
NJD	Further, he said, at the end of the peace, it's not clear that it will be a good example.	1
SURGEN	That's because negotiator got my car back to me in one peace.	1.5
Human	Life is a puzzle; look here for the missing peace.	3
3. Pun/alternative word pair: flour – flower		
NJD	Go, and if you are going on the flour.	1
SURGEN	Butter want to know who these two girls are, the new members of the holy flour.	1.5
Human	Betty crocker was a flour child.	4.5
4. Pun/alternative word pair: wait – weight		
NJD	Gordon Brown, Georgia's prime minister, said he did not have to wait, but he was not sure whether he had been killed.	0
SURGEN	Even from the outside, I could tell that he'd already lost some wait.	2
Human	Patience is a virtue heavy in wait.	3

Table 5: Examples of generated puns with average human ratings of funniness (1-5). 0 means that all ratings are N/A (does not make sense).

SURGEN performs the best and tripled the success rate of NEURALJOINTDECODER with improved funniness and grammaticality scores. Nevertheless, there is still a significant gap between generated puns and expert-written puns across all aspects, indicating that pun generation remains an open challenge.

Table 4 shows the pairwise comparison results among our best model SURGEN, NEURALJOINTDECODER, and expert-written puns. Given the outputs of two systems, we decided win/lose/tie by comparing the average scores of both outputs. We see that SURGEN dominates NEURALJOINTDECODER with $> 50\%$ winning rate on funniness and grammaticality. On success rate, the two methods have many ties since they both have relatively low success rate. Our generated puns were rated funnier than expert-written puns around 10% of the time.

4.4 Error Analysis

In Table 5, we show example outputs of our SURGEN, the NEURALJOINTDECODER baseline, and expert-written puns. SURGEN sometimes generates creative puns that are rated even funnier than human-written puns (example 1). In contrast, NEURALJOINTDECODER at best generates ambiguous sentences (example 2 and 3) and sometimes the sentences are ungrammatical (example 1) or hard to understand (example 4). The examples also show the current limitation of SURGEN. In example 3, it failed to realize that "*butter*" is not animate thus cannot "*want*" since our type consistency checker is very simple.

To gain further insights on the limitation of our system, we randomly sampled 50 unsuccessful generations (labeled by workers) to analyze the issues. We characterized the issues into 6 non-exclusive categories: (1) weak association between the local context and w^{a} (e.g., "*...in the form of a batty (bat)*"); (2) w^{p} does not fit in the local context, often due to different POS tags of w^{a} and w^{p} (e.g., "*vibrate with a taxed (text)*"); (3) the topic word is not related to w^{p} (e.g., "*pagan*" vs "*fabrication*"); (4) the topic word does not fit in its immediate context, often due to inconsistent types (e.g., "*slider won't go...*"), (5) grammatical errors; and (6) fail to obtain seed sentences or topic words. A breakdown of these errors is shown in Figure 3. The main issues lie in finding seed sentences that accommodate both the pun word and the topic word. There is also room for improvement in predicting pun-related topic words.

5 Discussion and Related Work

5.1 Humor Theory

Humor involves complex cognitive activities and many theories attempt to explain what might be considered humorous. Among the leading theories, the incongruity theory (Tony, 2004) is most related to our surprisal principle. The incongruity

theory posits that humor is perceived at the moment of resolving the incongruity between two concepts, often involving unexpected shifts in perspectives. Ginzburg et al. (2015) applied the incongruity theory to explain laughter in dialogues. Prior work (Kao et al., 2015) on formalizing incongruity theory for puns focuses on ambiguity between two concepts and the heterogeneity nature of the ambiguity. Our surprisal principle further formalizes unexpectedness (local surprisal) and incongruity resolution (global association).

The surprisal principle is also related to studies in psycholinguistics on the relation between surprisal and human comprehension (Levy, 2013; Levy and Gibson, 2013). Our study suggests it could be a fruitful direction to formally study the relationship between human perception of surprisal and humor.

5.2 Humor generation

Early approaches to joke generation (Binsted, 1996; Ritchie, 2005) largely rely on templates for specific types of puns. For example, JAPE (Binsted, 1996) generates noun phrase puns as question-answer pairs, e.g., "*What do you call a [murderer] with [fiber]? A [cereal] [killer].*" Petrovic and Matthews (2013) fill in a joke template based on word similarity and uncommonness. Similar to our editing approach, Valitutti et al. (2013) substitutes a word with a taboo word based on form similarity and local coherence to generate adult jokes. Recently, Yu et al. (2018) generates puns from a generic neural language model by simultaneously conditioning on two meanings. Most of these approaches leverage some assumptions of joke structures, e.g., incongruity, relations between words, and word types. Our approach also relies on specific pun structures; we have proposed and operationalized a local-global surprisal principle for pun generation.

5.3 Creative text generation

Our work is also built upon generic text generation techniques, in particular recent neural generation models. Hashimoto et al. (2018) developed a retrieve-and-edit approach to improve both grammaticality and diversity of the generated text. Shen et al. (2017); Fu et al. (2018) explored adversarial training to manipulate the style of a sentence. Our neural smoother is also closely related to Li et al. (2018)'s delete-retrieve-edit approach to text style transfer.

Creative generation is more challenging as it requires both formality (e.g., grammaticality, rhythm, and rhyme) and novelty. Therefore, many works (including us) impose strong constraints on the generative process, such as Petrovic and Matthews (2013); Valitutti et al. (2013) for joke generation, Ghazvininejad et al. (2016) for poetry generation, and Yao et al. (2019) for storytelling.

6 Conclusion

In this paper, we tackled pun generation by developing and exploring a local-global surprisal principle. We show that a simple instantiation based on only a language model trained on non-humorous text is effective at detecting puns (though is not fine-grained enough to detect the degree of funniness within puns). To generate puns, we operationalize the surprisal principle with a retrieve-and-edit framework to create contrast in the amount of surprise in local and global contexts. While we improve beyond current techniques, we are still far from human-generated puns.

While we believe the local-global surprisal principle is a useful conceptual tool, the principle itself is not quite yet formalized in a robust enough way that can be be used both as a principle for evaluating sentences and can be directly optimized to generate puns. A big challenge in humor, and more generally, creative text generation, is to capture the difference between creativity (novel but well-formed material) and nonsense (ill-formed material). Language models conflate the two, so developing methods that are nuanced enough to recognize this difference is key to future progress.

Acknowledgments

This work was supported by the DARPA CwC program under ISI prime contract no. W911NF-15-1-0543 and ARO prime contract no. W911NF-15-1-0462. We thank Abhinav Moudgil and Justine Kao for sharing their data and results. We also thank members of the Stanford NLP group and USC Plus Lab for insightful discussions.

Reproduciblility

All code, data, and experiments for this paper are available on the CodaLab platform: https://worksheets.codalab.org/worksheets/0x5a7d0fe35b144ad68998d74891a31ed6.

References

D. Aarons. 2017. Puns and tacit linguistic knowledge. *The Routledge Handbook of Language and Humor, Routledge, New York, NY, Routledge Handbooks in Linguistics*.

K. Binsted. 1996. *Machine Humour: An Implemented Model of Puns*. Ph.D. thesis, University of Edinburgh.

S. Doogan, A. Ghosh, H. Chen, and T. Veale. 2017. Idiom savant at SemEval-2017 task 7: Detection and interpretation of English puns. In *The 11th International Workshop on Semantic Evaluation*.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

A. Fan, M. Lewis, and Y. Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. 2018. Style transfer in text: Exploration and evaluation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

M. Ghazvininejad, X. Shi, Y. Choi, and K. Knight. 2016. Generating topical poetry. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Ginzburg, E. Breithholtz, R. Cooper, J. Hough, and Y. Tian. 2015. Understanding laughter. In *Proceedings of the 20th Amsterdam Colloquium*.

K. Guu, T. B. Hashimoto, Y. Oren, and P. Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics (TACL)*, 0.

F. Hao, C. Hao, S. Maarten, C. Elizabeth, H. Ari, C. Yejin, S. N. A, and O. Mari. 2018. Sounding board: A user-centric and content-driven social chatbot. *arXiv preprint arXiv:1804.10202*.

T. Hashimoto, K. Guu, Y. Oren, and P. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

J. T. Kao, R. Levy, and N. D. Goodman. 2015. A computational model of linguistic humor in puns. *Cognitive Science*.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

R. Levy. 2013. Memory and surprisal in human sentence comprehension. In *Sentence Processing*.

R. Levy and E. Gibson. 2013. Surprisal, the PDC, and the primary locus of processing difficulty in relative clauses. *Frontiers in Psychology*, 4.

J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *North American Association for Computational Linguistics (NAACL)*.

H. Manurung, G. Ritchie, and H. Thompson. 2000. Towards a computational model of poetry generation. *The University of Edinburgh Technical Report*.

J. R. Meehan. 1977. TALE-SPIN, an interactive program that writes stories. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

T. Miller, C. Hempelmann, and I. Gurevych. 2017. SemEval-2017 task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

A. Pauls and D. Klein. 2012. Large-scale syntactic language modeling with treelets. In *Association for Computational Linguistics (ACL)*.

N. Peng, M. Ghazvininejad, J. May, and K. Knight. 2018. Towards controllable story generation. In *NAACL Workshop*.

S. Petrovic and D. Matthews. 2013. Unsupervised joke generation from big data. In *Association for Computational Linguistics (ACL)*.

G. Ritchie. 2005. Computational mechanisms for pun generation. In *the 10th European Natural Language Generation Workshop*.

T. Shen, T. Lei, R. Barzilay, and T. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*.

V. Tony. 2004. Incongruity in humor: Root cause or epiphenomenon? *Humor: International Journal of Humor Research*, 17.

A. Valitutti, H. Toivonen, A. Doucet, and J. M. Toivanen. 2013. "let everything turn well in your wife: Generation of adult humor using lexical constraints. In *Association for Computational Linguistics (ACL)*.

J. Weizenbaum. 1966. ELIZA–a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, and R. Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Z. Yu, J. Tan, and X. Wan. 2018. A neural approach to pun generation. In *Association for Computational Linguistics (ACL)*.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*.