AMBIPUN: Generating Puns with Ambiguous Context

Anonymous ACL submission

Abstract

Computational humor has garnered interest in the natural language processing community due to its wide applications to real-world scenarios. One way to express humor is via the use of puns. In this paper, we propose a simple yet effective way to generate pun sentences that does not require any training on existing puns. Our approach is inspired by humor theories that ambiguity comes from the context rather than the pun word itself. Given a pair of definitions of a pun word, our model first produces a list of related concepts through a reverse dictionary. We then utilize one-shot GPT3 to generate context words and then generate puns incorporating context words from both concepts. We also investigate how the position of a pun word appearing in the sentence will influence the generation results. Human evaluation shows that our method successfully generates pun 52% of the time, outperforming well crafted baselines and the state-of-the-art models by a large margin.

1 Introduction

Humor has the tendency to provoke laughter and provide amusement. By creating an engaging and conducive environment, it is one of the most important forms of human communication (Booth-Butterfield and Wanzer, 2018). Teaching computers to understand and generate humorous texts such as puns pave the way for various practical applications, such as improving creativity in machine-aided writing and making chat-bots more engaging.

In this paper, we tackle the problem of generating homographic puns (Miller et al., 2017): two or more meanings of a **polysemy** for an intended humorous or rhetorical effect. For example, the three punning jokes listed in Figure 1 exploits two contrasting meanings of the word *sentence*: 1) a string of words that are grammatical, and 2) the punishment by a court assigned to a guilty person. Compared with heterographic puns where the ambiguity comes from two near homophones spelled in a different way, the challenge of processing homographic puns is even bigger: we must differentiate contrasting senses of words that sound and are spelled in the same way.

Sense 1 Definition	a string of words that is complete in itself, typically containing a subject and predicate
Sense 2 Definition	(criminal law) a final judgment of guilty in a criminal case and the punishment that is imposed
Ours 1	The sentence is ungrammatical. The jury didn't hear it.
Ours 2	I'm sorry I said the sentence was too long but punishments are endless.
Human	The Judge has got a stutter. Looks like I am not getting a sentence.

Figure 1: An illustration of homographic puns. The target pun word *"sentence"* and the two sense definitions are given to the model as input, and the desired outputs are many punning sentences. To make the target word interpretable in both senses, we propose to include context words related to both sense definitions. We highlight the context words of each sense in blue and pink.

Due to the lack of sizable training data, existing approaches to generate puns are all heavy-weighted in order to *not* rely on pun sentences for training. For example, (Yu et al., 2018) train a constrained neural language model Mou et al. (2015) from a general text corpus, and then use a joint decoding algorithm to guarantee that both definitions of the target pun word will make sense in the generated sequence. He et al. (2019) propose a local-global surprisal principle, and Luo et al. (2019) leverage the Generative Adversarial Nets (Goodfellow et al., 2014) to encourage ambiguity of the outputs via reinforcement learning. We, on the other hand, propose a simple yet effective way to tackle this problem: *encouraging ambiguity by incorporating context words related to each sense*.

Inspired by humor theories (Lippman and Dunn, 2000), we hypothesize that it is the contextual connections rather than the pun word itself that are

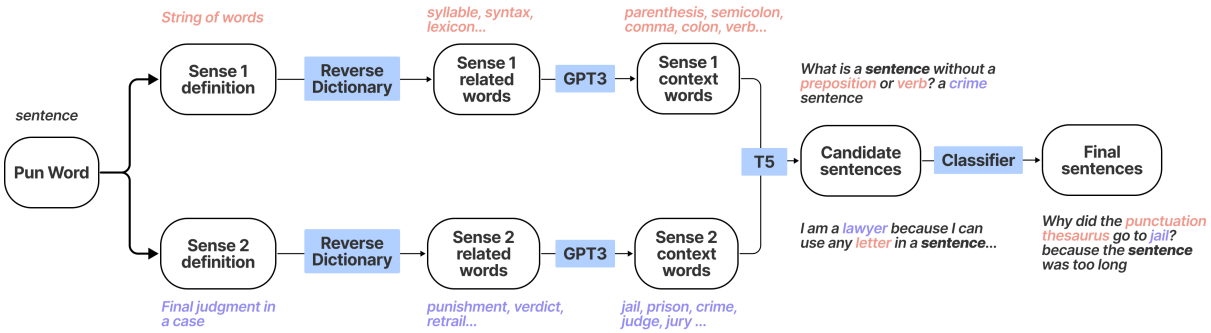

Figure 2: Overview of the approach. Given a pun word and its senses, we convert them to their sense definitions. then we use reverse dictionary to obtain the related words. Using few-shot GPT3, we generate context word for each related word. Using a combination of context words along with the pun word, we generate several candidates sentences using T5. Finally, we use a classifier to choose the most humorous sentences. We also give an example for pun word 'sentence' for each part of the approach.

crucial for the success of pun generation. For instance, in Figure 1 we observe that context related to both senses (e.g., *ungrammatical* and *jury*) appear in a punning sentence. Such observation is important as the error analysis of the state-of-the-art model (Luo et al., 2019) shows that 46% of the generated sentences fail to be puns due to single word sense, and another 27% fail due to being too general, both of which can be resolved by introducing more context.

Specifically, given the two sense definitions of a target pun word, we first use a reverse dictionary[1] to generate **related words** that are monosemous for both senses. This first step helps us circumvent the obstacle of processing pun words with the same written form. However, related words alone are not enough to generate coherent pun sentences because they are clustered and tend to be synonyms. We hence propose to use **context words** (described in Section 2.3) to link the contrasting senses and make our target pun word reasonable when interpreted in both definitions. We explore three different settings: retrieval-based (TF-IDF), similarity-based (Word2Vec), and generative-based (Few-shot GPT3). Finally, we finetune the T5 model (Raffel et al., 2020) on general non-humorous texts to generate coherent sentences given the pun word and contexts words as input.

Interestingly, our experimental results show that to retrieve-and-extract context words outperforms the giant few-shot GPT3 model in terms of generating funny pun sentences, although the latter has shown to be much more powerful in many sophisticated tasks (Brown et al., 2020). Our simple pipeline remarkably outperforms all the more heavy-weighted approaches including the constrained language models with special decoding (Yu et al., 2018) and the state-of-the-art Pun-GAN model (Luo et al., 2019).[2]

2 Methodology

2.1 Overview and Motivation

We first give an overview of different steps in our approach. Our input is the target pun word ($\mathcal{P}$) and its two senses (S_1, S_2), and the expected output is a *list* of humorous punning sentences where $\mathcal{P}$ can be interpreted in both senses. We implement the ambiguity principle proposed in (Kao et al., 2016): a pun sentence should contain one or more context words corresponding to each of the two senses of the pun word.

The overview of our approach is visualized in Figure 2 and formally written in Algorithm 1. Given two different sense descriptions ($SenseDef$), we first use a reverse dictionary to generate a list of words that semantically match the query descriptions. We call them **related words** (RW) and describe full details in Section 2.2. However, those related words are synonyms of each other and only cover a few focused topics, thus failing to compose humorous punning sentences. For example, for the sentence: "The Judge has got a stutter. Looks like I am not getting a sentence.", The word representing the first sense (i.e. a final judgment of guilty in a criminal case and the punishment that is imposed) is represented by *Judge*. *Judge* could not be generated using the sense definition, but is used frequently along with the sense definition.

[1] https://reversedictionary.org/

[2] Our code and data will be released upon acceptance.

Algorithm 1 Pun Generation

function GENPUN(P, S_1, S_2)
Input: Tuple of pun word and its sense - P, S_1, S_2
Output: List of final sentences - $Sent_{final}$
for P, S_1, S_2 in P, S_1, S_2 **do**
    $SenseDef$ = get_sense_definitions(S_1,S_2)
    RW = reverse_dictionary($SenseDef$) ▷ *get related words from sense definitions*
    $RW_{refined}$ = refine(RW)
    CW = get_context_words($RW_{refined}$) ▷ *get context words from related words*
    $CW_{refined}$ = refine(CW)
    $Sent_{candidates}$ = generate_sentences($CW_{refined}$)
    $Sent_{final}$ = classify_sentences($Sent_{candidates}$)
    return $Sent_{final}$

Algorithm 2 TF-IDF for context words

function GETCW(KW_1, KW_2)
Input: List of related words - KW_1, KW_2
Output: List of context words - CW_1, CW_2
Initialize CW_1, CW_2 to empty
for KW in zip(KW_1, KW_2) **do**
    for w in KW **do**
        S = generate_sentences(w) ▷ *Retrieve sentences for each word*
        kw = generate_keywords(S)
        F_S = get_freq_sentences(kw)
        F_C = get_freq_corpora(kw)
        FK = get_tfidf(F_S ,F_C)
        CW_1 = append(FK) ▷ *Add top TF-IDF words to context words list*
    return CW_1, CW_2

Considering such nuances, in Section 2.3 we propose three different methods to obtain the **context words** (CW_1, CW_2) of all the related words. They are TF-IDF (retrieve-and-extract), similarity-based (word2vec), and generative model (few-shot GPT3). Finally, in Section 2.4 and Section 2.5, we introduce a keyword-to-text generator to generate candidate sentences ($Sent_{candidates}$), and a humor classifier to rule out some of the non-pun sentences. Final sentences ($Sent_{final}$) are then randomly sampled for evaluation.[3] All our training data is general, non-humorous corpus except for the humor classifier.

2.2 Related words

We aim at differentiating the two senses of a polysemy by taking the related words, so that each sense will be represented by a set of monosemous words. To this end, we leverage the Reverse Dictionary (Qi et al., 2020; Zhang et al., 2020) which takes as input a description and generates multiple related words whose semantic meaning match the query description. For each sense definition, we generate five words.

2.3 Context words

For context words, we compare three different approaches. Refinement of the context words is mentioned in Section 3.2.

Method 1: TF-IDF For each related word, we retrieve sentences from the One Billion Word dataset that contains that word and then extract a few keywords. Next, we implement TF-IDF (Ramos, 2003) to rank them. For a given word and corpora, the TF-IDF value is given in Equation 1, where $\mathcal{F}_S$ corresponds to the frequency of that word in the retrieved sentences and $\mathcal{F}_C$ corresponds to the frequency of the word in the entire corpora. Based on this value, we choose the top 10 context words that are mostly likely to be used along with the related words and therefore the pun word. Detailed steps for the process are listed in Algorithm 2.

$$tf(\mathcal{W}, \mathcal{C}) = \frac{\mathcal{F}_S}{\mathcal{F}_C + 1} \quad (1)$$

Method 2: Word2Vec Inspired by the idea that "a word is characterized by the company it keeps" (Firth, 1957), we propose to get context words from word2vec (Mikolov et al., 2013), which provides distributed word representations. Following a previous work (Ghazvininejad et al., 2016), we train a continuous-bag-of-words model with window size 40 and word vector dimension 200, and then calculate the cosine similarity between words. Ghazvininejad et al. (2016) have also shown that the training corpus for word2vec plays a crucial role on the quality of generated context words. Hence, we try to train word2vec models on three different corpus: the largest available humorous Dataset, rJokes (Weller and Seppi, 2020), the English Gigaword (Graff et al., 2003) which is an archive of newswire text data, and the one-billion Wikipedia corpus[4]. We find that the topics covered by rJokes is far from what it needs to train a good word2vec model, and that the word2vec model trained on Gigaword strongly favors newsy words than the others. Hence, we train on Wikipedia.

Method 3: GPT3 For the generative version, we use the powerful language model, one-shot GPT3

[3] Note that all previous works produce only the *best* sentence during decoding time, while we aim at generating *tens or hundreds* of sentences for a target pun word, so that our task is actually more challenging.

[4] http://mattmahoney.net/dc/enwik9.zip

(Brown et al., 2020) to generate context words. We choose not to train another model because the output of one-shot GPT3 is already satisfactory. An example can be seen in Table 1, where we compare the output of context words for the pun word 'sentence'.

2.4 Candidate Sentence generation

After receiving context words for each sense, we generate humorous puns. For this step, we finetune a keyword-to-sentence model using T5 (Raffel et al., 2020), as it is capable of handling text-to-text tasks. To train this we need to create a dataset that replicate the expected behaviour i.e. given a prompt (in a specific format), generate a well formed and humorous sentence. The prompt will contain information about the pun word (P), and 2 words from each of the two senses (S_{1a}, S_{1b}, S_{2a}, S_{2b}). We expect our generated output to contain the pun word and context words related to each sense. The following prompt is given to the trained model:

generate sentence: P, S_{1a}, S_{1b}, S_{2a}, S_{2b}.

For example for the word 'sentence', a possible prompt can be *generate sentence: sentence, judge, trail, noun, comma.* However, we also investigate whether the position of the pun word will affect the quality of generated sentences. We insert the pun word in the start (first place), middle (third place), and end (fifth place) of the prompt and generate candidate sentences using these prompt configurations. We discuss our findings in Section 4.3.

2.5 Humor Classification

Finally, we introduce a classification model to assist us in selecting (i.e., ranking) punning sentences. Since we do not have sizable training data for puns, we propose to train our classification model on humorous dataset in a distantly supervised fashion. Specifically, we train BERT-large (Devlin et al., 2019) on the ColBERT dataset that contains 200,000 jokes and non-jokes used for humor detection. We use the probability produced by the classification model to rank our candidate sentences.

Our error analysis shows that our distantly supervised classification model can successfully rule out the bad generations, i.e., non-puns, as puns are humorous by nature. However, the model is not great at choosing the best samples.[5]. Therefore, we use this classifier only to remove the bottom third candidates. We leave this for open future work to accurately pick out high-quality punning sentences instead of funny sentences.

3 Experiments

3.1 Datasets

Training dataset: For the context word generation steps, we use the One Billion word dataset (Chelba et al., 2013) to retrieve sentences for.a given word. To calculate TF-IDF, we use this dataset to calculate the frequency of words. This dataset contains roughly 0.8B words and is obtained from WMT 2011 News crawl data.

For training the candidate generation module, we use ColBERT dataset (Khattab and Zaharia, 2020). It contains 100k positives and 100k negative samples collected from various sources like Reddit, news headlines, etc. For each sentence, we extract the keywords using RAKE (Rose et al., 2010). We also use the same data to finetune BERT-large to develop our humor classifier.

Evaluation dataset: On lines of other recent pun generation works, we use the SemEval 2017 Task 7 (Miller et al., 2017) for evaluation. The dataset contains 1,163 human written pun sentences with a total of 895 unique pun words. Each sentences has the target pun word, location of the pun word and the WordNet sense keys of the two senses.

3.2 Implementation Details

Experimental Settings For the word2vec model we train a continuous-bag-of-words model with window size 40 and word vector dimension 200. For the candidate generation module, we train the T5-base model on 10 epochs and select the best performing model based on validation loss. Max sequence length for target and source is set to 30. Batch size is set to 64.

Data Refinement The process to generate keywords (i.e., both related and context words) can entail many words that are not ideal. Continuing with these words would further propagate and enlarge the noise. Hence, to minimize this noise, we implement the following data refinement steps to ensure the keywords stick to our standards: we avoid using polysemous words as keywords during

[5] A few samples along with their assigned probabilities can be seen in Table **??** in the appendix

	Sense 1	Sense 2
Definition	a string of words satisfying the grammatical rules of a language	a final judgment of guilty in a criminal case and the punishment that is imposed
Related words	syllable, syntax, lexicon, thesaurus, grammatical	punishment, verdict, sentencing, retrial, penalty
TF-IDF	syllables, words, three, spelling, even, said, describe, typos	cruel, expected, end, court, scheduled, set, spector, seeking
Word2Vec	syllable, pronounced, words, rhyme, verbs, meaning, hence, example	punished, crimes, offender, torture, moral, guilt, abuse, offender
GPT3	words, letters, punctuation, grammar, synonym, dictionary, meaning, comma	prison, judge, jury, trial, justice, lawyer, court, evidence

Table 1: Comparison of the three different context word generation mechanism. We take this example for the word 'sentence'. The table lists sense definitions of the two senses. Then list the related words obtained from the sense definitions. For these related words, we obtain context words using three different mechanisms.

intermediate steps because their perceived sense is highly ambiguous. We also disregard any numbers and special characters produced by our systems.

3.3 Baselines

There are two existing works on homographic pun generation, the same task as ours. Besides, we also compare our model with the powerful few-shot learner, GPT3 (Brown et al., 2020).

Neural Pun Yu et al. (2018) propose the first neural approach to homographic puns based on a conditional neural language model. A constrained beam search algorithm is proposed to jointly decode the two distinct senses of the same word.

Pun-GAN The state-of-the-art-model introduced by Luo et al. (2019) that adopts the Generative Adversarial Net (GAN) (Goodfellow et al., 2014) to generate homographic puns. Specifically, a generator is responsible for generating a pun sentence, and a discriminator is trained to tell human-written puns from machine generated puns. Such setting encourages the ambiguity of the generated sentence via reinforcement learning (RL).

Few-shot GPT3 We also generate puns with a few examples feeding into GPT3 davinci-instruct-beta, the most capable model in the GPT3 family to follow the instructions and generate creative language.[6] We provide the target pun word and its two senses in our prompt along with the instruction.

Ablations of our own models We also compare three methods proposed by us to obtain the context words (described in Section 2.3). We call them Ext AMBIPUN, Sim AMBIPUN, and Gen AMBIPUN.

[6]https://beta.openai.com/docs/engines/instruct-series-beta

3.4 Evaluation

Automatic Evaluation Previous works use two metrics to automatically evaluate the quality of the generated puns. First, both (He et al., 2019) and Luo et al. (2019) report the the *unusualness* of an n-length output, which is defined as the normalized log-probability of each token x_i subtracted by its training probability under a language model

$$\mathrm{U} \triangleq -\frac{1}{n} \log \left(p\left(x_1, \ldots, x_n\right) / \prod_{i=1}^{n} p\left(x_i\right) \right). \quad (2)$$

Although He et al. (2019)further show that unusualness does not correlate well with human ratings of puns, we still follow the same procedure. Besides, (Luo et al., 2019) and (Yu et al., 2018) use distinct unigram and bigrams (Li et al., 2015) to measure the *diversity* of each system on a sentence-level. However, we observe that certain systems tend to generate sentences with fixed patterns. Namely, those generation models lack diversity corpus-wise, but could still gain high distinctiveness score sentence-wise. Hence, we propose to measure the diversity from both levels. We also report the the average sentence length produced.

Human Evaluation It is known that currently available automatic evaluation metrics could not reflect the nuances of language, including humor and creativity. Following the procedure of previous works (Yu et al., 2018; He et al., 2019), we randomly shuffle and select 100 sentences for human evaluation. We collected our human ratings on Amazon Mechanical Turk (AMT). For each sentence, three workers are explicitly given the target pun word. We first ask them to judge if a given sentence is a pun sentence on a binary scale. Then, they are asked the questions:"How funny is this

Model	Avg Sequence Length	Sentence-level Diversity		Corpus-level Diversity		Unusualness
		Dist-1	Dist-2	Dist-1	Dist-2	
Few-shot GPT3	12.3	**37.1**	**80.4**	94.5	91.5	0.09
Neural Pun	12.6	30.2	73.0	91.3	90.5	0.22
Pun GAN	9.7	34.6	71.9	90.2	87.6	**0.47**
Sim AMBIPUN	13.4	32.4	77.1	92.9	91.2	0.26
Gen AMBIPUN	13.5	32.8	77.8	93.6	91.2	0.26
Ext AMBIPUN	**14.0**	31.7	78.7	**96.3**	**92.3**	0.28
Human	14.1	36.6	81.9	95.5	92.4	0.35

Table 2: Results of automatic evaluation on average sequence length, sentence-level and corpus-level diversity, and the unusualness scores. Boldface denotes the best performance and underline denotes the second best performance among systems. We compare with three strong baselines: Few-shot GPT3, Neural Pun (Yu et al., 2018), and Pun GAN (Luo et al., 2019), and three variations of our own method: similarity-based context generation(Sim AMBIPUN), generative context generation (Gen AMBIPUN) and extraction-based context generation (Ext AMBIPUN). Note that unusualness has been shown to have weak correlation with human ratings by He et al. (2019).

Model	Success Rate	Fun	Coherence
Few-shot GPT3	13.0%	1.82	**3.77**
Neural Pun	35.3%	2.17	3.21
Pun GAN	35.8%	2.28	2.97
Sim AMBIPUN	45.5%	2.69	3.38
Gen AMBIPUN	50.5%	2.94	3.53
Ext AMBIPUN	**52.2%**	**3.00**	3.48
Human	70.2%	3.43	3.66

Table 3: Human evaluation results on all the pun generation systems. We how the success rates, and average scores of funniness and coherence of each system. Overall, Ext AMBIPUNperforms the best.

sentence?" and "How coherent or fluent is this sentence?" on a scale from 1 (not at all) to 5 (extremely). We provide detailed instructions and examples with explanation for each criteria. We also adopt attention questions and qualification types to make our collected results more reliable. For pun judgement (binary), we take the majority vote among three workers, while for funniness and coherence (1 to 5), we take the average ratings. We then use the pairwise kappa coefficient to measure the inter-annotator agreement. The average inter-annotator agreement of all raters for pun success, funniness and coherence are 0.55, 0.48 and 0.40, meaning that our collected results are reliable.

4 Results and Analysis

4.1 Pun Generation Results

Automatic Evaluation Results of the automatic evaluation can be seen in Table 2. We compare three baselines, three variations of our own model AMBIPUN, and the human written puns. First, the average length of our generated sentence are closest to human written sentences. Although our baseline Pun-GAN has higher distinct ratio at sentence level, we observe that is mainly due to a short sequence length. Moreover, it falls short in corpus-level diversity, meaning that the generated sentences have similar syntax patterns. On the other hand, our Ext AMBIPUN achieves the highest corpus-level diversity. As for unusualness, Pun GAN also obtains unreasonably high score compared with gold. A possible explanation is that the model generate incoherently, which is also verified by our human ratings. Our experimental results resonate with the findings by He et al. (2019) that unusualness does not correlate well with human ratings.

Human Evaluation Results from the automatic evaluation can be seen in Table 3. We evaluate the success rate, funniness, and coherence of the generated outputs. The superiority of our models are obvious. All three of our systems outperform the baselines in terms of success rate and funniness. On the other hand, GPT3 could generate even more coherently than humans.

Analysis between extractive and generative method. Interestingly, Ext AMBIPUN has higher success rates and is funnier than Gen AMBIPUN, indicating that extracting salient words from human written sentences could introduce more surprising and uncommon words than language models. We posit that those atypical words refresh people's eyes and thus boost the pun success rate as well as the funniness score. On the other hand, we also tried to equip GPT3 with greater creatively by top-k sampling with a large temperature T. However, larger Ts also result in arbitrary responses that human may find unreadable. We hope our discovery could

Pun word	Irrational		
Sense 1	Real but not expressible as the quotient of two integers		
Sense 2	Not consistent with or using reason		
Model	**Example**	**Pun**	**Funny**
GPT3	I can't make a decision with all this irrationality going on.	No	1.4
Neural Pun	Note that this means that there is an irrational problem.	Yes	2.4
Pun-GAN	It can be use the irrational system.	No	1.2
Ext AMBIPUN	I have an irrational paranoia about mathematical integers.	Yes	**3.8**
Gen AMBIPUN	My calculator is unjust and illogic. It's irrational.	Yes	3.4
Human	Old math teachers never die, they just become irrational.	Yes	**3.8**
Pun word	**Drive**		
Sense 1	A journey in a vehicle (usually an automobile)		
Sense 2	The trait of being highly motivated		
Model	**Example**	**Pun**	**Funny**
GPT3	I am exhausted, I need a nap before I can drive any more.	No	2.0
Neural Pun	It is that it can be use to drive a variety of function?	No	1.6
Pun-GAN	In he drive to the first three years.	No	1.2
Ext AMBIPUN	What do you call a genius with cunning drive? racecar driver.	Yes	3.6
Gen AMBIPUN	I have the determination to travel to my destination. But i don't have the drive.	Yes	4.0
Human	A boy saving up for a car has a lot of drive.	Yes	**4.2**

Table 4: We show generated sentences for the word 'Irrational' and 'Drive' in the above table, along with their two senses. For the results of our top performing models Gen AMBIPUN and Ext AMBIPUN, we underline the context words that are related to each sense. All the generations are evaluated by external annotators, not the authors.

	Success Rate
Beginning	46.7%
Middle	52.0%
End	54.7%

Table 5: The pun success rate sentences based on their position annotated by human.

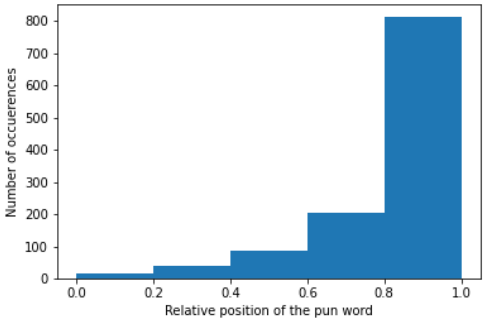

Figure 3: Analysis of the position of pun word in human written puns. The y-axis indicates the number of sentences and the x-axis indicates the position of pun word on a scale from 0 (start) to 1 (end). The analysis is based on 1,163 human written sentences included in the SemEval 2017 Task 7 (Miller et al., 2017).

draw the community's attention to those traditional techniques for creative generation.

4.2 Case Study

To better understand the advantages of our method from a qualitative perspective, we conduct a case study for the pun word "Irrational" and "Drive" and evaluate the generated samples by our top performing models as well as the baselines. The generated outputs along with human evaluation results can be seen in Table 4. For both the examples pun words, at most one of the baselines successfully generates a punning sentence. As discussed earlier, one possible reason is the absence of both senses. On the other hand, both Ext AMBIPUN and Sim AMBIPUN introduce context words for the two senses and thus are able to generate of high quality puns that almost match the human written puns in terms of the funniness score.

4.3 The Position of Pun Words

As is mentioned in Section 2.4, we play with the position of the pun word in the prompt given to the candidate generation model. We try three variants by putting the target pun word at the start, in the middle, and at the end. For each variant, we then ask Mechanical Turkers to judge if the given sentences are puns. Again, each sentence is rated by three Turkers and we take the majority answer if the workers disagree. Results from this analysis can be seen in Table 5. We observe that people find a sentence more likely to be a pun when the target word appears at the end.

To verify such hypothesis, we also calculate the position of the pun words of 1,163 human written pun sentences and report the distribution in Figure 3. The histogram corroborates with the human

annotated samples in that both suggest that keeping the pun word at the end of the sentence generates funnier puns. Theory of humor which says that the "joke" in a funny sentences some towards the end of the sentence (Shahaf et al., 2015) validates our analysis.

5 Related Works

5.1 Creative Language Generation

Pun generation. Many of the previous works on pun generation have focused on phonological or syntactic pattern rather than semantic pattern (Miller and Gurevych, 2015; Hong and Ong, 2009; Petrović and Matthews, 2013; Valitutti et al., 2013) thus lacking creativity and flexibility. He et al. (2019) make use of local-global surprisal principle to generate homophonic puns and Yu et al. (2020) uses constrained lexical rewriting for the same task. Hashimoto et al. (2018) use a retrieve and edit approach to generate homographic puns and Yu et al. (2018); Luo et al. (2019) propose complex neural model architecture such as constrained language model and GAN, and do not put emphasis on the linguistic structure of puns. We identify their absence of both the senses as a shortcoming and build our approach from there.

Figurative language generation. There have been several attempts to generate other types of figurative language such as metaphor, simile (Chakrabarty et al., 2020b), sarcasm, etc. Yu and Wan (2019) use metaphorically used verbs to generate metaphors in an unsupervised fashion. (Chakrabarty et al., 2021) generates metaphors using symbolism and discriminative decoding. Stowe et al. (2021) study diverse metaphor generation using conceptual mapping. Mishra et al. (2019) propose a modular architecture for unsupervised sarcasm generation Chakrabarty et al. (2020a) use commonsense knowledge for the same task. Tian et al. (2021) leverage semantic structure and commonsense and counterfactual knowledge to generate hyperbole.

As for stories, recent works focus on hierarchical story generation that first plans a plot and then writes stories based on the storyline (Martin et al., 2018; Yao et al., 2019; Fan et al., 2019). Goldfarb-Tarrant et al. (2020) incorporates SRL extracted event representations in storylines with several event related decoding objectives.

Humor generation. With the recent advent of diverse datasets (Hasan et al. (2019), Mittal et al. (2021), Yang et al. (2021)), it has become easier to detect and generate humor. While large pre-trained model have fairly successful in detection, humor generation still remains an unsolved problem. Therefore, humor generation is usually studied in a specific setting. Petrović and Matthews (2013) generates joke of the type 'I like my X like I like my Y, Z'. Garimella et al. (2020) develops a model to fill blanks in madlibs format to generate humorous sentences and Yang et al. (2020) edit headlines to make them funny. More research is required to generate humorous sentences that are not constrained by their semantic structure.

5.2 Pun detection

Being able to detect puns can be an essential step to generate them as will be evident in the coming sections. SemEval 2017 Task 7 (Miller et al., 2017) introduced the challenge of pun detection, location detection and sense interpretation for homographic and homophonic puns. It also released a dataset which becomes the backbone of our and several other related works. Diao et al. (2019) make use of Gated Attention network to detection homophonic puns. Zou and Lu (2019) introduces a tagging schemes which lets them detect puns as well as their location. They apply this approach to both homophonic and homographic puns.

6 Conclusion

We propose a novel approach towards homographic puns generation. Unlike previous works that are mathematically heavy, our approach is back-boned by the humor theory that ambiguity is achieved by the context. Both automatic and human evaluations show that our model AMBIPUN outperforms the current state-of-the-art model by a significant margin. We also analyze why our extraction-based variation are more humorous than generation-based variation, and investigate the role of the position of pun words, which corresponds with human written sentences. In future work, we want to make a step further and explore the part of speech tags by filtering out the context words based on their POS tags and make combinations accordingly. Another interesting direction could be to apply our proposed approach to set phrases, which also make use of different senses.

References

Melanie Booth-Butterfield and Melissa Wanzer. 2018. Humor in interpersonal communication.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Tuhin Chakrabarty, Debanjan Ghosh, Smaranda Muresan, and Nanyun Peng. 2020a. R^3: Reverse, retrieve, and rank for sarcasm generation with commonsense knowledge. *arXiv preprint arXiv:2004.13248*.

Tuhin Chakrabarty, Smaranda Muresan, and Nanyun Peng. 2020b. Generating similes effortlessly like a pro: A style transfer approach for simile generation. *arXiv preprint arXiv:2009.08942*.

Tuhin Chakrabarty, Xurui Zhang, Smaranda Muresan, and Nanyun Peng. 2021. Mermaid: Metaphor generation with symbolism and discriminative decoding.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Yufeng Diao, Hongfei Lin, Liang Yang, Xiaochao Fan, Di Wu, Dongyu Zhang, and Kan Xu. 2019. Heterographic pun recognition via pronunciation and spelling understanding gated attention network. In *The World Wide Web Conference*, WWW '19, page 363–371, New York, NY, USA. Association for Computing Machinery.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *ACL*.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Aparna Garimella, Carmen Banea, Nabil Hossain, and Rada Mihalcea. 2020. "judge me by my size (noun), do you?" YodaLib: A demographic-aware humor generation framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2814–2825, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. *arXiv preprint arXiv:2009.09870*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Md Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency, and Mohammed (Ehsan) Hoque. 2019. Ur-funny: A multimodal language dataset for understanding humor. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. *arXiv preprint arXiv:1904.06828*.

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*, pages 24–31, Boulder, Colorado. Association for Computational Linguistics.

Justine T Kao, Roger Levy, and Noah D Goodman. 2016. A computational model of linguistic humor in puns. *Cognitive science*, 40(5):1270–1285.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Louis G Lippman and Mara L Dunn. 2000. Contextual connections within puns: Effects on perceived humor and memory. *The journal of general psychology*, 127(2):185–197.

Fuli Luo, Shunyao Li, Pengcheng Yang, Baobao Chang, Zhifang Sui, Xu Sun, et al. 2019. Pun-gan: Generative adversarial network for pun generation. *arXiv preprint arXiv:1910.10950*.

Lara J Martin, Prithviraj Ammanabrolu, Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. 2018. Event representations for automated story generation with deep neural nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of English puns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 719–729, Beijing, China. Association for Computational Linguistics.

Tristan Miller, Christian F 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.

Abhijit Mishra, Tarun Tater, and Karthik Sankaranarayanan. 2019. A modular architecture for unsupervised sarcasm generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6144–6154.

Anirudh Mittal, Pranav Jeevan, Prerak Gandhi, Diptesh Kanojia, and Pushpak Bhattacharyya. 2021. "so you think you're funny?": Rating the humour quotient in standup comedy.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015. Backward and forward language modeling for constrained sentence generation. *arXiv preprint arXiv:1512.06612*.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria. Association for Computational Linguistics.

Fanchao Qi, Lei Zhang, Yanhui Yang, Zhiyuan Liu, and Maosong Sun. 2020. Wantwords: An open-source online reverse dictionary system. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–181.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. *Automatic Keyword Extraction from Individual Documents*, pages 1 – 20.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1065–1074, New York, NY, USA. Association for Computing Machinery.

Kevin Stowe, Tuhin Chakrabarty, Nanyun Peng, Smaranda Muresan, and Iryna Gurevych. 2021. Metaphor generation with conceptual mappings.

Yufei Tian, Arvind krishna Sridhar, and Nanyun Peng. 2021. HypoGen: Hyperbole generation with commonsense and counterfactual knowledge. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1583–1593, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alessandro Valitutti, Hannu Toivonen, Antoine Doucet, and Jukka Toivanen. 2013. "let everything turn well in your wife": Generation of adult humor using lexical constraints. volume 2.

Orion Weller and Kevin Seppi. 2020. The rjokes dataset: a large scale humor collection. In *Proceedings of The 12th language resources and evaluation conference*, pages 6136–6141.

Ziqing Yang, Yiming Cui, Zhipeng Chen, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. 2020. TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 9–16, Online. Association for Computational Linguistics.

Zixiaofan Yang, Shayan Hooshmand, and Julia Hirschberg. 2021. CHoRaL: Collecting humor reaction labels from millions of social media users. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4429–4435, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

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.

Zhiwei Yu and Xiaojun Wan. 2019. How to avoid sentences spelling boring? towards a neural approach to unsupervised metaphor generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 861–871, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhiwei Yu, Hongyu Zang, and Xiaojun Wan. 2020. Homophonic pun generation with lexically constrained rewriting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2870–2876, Online. Association for Computational Linguistics.

Lei Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Multi-channel reverse dictionary model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 312–319.

Yanyan Zou and Wei Lu. 2019. Joint detection and location of English puns. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2117–2123, Minneapolis, Minnesota. Association for Computational Linguistics.

Appendix

A Humor Classifier Results for Selecting Puns

To further discuss the accuracy and recall of our humor classifier, we show a representative output in Table 6. The table contains a few selected sentences ranked my the humor classifier. We also label each sentence as *yes*, *no*, and *maybe* to indicate if it is a pun or not. As discussed in the methodology, we train our classifier on humor dataset. As puns are an important part of humor generation, this model can help rule out some options. Basic theories of humor such as incongruity and surprise apply to both of them. As can be seen in the table, our classifier is able to successfully pull aside unfunny or non-coherent sentences. Looking at the examples at the top and the middle, it can be observed that some better examples are classified lower than others. Making this observation across many pun words, we decided to use the classifier only to rule out the bottom third samples. For the rest of the generations, we randomly sample them.

On manual observation, we realised that when we cherrypick samples, we're able to find many sentences that meet our expectations. Therefore, building a classifier that can accurately find these sentences can increase the accuracy by a large margin. We treat this as an opportunity for future work.

B More Examples of Generated Puns

We compile more examples generated by AMBIPUNin Table 7 for the following pun words: *sentence*, *case*, *bugs*, *delivery*.This table further supports our point that on manual selection our model is able to generate human-like sentences.

Sentence	Rank	Pun
What's the interest rate on a home mortgage? No interest.	1	Yes
My bank said I think they're interested in me. I said no.	2	No
My girlfriend said she had an interest in banking so i loan her a quarter	3	Yes
I have no interest in being a guardian. It's free.	4	Maybe
I've never had interest placed on borrowings. It's a waste of time.	5	Yes
Why did the republican attack the bank? Because it was in its interest.	6	Maybe
What is the republican's strategy? The interest rate.	7	No
What is the most dispensable interest in investment?	8	No
If trump had an interest in president he would make it an president-of-interest.	9	No

Table 6: An example of candidate pun sentences ranked by the humor classifier. As can be seen, the model is able to rule out non-pun sentences but fails to pick out high-quality ones.

Target word	**sentence**
Sense 1	A string of words satisfying the grammatical rules of a language
Sense 2	(Criminal law) a final judgment of guilty in a criminal case and the punishment that is imposed
1	The word jail is a sentence.
2	What's the punishment for using antonyms in a sentence syntax is it a sentence?
3	I'm sorry I said the sentence was too long but punishments are endless.
4	The sentence in the dictionary doesn't sound very guilty.

Target word	**case**
Sense 1	A portable container for carrying several objects
Sense 2	A statement of facts and reasons used to support an argument
1	What's the most durable luggage for a detective? jury case
2	A jury just found a container of leather there's no reason to argue it's a case
3	What do you call a container used for investigation research? a case study
4	Why did the cardboard get into a trial? because it was an investigation case

Target word	**bugs**
Sense 1	General term for any insect or similar creeping or crawling invertebrate
Sense 2	A fault or defect in a computer program, system, or machine
1	Why did the garden restart its computer? it had bugs in it.
2	What do you call a pest that's slow programmer? bugs bug
3	Why did the compost crash? it had bugs in it.
4	What do you call a bug that's disgusting? a glitch in the internet

Target word	**delivery**
Sense 1	the act of delivering or distributing something (as goods or mail)
Sense 2	your characteristic style or manner of expressing yourself orally
1	What did the letter say to the parcel? clear delivery!
2	What do you call a trucking truckdriver with no articulation? delivery driver.
3	The distribution center has a pronunciation dictionary. it's a delivery service
4	What do you call a parcel with no dialogue and an accent? delivery service.

Table 7: More examples generated by Ext AMBIPUN.