"Barking Up the Right Tree", a GAN-Based Pun Generation Model through Semantic Pruning

JingJie Zeng[1], Liang Yang[*1], Jiahao Kang[1], Yufeng Diao[2], Zhihao Yang[1], Hongfei Lin[1]
[1]School of Computer Science and Technology, Dalian University of Technology, Dalian, China
[2]School of Computer Science and Technology, Inner Mongolia Minzu University, Taoliao, China
jjtail@mail.dlut.edu.cn, liang@dlut.edu.cn, kangjiahao@mail.dlut.edu.cn,
diaoyufeng@imun.edu.cn, yangzh@dlut.edu.cn, hflin@dlut.edu.cn

Abstract

In the realm of artificial intelligence and linguistics, the automatic generation of humor, particularly puns, remains a complex task. This paper introduces an innovative approach that employs a Generative Adversarial Network (GAN) and semantic pruning techniques to generate humorous puns. We initiate our process by identifying potential pun candidates via semantic pruning. This is followed by the use of contrastive learning to decode the unique characteristics of puns, emphasizing both correct and incorrect interpretations. The learned features from contrastive learning are utilized within our GAN model to better capture the semantic nuances of puns. Specifically, the generator exploits the pruned semantic tree to generate pun texts, while the discriminator evaluates the generated puns, ensuring both linguistic correctness and humor. Evaluation results highlight our model's capacity to produce semantically coherent and humorous puns, demonstrating an enhancement over prior methods and approach human-level performance. This work contributes significantly to the field of computational humor, advancing the capabilities of automatic pun generation.

Keywords: Pun generation, Semantic pruning, Contrastive learning, GAN

1. Introduction

Puns are a fascinating linguistic phenomenon that contains humor through their unique ability to convey two or more meanings using a single expression. As a cornerstone of wordplay, puns produce a humorous effect by exploiting linguistic ambiguity, creating unexpected connections and interpretations that evoke surprise and delight. One meaning of such an expression is often intuitive and clear, while the other may be more subtle, requiring the reader or listener to have a certain understanding and decoding ability to capture(Attardo, 2010). Puns are often used in our daily lives, as discovering and understanding the hidden meanings in puns often brings people a sense of intellectual satisfaction, which is very similar to our feelings when we comprehend humor. Moreover, many jokes and humorous stories rely on puns to build punchlines(Chiaro, 2006; Hempelmann, 2004; Alexander, 1997).

The task of generating puns is a complex creative work that has recently garnered interest within the realm of academic research(Mittal et al., 2022; Sun et al., 2022a; Tian et al., 2022). Understanding and generating puns are notably difficult tasks. They necessitate a broad spectrum of commonsense and worldly knowledge, which can be challenging even for humans to master and utilize. Despite advancements in large language models like ChatGPT, which appear to make the task of pun generation almost effortless, there's still room for improvement in computational humor. In an evaluation of over 1000 generated jokes, more than 90% were found to be variations of the same 25 jokes(Jentzsch and Kersting, 2023). This demonstrates that ChatGPT and similar models have not yet fully mastered the complex art of humor generation, especially in a pun style.

The majority of current research on pun generation tends to concentrate on creating puns given a pair of words or senses with pun potential. Currently, research approaches to heterographic and homographic puns are distinct. The former involves utilizing pair of homophones, consisting of a pun word and an alternative word(Mittal et al., 2022; Yu et al., 2020), while the latter is characterized by applying various senses of a single polysemous word(Yu et al., 2018; Tian et al., 2022; Sun et al., 2022b). This classification divides puns into two categories: homophonic puns, which exploit similar sounds, and semantic puns, which play on multiple meanings of a word.

Inspired by real-world observations, we notice that puns naturally emerge in language descriptions, such descriptions can carry multiple meanings. This multiplicity of meanings is inherent to the nature of puns, and it's what makes them both interesting and challenging to interpret. Each pun sentence can potentially contribute to multiple interpretations, and depending on the path chosen through these interpretations, different understandings of the sentence can be formed.

*Corresponding author

LREC-COLING 2024, pages 2119–2131
20-25 May, 2024.

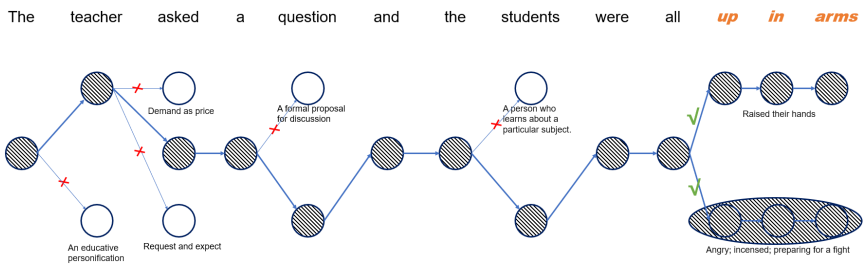

Figure 1: Each small dot under the sentence correlates to the interpretation of a word, and the black dots represents the selected meanings within the sentence. The larger circle at the end represents a phrase, which, if filled in black, indicates its selection as well. Using the initial part of the sentence as an example, paths that are unattainable are marked with a red "X".

Consider the example: "*The teacher asked a question and the students were all* ***up in arms***." This sentence can be interpreted in two ways either the students all raise their hands, or the students are extremely angry. Both interpretations are reasonable, and each represents a different path through the possible meanings of the words in the sentence. This leads us to the concept of a "semantic tree" for understanding pun sentences. In this semantic tree, each word in the sentence is a node, and each possible interpretation of the word is a branch leading from that node. The different paths through the tree represent the different possible interpretations of the sentence. However, not all branches in this tree are relevant or useful for understanding the pun. Some branches may lead to interpretations that are far-fetched or inconsistent with the rest of the sentence. To address this, we propose a method for pruning the semantic tree, which requires cutting out irrelevant branches to leave only the most plausible interpretations. Our research begins by constructing this semantic tree for a given pun sentence, exploring the meaning of each word, and then pruning unnecessary branches. The process of constructing and pruning the semantic tree is demonstrated in Figure 1.

After constructing the semantic tree, our next step is to leverage it. We design a generative model that can produce pun sentences taking the semantic tree as input. This is not straightforward, as it involves navigating the complex structure of the tree and selecting the appropriate branches to form a coherent and punny sentence.

To tackle this challenge, we turn to the power of Generative Adversarial Networks (GANs)(Goodfellow et al., 2014). GANs have shown remarkable success in generating realistic data, and we believe they can be adapted to our task of pun generation. However, simply using a standard GAN model is not sufficient, as it may not fully capture the unique characteristics of puns. Therefore, we propose an innovative approach that incorporates aspects of contrastive learning into the GAN model. Contrastive learning is a technique that encourages the model to produce outputs that are similar to the target data and dissimilar to other data. By introducing these contrastive learning features, we aim to ensure that the artificial pun sentences generated by the GAN can more closely align with the original description's puns, thereby enhancing the quality and authenticity of the generated puns.

The contributions of our work are threefold:

- Our key contribution is the innovative semantic tree approach, effectively capturing the multifaceted meanings of puns. By pruning irrelevant branches, we maintain only meaningful interpretations, thereby boosting the efficiency and accuracy of pun interpretation and generation.
- Our work introduces a novel use of contrastive learning and GANs for pun generation. This method allows the model to encapsulate semantic rules and constraints to puns, reducing dependence on manual annotations and enhancing the diversity and quality of the puns generated.
- Our contribution lies in the **P**un**I**ntended model's robust performance in the generation of puns, as demonstrated by the consistency in our experimental results. Across both homophonic and homographic pun categories, our model achieved competitive scores in line with the leading model and approached human-level performance.

2. Related work

Various approaches have been explored in pun generation research. Some researchers utilize conditional language models and decoding algorithms to create puns, emphasizing two intended senses of a homographic word(Yu et al., 2018; Cai et al., 2018; Hashimoto et al., 2018). Some studies have provided polysemous words and attempted to integrate their various interpretations to create pun(Mittal et al., 2022). Others have generated puns by rewriting ordinary sentences to express different semantic meanings of two homophones(Yu et al., 2020). Another line of research

has employed GANs, where the generator produces sentences that contains two meanings of a target word(Luo et al., 2019). A further approach incorporates key linguistic characteristics of puns, such as ambiguity, distinctiveness, and surprise, to enhance pun quality(Tian et al., 2022). However, these methods usually handle homophonic and homographic puns separately, primarily generating puns from a "pun pair," which may limit their creativity and characteristics.

3. Approach

In this section, we will introduce our model procedure in detail. This encompasses the assembly and pruning of the pun semantic tree, the development and feature extraction processes within the contrastive learning framework, and the specific operational intricacies of the generator and discriminator within our GAN model. Its specific flow chart is shown in Figure 2. We have chosen to name our model **PunIntended (PI)**. This name is a playful take on the familiar phrase "No Pun Intended". However, in our case, the pun is absolutely intended and consistent with our model's main objective, which is the deliberate generation and understanding of puns.

3.1. Pun Semantic Tree

In this subsection, we dive into the process of constructing and refining a "pun semantic tree", a unique structure that contains the multiple meanings inherent in a pun sentence. We start by assembling the tree, using tools such as the Constituency Parser from Stanza(Qi et al., 2020) and WordNet(Fellbaum, 1998) to parse the sentence and query the meanings of each word. We then prune the tree by comparing the similarities of the interpretations using the BERT-large-uncased(Devlin et al., 2018) model, aiming to select the most appropriate interpretations for each word. This process allows us to effectively capture the rich semantic complexity of puns and provides a solid foundation for subsequent pun generation and understanding tasks.

Assembly For instance, the sentence "*The teacher asked a question and the students were all up in arms.*" Depending on the context, this sentence can express two distinct meanings. This prompts us to consider whether a pun sentence might possess a "semantic tree", a collection of paraphrases for all words in the sentence, with puns frequently appearing at the tree's branching points. Accordingly, we have devised a method for generating such a pun-specific semantic tree.

Firstly, for a given original pun sentence $S = [w_1, w_2, ..., w_n]$, the symbol w_n represents each individual word within the pun sentence. We use a suite of toolkits to facilitate our work. To start, we employ the Constituency Parser from Stanza to derive the constituency tree of the pun sentence and ensure to preserve the part of speech for each constituent word. Following this, we utilize WordNet to query the meanings of each word, guided by its part of speech, and carry out an initial screening process. For each word w_n in the sentence, the corresponding interpretations obtained take the form $[i_1, i_2, ..., i_n]$.

Of note is the rationale for obtaining the constituency tree through the Constituency Parser: puns may contain phrases (such as "up in arms") that contribute significantly to their overall effect. Considering that WordNet does not support phrase querying, we have turned to an online dictionary for these inquiries.[1]

Pruning Upon acquiring these interpretations, we are faced with an excessively large set of candidate interpretations for each word, despite preliminary filtering based on the speech part. To select appropriate interpretations, we employ a method to compare similarities, utilizing the BERT-large-uncased model as a foundation.

Consider the sentence: "The teacher asked a question, and the students were all up in arms." For the word "teacher", two potential interpretations emerge. We integrate each interpretation into the context of the sentence, forming two new

Algorithm 1 Assembly and Pruning of the Pun Semantic Tree

Require: Original pun sentence $S = [w_1, w_2, ..., w_n]$
Ensure: Pruned pun semantic tree
$\mathcal{A}$**ssembly**
Derive constituency tree of S using Stanza Constituency Parser
for each word w_n in S **do**
  Query meanings of w_n using its part of speech with WordNet
  Form interpretations $[i_1, i_2, ..., i_n]$ for w_n
end for
$\mathcal{P}$**runing**
Initialize BERT-large-uncased model
for each word w_n in S **do**
  for each interpretation i_m of w_n **do**
    Construct sentence by replacing w_n with i_m in W
    Obtain vector representation v_m of the constructed sentence using BERT
  end for
  Obtain vector representation v_0 of w_n in the original sentence using BERT
  Compare v_0 with each v_m, retain interpretations with similarity above threshold
end for

[1] https://dictionaryapi.dev/

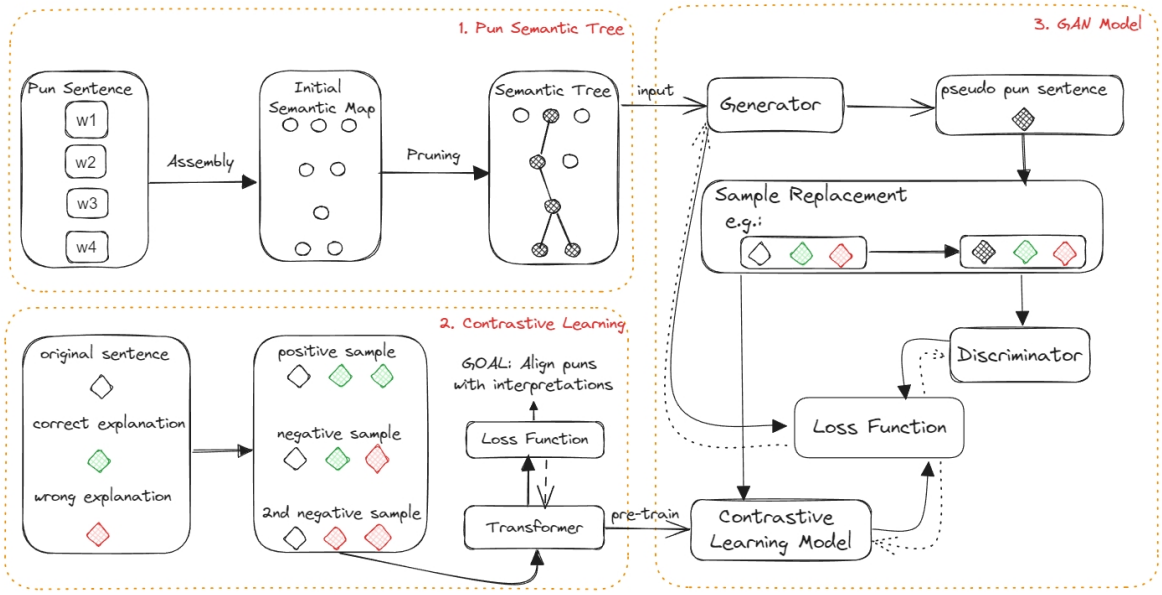

Figure 2: The PunIntended (PI) model, as shown in the figure, effectively abstracts puns into a semantic tree, which in turn can be used to generate puns. The semantic tree helps our model to focus its attention on the most plausible interpretations, thus reducing the complexity of the task and increasing the efficiency of the learning process. By aligning the representations of similar interpretations and distancing dissimilar ones, the semantic tree effectively leverages the characteristics of contrastive learning in the process of pun generation. This approach provides a robust guide for the GAN model.

fragments. When feeding these fragments into the language model, we obtain vector representations v_1 and v_2, in the following format:
Original: "The teacher asked a question and the students were all up in arms."
Sentence 1: "The teacher (a person whose occupation is teaching) asked a question and the students were all up in arms."
Sentence 2: "The teacher (a personified abstraction that teaches) asked a question and the students were all up in arms."

Similarly, we derive a vector representation, v_0, for "teacher" in the original sentence.

The objective is to identify which interpreted context vector (v_1 or v_2) aligns most closely with the original context vector (v_0). By setting the threshold, we can get the pruned pun semantic tree. The specific process is shown in Algorithm 1.

3.2. Contrastive Learning with Puns

In this subsection, we illustrate the development of a contrastive learning model and the extraction of features that are crucial to understanding and generating sentences. Drawing inspiration from the semantic tree of puns, we recognize the need for a systematic exploration of the semantic space of puns. The tree structure provides a rich set of potential interpretations for each word in the sentence, enabling our model to understand the pun from various perspectives. This understanding, in turn, enhances the model's ability to generate and comprehend puns. Consequently, we employ contrastive learning to leverage these insights and further refine our model's capabilities.

The dataset we use in our study comprises pun sentences paired with several (0-5) accurate interpretations Sun et al. (2022a). This dataset is an improvement over the SemEval 2017 Task 7 dataset Miller et al. (2017)[2]. To maximize the use of these precious annotated data, we have trained a contrastive learning model whose objective is to effectively differentiate between puns, correct interpretations, and incorrect interpretations.

Wrong Explanation Selecting positive examples in this dataset is clear-cut, but opting for negative examples, interpretations that neglect the pun, poses a challenge. Such choices markedly influence the contrastive learning model's grasp of the correct versus incorrect explanations. To address this problem, we use the GPT-3.5-Turbo API[3] to generate negative examples, with the prompt details included in the following frame. This method guarantees that the incorrect interpretation does not accurately delineate the pun sentence's meaning, tending more towards a literal interpretation.

Contrastive Learning We have also chosen the BERT-large-uncased model as the foundational model for contrastive learning. However, our goal is to align the representation of pun sentences as

[2] https://alt.qcri.org/semeval2017/task7/

[3] https://platform.openai.com/

I will provide you with a pun sentence as follows:

PUN: Time flies like an arrow; fruit flies like a banana.
RIGHT_EXPLAIN: ["Time passes quickly, as fast as an arrow. However, fruit flies are fond of bananas."]
WRONG_EXPLAIN: ["Time flies in the same way an arrow does; fruit flies in the same way a banana does."]

Your task is to generate similar **"wrong_explain"** sentences based on the provided input. The wrong interpretation should be derived from a nonpunning perspective.

Thus, the expected output format should be:
PUN: {pun},
RIGHT_EXPLAIN: {right_explain},
WRONG_EXPLAIN: {generated_wrong_explain}

Figure 3: The above depicts the prompt input for GPT-3.5-Turbo, where {pun} and {right_explain} serve as placeholders, which will be provided during actual implementation.

closely as possible with correct interpretations and to distance them from incorrect interpretations. To achieve this, we strategically set the input samples and loss functions for comparative learning, which are detailed as follows:

For the construction of input samples in our study, we have adopted a tri-categorization approach. Initially, we generate positive samples by pairing each original text with two correct interpretations, assigning these a label of 1. Subsequently, we construct negative samples by associating each original text with one correct and one incorrect interpretation, which are given a label of 0. Lastly, a distinct category of negative samples is formulated by pairing each original text with two incorrect interpretations, and these are marked with a label of 2. Hence, the final input format is represented as (S, E_1, E_2, L), where S denotes the original sentence, E_1 and E_2 are the interpretations placed at the first and second positions, respectively, and L indicates the type of sample according to our categorization.

In this context, L represents the label of a sample. The distance between any two points, say i and j, is denoted by d_{ij}. For instance, the distance between S and E_1 is represented as d_{12}, and the distance between E_1 and E_2 is denoted as d_{23}. Additionally, m stands for a predefined margin that is employed in the calculation of certain loss functions, and λ symbolizes a scaling factor that adjusts the impact of certain distances within the loss computation.

This structured input sample categorization and notation system is crucial for the clarity and precision required in the computational processes of our study, particularly in the formulation and computation of contrastive loss functions designed to align the model's interpretations of pun sentences with their correct meanings while distancing them from incorrect interpretations.

$\mathcal{L}_1$ calculates the loss for samples with labels 0 or 1, with an aim to ensure higher similarity among similar interpretations. It integrates a term proportional to the squared distance for samples with label 0 and a term involving the square of the margin-adjusted distance for samples $\mathcal{L}_1$:

$$\mathcal{L}_1 = (1-L)\cdot d_{12}^2 + L\cdot \mathsf{max}(0, m-d_{12})^2 + L\cdot \mathsf{max}(0, m-d_{13})^2 \quad (1)$$

$\mathcal{L}_2$ computes the additional loss for label 2 samples, with a goal to align two incorrect interpretations towards similar outputs. This term gets activated only when the $\mathcal{L}_2$:

$$\mathcal{L}_2 = \begin{cases} d_{23}^2 + (d_{12}-d_{13})^2 & \text{if } L=2 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$\mathcal{L}_{add}$ introduces a contrast term that diminishes the similarity between the original text and the incorrect interpretation when the $\mathcal{L}_1$. This is achieved by augmenting the loss when the distance d_{13} between the original and incorrect interpretation exceeds the distance d_{12} between the original and correct interpretation:

$$\mathcal{L}_{\mathsf{add}} = \begin{cases} \mathsf{log}(1+\mathsf{exp}(\lambda\cdot(d_{13}-d_{12}))) & \text{if } L=1, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

$\mathcal{L}_{total}$ is the aggregate of these three terms and represents the total loss:

$$\mathcal{L}_{total} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_{add} \quad (4)$$

The flow of the algorithm is shown in Algorithm 2. Through this loss function, we ensure that our contrastive learning model brings the pun sentence closer to the correct interpretation and distances it from the incorrect interpretation. This approach facilitates effective discrimination and feature extraction for the subsequent GAN model.

3.3. Details of the GAN Model

In this subsection, we state clearly the details of the implementation. In general terms, the architecture consists of a generator and a discriminator. The generator uses the T5 model(Raffel et al., 2020), taking the pun semantic trees, constructed in Section 3.1, as input, and generating pun text as output. The discriminator, on the other hand, is a

Algorithm 2 Contrastive Learning and Feature Extraction

Require: Original pun sentence S, Correct interpretation C, Incorrect interpretation I
Ensure: Feature extracted
Initialize GPT-3.5-Turbo API, BERT-large-uncased model
Initialize m, λ and distance function d
for each sentence S **do**
  Generate incorrect interpretation I using GPT-3.5-Turbo API
  Construct positive, negative samples by pairing S with C and I
  for each sample **do**
    Compute $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_{add}$
    Compute $\mathcal{L}_{total}$
    Optimize model parameters with $\mathcal{L}_{total}$
  end for
end for

BERT-large-uncased model. It utilizes the $[CLS]$ token representation from the final hidden layer as a feature for classification.

Process details We denote the original pun sentence as S, the pun interpretation tree as T, and the sample pair constructed for contrastive learning as (E_1, E_2), with E_1 and E_2 representing two explanations, and L as the label. We initially perform a simple concatenation of the interpretations of the semantic tree, using $[PlaceHolder]$ as the linking symbol, followed by encoding. For added randomness, we randomly mask the interpretation of some words before encoding, signifying the masked segments with $[Interpretation]$ and setting the probability at 15%.

Subsequently, we input this into the generator to obtain the pseudo-data denoted S'. Replacing the original sample pair for contrastive learning with the generated pseudo-data, we transition from (S, E_1, E_2, L) to (S', E_1, E_2, L), and feed this into the contrastive learning model to obtain the contrastive learning loss for the pseudo-data, denoted as $\mathcal{L}_{\text{pseudo}}$. The original contrastive learning loss is denoted as $\mathcal{L}_{\text{true}}$.

We define the feature extracted from pun sentence S using contrastive learning as $\mathcal{F}_{\text{true}}$ and the feature of the pseudo-data as $\mathcal{F}_{\text{pseudo}}$. The similarity between these two features is represented by $\mathcal{P}_{\text{similarity}}$. It's crucial that the generated pseudo-data is not too similar to the true data. This precaution ensures that there isn't a direct copy-paste from the true data to the pseudo-data, preventing them from being semantically very close.

The loss function for the discriminator, which distinguishes between the original and pseudo-data, is calculated as:

$$\mathcal{L}_D = \text{BCE}(D(x_{\text{true}}), 1) + \text{BCE}(D(x_{\text{pseudo}}), 0) + \lambda \cdot \text{CL} \quad (5)$$

where CL is the contrastive loss represented by $|\mathcal{L}_{\text{true}} - \mathcal{L}_{\text{pseudo}}|$, BCE is the binary cross-entropy loss function, $D(x_{\text{true}})$ and $D(x_{\text{pseudo}})$ are the discriminator's outputs for true and pseudo-data, respectively, and λ is the weight of the contrastive loss, serving to adjust the importance of the contrastive loss within the total loss.

After updating the parameters of the discriminator and reevaluating the pseudo-data, we calculate the generator's loss function. The total loss for the generator is the sum of the pseudo-data loss and the weighted contrastive loss:

$$\mathcal{L}_G = \text{BCE}(D(G(z)), 1) + \lambda \cdot \mathcal{L}_{\text{pseudo}} \quad (6)$$

where the generator is only utilize the pseudo-data contrastive loss.

Then, we calculate the loss for the generator T5 model, and the total loss for the generator is calculated as:

$$\mathcal{L}_{G_{\text{all}}} = \mathcal{L}_G + \mathcal{L}_{\text{T5}} + \mathcal{P}_{\text{similarity}} \quad (7)$$

Subsequently, we update the parameters of the generator according to the newly derived loss. Finally, the parameters of the contrastive learning model are updated, with the loss being:

$$\mathcal{L}_C = \mathcal{L}_{\text{true}} + \mathcal{L}_{\text{pseudo}} \quad (8)$$

As described in Algorithm 3, we describe the pseudo-code to combine contrastive learning with GANs. In essence, the pun semantic tree forms the backbone of our approach, guiding the development and feature extraction in contrastive learning and laying a solid foundation for the subsequent GAN model.

Moreover, it is important to stress our rationale for utilizing a GAN and contrastive learning to undertake this task. We have fully exploited manually annotated data such as "why is it a pun" and other similar annotation information. However, not every pun has this kind of annotation, and when generating puns, no such interpretation is provided. Therefore, we employ the synergy of contrastive learning and GAN to incorporate the semantic rules and constraints of puns into the model itself. This approach allows us to transcend the limitations of the existing annotation-based system and generate a broader range of high-quality puns.

4. Experiment

4.1. Experimental Design

Dataset Selection We utilize the ExPUNations (Sun et al., 2022a) dataset, an enhancement of the original SemEval 2017 Task 7 dataset. The dataset includes pun sentences paired with several accurate interpretations, enabling an effective combination of puns and human understanding.

Algorithm 3 Procedure for contrastive learning with GANs

Require: Pun sentence S, Pun interpretation tree T, Sample pair (E_1, E_2), Label L

Encode interpretation tree T into generator to obtain the pseudo-data S'.
Replace original sample pair from (S, E_1, E_2, L) to (S', E_1, E_2, L)
Compute contrastive learning loss $\mathcal{L}_{\text{pseudo}}$ and $\mathcal{L}_{\text{true}}$.

Extract features $\mathcal{F}_{\text{true}}$ and $\mathcal{F}_{\text{pseudo}}$ from S and S'.
Calculate $\mathcal{P}_{\text{similarity}}$ between $\mathcal{F}_{\text{true}}$ and $\mathcal{F}_{\text{pseudo}}$.
Compute discriminator loss $\mathcal{L}_D$.
Update discriminator parameters and reevaluate pseudo-data.
Compute generator loss $\mathcal{L}_G$ and T5 model loss $\mathcal{L}_{\text{T5}}$.

Compute and update parameters based on $\mathcal{L}_{G_{\text{all}}}$.
Update contrastive learning model parameters. The loss is $\mathcal{L}_C = \mathcal{L}_{\text{true}} + \mathcal{L}_{\text{pseudo}}$.

Experimental Setting The learning rate for the contrastive learning model is set at 1e-6, while for the generator and discriminator models in the GAN, the rates are 3e-4 and 1e-6 respectively. Notably, the contrastive learning model undergoes a round of pre-training before being trained in conjunction with the GAN. Additionally, the hyperparameter λ in the GAN is set to 1. All experiments are conducted on a single A100 GPU with 40GB of VRAM.

4.2. Baseline Model

We compare our approach with the following existing baseline:

UnifiedPun(Tian et al., 2022) A model that integrates key linguistic characteristics of puns, ambiguity, distinctiveness, and surprise.

AMBIPUN(Mittal et al., 2022) A model that generates puns by incorporating context words from both senses.

Pun-GAN(Luo et al., 2019) A model that employs GAN to foster ambiguity.

LCR(Yu et al., 2020) The model that initially identifies suitable lexical constraints and subsequently restructures the sentence.

SurGen(He et al., 2019) A model that employs the principle of local-global surprisal in a retrieve-and-edit framework.

Neural Pun(Cai et al., 2018) A model leverages bidirectional LSTM network to model each sequence of word senses in pun tasks.

4.3. Metrics

For the evaluation of pun generation tasks, we implement a dual set of metrics. The traditional metrics include Ambiguity(Kao et al., 2016), Dist-1, and Dist-2. Ambiguity, although a crucial aspect, presents challenges in quantification due to its subjective nature, which may not always accurately reflect the true quality of the pun text. On the other hand, while Dist-1 and Dist-2 excel in measuring lexical diversity, they may overlook semantic diversity, coherence, and readability—key factors in evaluating the quality of generated content. Recognizing these limitations, we argue that these metrics alone do not intuitively capture the characteristics of the pun generation task.

Therefore, we adopt the following strategy: Studies on pun generation have traditionally employed custom, often manually-driven, evaluation metrics. Our research, however, takes a distinct approach. We choose an entirely objective, automated evaluation framework with the LLM (like GPT-4.0 and Claude-2) acting as the central arbiter. Recent studies have shown that GPT-4's opinions significantly overlap with human reviewers(Liang et al., 2023), which points out that GPT-4's opinions aligned with at least one human reviewer 57.55% of the time. Hence, our evaluation spotlights the following essential aspects of puns:

Creativity and Humor(C&H)(Attardo and Raskin, 1991): Good puns typically exhibit creativity, cleverly connecting two or more meanings in a way that elicits laughter or thoughtful reflection. Humor is a key criterion, determining whether the pun can bring about amusement.

Clarity(Cl)(Ritchie, 2004): A pun should be understandable without being overly obscure or hard to grasp. If people need too much time to understand it, it may not be a good pun.

Originality(O)(Koestler, 1964): Good puns should be original rather than repetitive clichés. Originality can increase the appeal of a pun.

Context Sensitivity(CS)(Chiaro, 2006): The effectiveness of a pun should align well the specific context to have the greatest impact. A good pun should fit seamlessly into its surroundings.

Multiple Layers of Meaning(MLM)(Wilson, 2006): Some puns may have multiple layers of meaning, capable of eliciting different interpretations or reactions in various contexts. This richness can enhance their appeal.

For each criterion, the LLM rates puns on a scale of 0-10, with the average scores providing a holistic view. To account for the inherent randomness in the evaluation of LLMs, each assessment was repeated 10 times, and the resulting scores were averaged. The strengths of this method encompass: 1) **Objectivity**: Employing LLM mitigates biases typically associated with manual evaluations. 2) **Depth of Understanding**: With its extensive training data, LLM is adept at discerning the finest linguistic nuances, positioning it as an

Pun pair	mane-main	Score (GPT-4 and Claude-2) C&H	Cl	O	CS	MLM	Avg.
LCR	The mane object of the hair was accomplished.	5.8|5.8	6.1|7.9	5.4|5.3	5.1|6.5	4.3|4.8	5.3|6
SurGen	A trot later, he was sitting away from the mane dining area.	5.5|4.9	6.5|7.3	5.1|4.8	6.3|5.9	4.5|4.3	5.6|5.4
UnifiedPun	In some places, hair also makes up the mane entrance to fashion salons.	6.4|6.7	7.4|6.6	5.4|6.6	8|7.6	5.3|5.5	6.5|6.6
Human	Lions don't have to worry about every little detail in life, just the mane thing.	7.3|7.1	8.4|7.4	5.7|5.9	6.9|7.6	5.4|5.6	6.7|6.7
PI (Ours)	Lions mane-tain that their haircuts are the mane event.	7.7|7.1	7.4|6.7	6.3|5.6	6.7|7.8	6.3|5.3	6.9|6.5
	Even though the tiger kept his mane mostly hairless, he still made a lot of bucks.	6.8|7	7.1|7.8	6.4|6.6	5.5|6.5	6.1|5.4	6.4|6.7
Pun pair	sentence → clause	Score (GPT-4 and Claude-2) C&H	Cl	O	CS	MLM	Avg.
Pun-GAN	Due to the sentence it is in the United States.	2|0.6	1.3|2.3	3|1.9	1.5|0.8	1.2|0.5	1.8|1.22
AmbiPun	The sentence is ungrammatical. The jury didn't hear it.	5.2|3.7	6.3|6.3	5.4|4.9	5.4|5.9	4.2|3.7	5.3|4.9
UnifiedPun	Ours The language on a two-page sentence for fraud is full of guilt.	2.3|2.5	1.4|3.6	3.3|3.8	2.1|2.9	2.1|2.8	2.2|3.1
Human	The judge has got a stutter. Looks like I am not getting a sentence.	7.3|7.6	7.8|7.3	6.7|6.6	6.8|7.3	6|6	6.9|6.96
PI (Ours)	Convict, I find that sentence very punitive.	6|6.2	7.4|7.7	5.2|5.3	6.4|6.9	5.2|5.4	6|6.3
	When a jury is seasoned off, the verdict is almost well-done.	7.8|7.2	6.9|6.2	7.5|7.5	6.8|6.3	6.7|6.4	7.1|6.72

Table 1: Example outputs of different models. Including scoring using LLMs

Sentence	C&H|Cl|O|CS|MLM	Avg.
(a) GPT-4: Why did the grammar teacher serve jail time? For too many incomplete sentences.	8.5|9.5|8|8.5|8	8.5
(b) **PI (Ours)**: If your teacher tells you grammar rules you must completely skip, you might get more out of lesson.	7|8|8|8.5|7.5	7.8
(c) Claude-2: The defendant was outraged when the judge imposed a 20 year sentence. He insisted it was only a short sentence.	9|9.5|8.5|9|8.5	8.9
(d) **PI (Ours)**: The defendant bluffed away his sentence, and retold an innocent truth.	8|9|8.5|9|8	8.5

Table 2: Comparing Our Model with LLMs

unparalleled evaluator for puns.

However, it must be emphasized that the standards we have established are based on specific criteria for evaluating puns. Although these standards are crucial, they may inadvertently affect the results and bypass other evaluation criteria. Moreover, the precise prompts provide to the LLMs can be found in Appendix A, where we have detailed descriptions and processes to guarantee that LLMs comprehend our objectives and tasks accurately. Additionally, to ensure a comprehensive assessment, we compare our model's output with the exemplary puns showcased in another research paper, considering them as prime examples of puns from published works.

4.4. Result and Analysis

In traditional metrics, our results are as shown in Table 3. The outcomes indicate that our model achieves the best performance, with the only exception being slightly lower in Dist-1 compare to the few-shot GPT-3. However, we believe these results do not intuitively demonstrate the quality of our generated outcomes. Hence, we conduct another experiment, directly comparing our results with the showcase examples presented in their papers.

In our experiment, we select the showcase directly from UnifiedPun(Tian et al., 2022) in an effort to maintain consistency with the comparison experiment. In the main body, we focus solely on comparisons using one paper's showcase. Comprehensive experiments conducted to compare other generated showcases are detailed in Appendix B.

Our model's input with the definitions of these two words, although ideally, the input to our model should be a complete sentence. This experiment is conducted twice due to certain inherent randomness in the experimental design, such as the temperature parameters. Finally, the output is evaluated using LLMs for scoring.

Model	**Ambiguity**	**Dist-1**	**Dist-2**
UnifiedPun	20.6	-	-
AmbiPun	17.1	31.7	78.7
GPT-3 (Few-Shot)	-	37.1	80.4
Neural Pun	-	30.2	73.0
Pun GAN	20.1	34.6	71.9
PunIntended (Ours)	23.2	36.4	84.3

Table 3: Evaluation metrics for pun generation models.

As illustrated in Table 1, our model demonstrates impressive performance. Larger models produce a greater number of puns than humans typically generate. The first example, "mane-main", represents a homophonic pun, while the second example, "sentence → clause", is an instance of a parody pun. Despite the input for this comparative experiment being word paraphrasing without context, our model consistently shows its capability. From the outset of the paraphrase tree construction, we have anticipated encountering such pun words. Although we expect inputs to be collections of word definitions, the 15% masking during training frequently leads to the creation of semantically rich and diverse sentences.

4.5. What is the Perfect Pun in LLMs?

We utilize LLMs to assess pun sentences. If a sentence receives a perfect score of 10, we use it as a foundation for our next-generation process. The central inquiry here is: **How does our model**

compare to large language models like GPT-4 in terms of pun generation? In this iteration, instead of inputting the pun pair, our model adopted the pun semantic tree.

Utilizing the pun pair "sentence → clause" as an example, we procure two formal sentences that fulfill the criteria, from both GPT-4 and Claude-2. As shown in Table 2, (a) is generated by GPT-4, while (c) is generated by Claude-2. However, when we revert GPT-4's memory to the evaluation phase and input these sentences, they receive scores of 8.5 and 8.9, respectively. After deriving the pun semantic tree according to our procedure, we feed the two acquired sentences tree into our model. Table 2 also displays the sentence generation results: (b) is derived from (a), and (d) is derived from (c). Upon evaluation, GPT-4 awards sentences with score 7.8 and 8.5, respectively. Actually, our model tends to generate sentence structures that simulate those of the LLMs, at least in terms of vocabulary. The slightly lower score for (b) might be attributed to the sentence losing its relevance to "sentence → clause", making it seem somewhat out of place in this group. However, this evaluation also unveils some critical findings:

Firstly, the incorporation of high-quality semantic trees has been shown to significantly enhance our model's pun generation capabilities, proving the effectiveness of semantic structures in this domain. Secondly, our model demonstrates the ability to mimic the sentence structures of larger models, achieving outcomes that are comparable with those sophisticated models.

4.6. Case Study

Using the classic pun, 'Time flies like an arrow; fruit flies like a banana.' we apply our methodology and observe the following result:

> The butterflies perceived it as a matter of life or death.

Utilizing the method in Subsection 3.1, we analyze why this is a pun and infer that "butterflies" can be interpreted as a feeling of nervousness, which "life or death" serves to emphasize. Alternatively, "butterflies" may be understood in the literal sense, with "life or death" depicting the survival challenges faced by butterflies. Thus, the "butterflies" unfolds two distinct interpretative paths.

In another experiment, inputting the definitions of "poker," "deception," and "opponent" yield:

> When bullets fly out of a hole, they don't bluff.

Though the sentence shows one path in a semantic tree analysis, it still appears as a pun. "Bluff" can mean a gaming tactic or describe bullet behavior. It implies bullets either bluff metaphorically or when shot, head straight to their target without "intimidating". This underscores "bluff's" layered meanings and suggests an oversight in the semantic tree approach, hinting at a new direction for pun studies with metaphor.

5. Conclusion

Puns as a complex rhetorical device of language where sentences convey "dual" meanings. To effectively analyze puns, our work introduces a new model to generate these wordplays, making the pun analysis more accessible and feasible. By creating a semantic tree that encapsulates the intrinsic semantics of puns, we have paved the way for more intuitive pun generation. The assembly and pruning strategies innovated guarantee that this semantic tree only contains coherent interpretations. A contrastive learning model trained on a rich dataset, utilizing a multi-component loss function, promotes correct interpretations while penalizing erroneous ones. We integrate this learning model with a GAN to further refine the training process and enable pun generation precisely.

In addition, our model does not rely on pre-provided pairs of words or senses, presenting a significant stride towards autonomous pun generation. Through a comprehensive set of experiments, our model has proven to outperform current benchmarks, signifying the effectiveness of our approach in contributing to the advancement of linguistic creativity in pun generation.

6. Future work

In the future, our goal is to conduct in-depth research on the distribution of puns, understand their context driven characteristics and inherent creativity. We will consider the potential shift from GAN to Variational Autoencoder (VAE) to achieve more interpretable and controllable pun generation. Our analysis will extend to puns in sentence context and delve into the intricate relationship between metaphors and puns. In addition, how to express emotions vividly and unexpectedly through pun generation will also be an important trend. We will further explore the study of linguistic mechanisms to achieve a deeper understanding of complex linguistic phenomena.

7. Acknowledgements

We thank reviewers for their comments, which provided some insights on this research that will further influence our future work. This work is partially supported by grants from the Natural Science Foundation of China (62076046, 62006130, 62366040, 62076051) and Inner Monoglia Science Foundation (No.2022MS06028).

8. Ethical Consideration

Automating pun generation requires a discourse on certain ethical aspects. Firstly, the potential for misinterpretation and miscommunication may occur, especially in culturally diverse or sensitive settings, given the subjective and complex characteristics of humor. Secondly, the cultural intricacies embedded in humor are also a challenge; our model, despite its innovative semantic tree approach and contrastive learning techniques, may inadvertently generate content that could be perceived as disrespectful or misrepresentative of certain cultures or social groups. Lastly, the inherent risk of bias still exists, akin to other AI systems, where biases in the training data could transpire into the generation of biased puns, potentially existing social prejudices.

9. Limitations

Despite the promising outcomes in pun generation, our model still has limitations. The semantic pruning technique, while effective, could lead to over-pruning, omitting unconventional pun interpretations, reflecting challenges seen in neural machine translation(Gu et al., 2021). Additionally, the model, although competitive, hasn't comprehensively achieved human-level humor comprehension and generation, with the complexity of humor often requiring deep cultural or contextual understanding remaining a challenge(Veale, 2011). The generalizability of our model across different languages and cultures remains untested, indicating areas for future exploration to enhance the model's adaptability and effectiveness in automated pun generation.

10. Bibliographical References

Richard Alexander. 1997. *Aspects of verbal humour in English*, volume 13. Gunter Narr Verlag.

Salvatore Attardo. 2010. *Linguistic theories of humor*, volume 1. Walter de Gruyter.

Salvatore Attardo and Victor Raskin. 1991. Script theory revis (it) ed: Joke similarity and joke representation model.

Yitao Cai, Yin Li, and Xiaojun Wan. 2018. Sense-aware neural models for pun location in texts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 546–551, Melbourne, Australia. Association for Computational Linguistics.

Delia Chiaro. 2006. *The language of jokes: Analyzing verbal play*. Routledge.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT press.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Shuhao Gu, Yang Feng, and Wanying Xie. 2021. Pruning-then-expanding model for domain adaptation of neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3942–3952, Online. Association for Computational Linguistics.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. *Advances in Neural Information Processing Systems*, 31.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1734–1744, Minneapolis, Minnesota. Association for Computational Linguistics.

Christian F Hempelmann. 2004. Script opposition and logical mechanism in punning.

Sophie Jentzsch and Kristian Kersting. 2023. ChatGPT is fun, but it is not funny! humor is still challenging large language models. In *Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 325–340, Toronto, Canada. 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.

Arthur Koestler. 1964. The act of creation.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Ding, Xinyu Yang, Kailas

Vodrahalli, Siyu He, Daniel Smith, Yian Yin, Daniel McFarland, and James Zou. 2023. Can large language models provide useful feedback on research papers? A large-scale empirical analysis. In *arXiv preprint arXiv:2310.01783*.

Fuli Luo, Shunyao Li, Pengcheng Yang, Lei Li, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Pun-gan: Generative adversarial network for pun generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3386–3391. Association for Computational Linguistics.

Anirudh Mittal, Yufei Tian, and Nanyun Peng. 2022. AmbiPun: Generating humorous puns with ambiguous context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1053–1062, Seattle, United States. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Graeme Ritchie. 2004. *The linguistic analysis of jokes*, volume 2. Routledge.

Jiao Sun, Anjali Narayan-Chen, Shereen Oraby, Alessandra Cervone, Tagyoung Chung, Jing Huang, Yang Liu, and Nanyun Peng. 2022a. ExPUNations: Augmenting puns with keywords and explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4590–4605, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiao Sun, Anjali Narayan-Chen, Shereen Oraby, Shuyang Gao, Tagyoung Chung, Jing Huang, Yang Liu, and Nanyun Peng. 2022b. Context-situated pun generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4635–4648, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yufei Tian, Divyanshu Sheth, and Nanyun Peng. 2022. A unified framework for pun generation with humor principles. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3253–3261, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tony Veale. 2011. Creative language retrieval: A robust hybrid of information retrieval and linguistic creativity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287.

Deirdre Wilson. 2006. The pragmatics of verbal irony: Echo or pretence? *Lingua*, 116(10):1722–1743.

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

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.

11. Language Resource References

Miller, Tristan and Hempelmann, Christian and Gurevych, Iryna. 2017. *SemEval-2017 Task 7: Detection and Interpretation of English Puns*. Association for Computational Linguistics. PID https://alt.qcri.org/semeval2017/task7/.

Sun, Jiao and Narayan-Chen, Anjali and Oraby, Shereen and Cervone, Alessandra and Chung, Tagyoung and Huang, Jing and Liu, Yang and Peng, Nanyun. 2022. *ExPUNations: Augmenting Puns with Keywords and Explanations*. Association for Computational Linguistics. PID https://github.com/amazon-science/context-situated-pun-generation.

A. Detailed Prompt to Evaluation

When employing the LLMs as our evaluation tool, we utilize the following prompt structure. Within this structure, each evaluation index is meticulously detailed, providing a comprehensive understanding of its significance and rationale for inclusion.

> You as a linguist, evaluating whether a pun is good or bad doesn't have a fixed criterion, as its appreciation and evaluation are influenced by cultural nuances, background knowledge, personal tastes, and context. Nonetheless, when judging a pun, one can consider the following facets:
>
> Creativity and Humor: Good puns typically exhibit creativity, cleverly connecting two or more meanings in a way that evokes laughter or thoughtful reflection. Humor is a key criterion, determining whether the pun can bring about amusement.
>
> Clarity: A pun should be understandable without being overly obscure or hard to grasp. If people need too much time to understand it, it may not be considered a good pun.
>
> Originality: Good puns should be original rather than repetitive clichés or tired tropes. Originality can increase the appeal of a pun.
>
> Context Sensitivity: The effectiveness of a pun is often context-dependent. It should align well with the specific context to have the greatest impact. A good pun should fit seamlessly into its surroundings.
>
> Multiple Layers of Meaning: Some puns may have multiple layers of meaning, capable of eliciting different interpretations or reactions in various contexts. This richness can enhance their appeal.
>
> Please assess the subsequent sentences using the aforementioned criteria. Rate each on a scale of 0-10. Then, based on your judgment, compute a weighted average score for each sentence:
> {Sentence}

Figure 4: The above depicts the prompt input for GPT-4, Where {Sentence} serve as placeholders, which will be provided during actual pun sentence list.

B. Further Experiments

The table 4 provides a comprehensive breakdown of the indicators referenced in the main text. In the primary section, only average values are presented to conserve space. In this detailed table, we furnish scores for each specific metric: Creativity and Humor (CH), Clarity (CI), Originality (O), Context Sensitivity (CS), and Multiple Layers of Meaning (MLM).

The detailed comparison involves experimental results against showcases from **AMBIPUN**(Mittal et al., 2022), **Pun-GAN**(Luo et al., 2019), **LCR**(Yu et al., 2020), and **SurGen**(He et al., 2019), displayed sequentially from top to bottom.

These showcases are considered to represent the most focuses outcomes of their respective papers. Our evaluation against these results is conducted through the lens of LLMs, which provides scores over 10 iterations, with the average of these iterations taken as the final score. To mitigate any potential bias stemming from the order of presentation, the sequence of sentences is randomized in each iteration. This method ensures a fair and unbiased comparison, allowing us to accurately gauge our model's performance in relation to significant existing works in pun generation.

From the results, it is evident that our model essentially reaches human-level performance, including some high-quality puns that stand out for their creativity. For instance, the pun "**Waiting for her to die, I felt heavy.**" cleverly merges the homophones "wait" and "weight" to convey a profound experience of emotional and psychological burden. This burden stems from the anticipation of losing someone, and the sentence vividly expresses the "weight" of emotion in an unexpected manner. Such deep emotional expressions provide a crucial focal point for our future studies integrating sentiment analysis.

Another example is "**When the salt was touchy, his character flaked out.**" This showcases the model's ability to apply human characteristics to inanimate objects, creating humor and unexpectedness through linguistic creativity: By applying words associated with human emotions and behaviors ("touchy" and "flaked out") to describe the nature of "salt," the sentence creates an amusing visual and conceptual image: salt "flaking out" due to its "emotional" character. This pun humorously endows salt with human traits, achieving a humorous and unexpected effect.

Pun Word	Irritional	Score (GPT-4 and Claude-2)					
Sence 1	Real but not expressible as the quotient of two integers.	C&H	CI	O	CS	MLM	Avg.
Sence 2	Not consistent with or using reason.						
Ext AMBIPUN	I have an irrational paranoia about mathematical integers.	8|7	9|8	8.5|6	9|7	8|6	8.5|6.8
Gen AMBIPUN	My calculator is unjust and illogic. It's irrational.	8|5	8.5|7	8.5|5	8.5|5	8|6	8.3|5.6
Human	Old math teachers never die, they just become irrational.	8.5|5	9|8	8|4	9|6	8|4	8.5|5.4
PI (Ours)	Sometimes real problems are just too big to be squared away.	9|8	9|9	8.5|7	9|8	8.5|7	8.8|7.8
	"Please take me out of the equation with the square root" asked Tom rational.	9|9	9|8	9|9	9|10	9|9	9|9

Pun Word	Drive	Score (GPT-4 and Claude-2)					
Sence 1	A journey in a vehicle (usually an automobile)	C&H	CI	O	CS	MLM	Avg.
Sence 2	The trait of being highly motivated						
Ext AMBIPUN	What do you call a genius with cunning drive? racecar driver.	8|7	9|8	8|6	9|5	7|4	8.2|6
Gen AMBIPUN	I have the determination to travel to my destination. But i don't have the drive.	7.5|4	9|7	8|5	8|8	8|6	8.1|6
Human	A boy saving up for a car has a lot of drive.	8|5	9|9	7.5|4	8.5|7	8|5	8.2|6
PI (Ours)	Driving a car doesn't drive me crazy.	8|5	9|8	7.5|4	9|6	8|5	8.3|5.6
	When it was driving instructor Dan Carrel's turn to drive, his motivation was clear.	8.5|6	8.5|7	9|5	9|8	8|6	8.6|6.4

Pun Word	State	Score (GPT-4 and Claude-2)					
Sence 1	An organized political community forming part of a country.	C&H	CI	O	CS	MLM	Avg.
Sence 2	Mode or condition of being.						
Pun-GAN	In the state, the national assembly was established.	6|3	9|7	6.5|5	8|6	6|3	7.1|4.8
Human	Many people need to learn to be happy with the state they are in.	8|8	9|8	8|7	7.5|9	9|5	8.3|7.4
PI (Ours)	Politics - state your case.	7.5|8	8.5|9	7.5|8	9|8	7.5|5	8.0|7.6
	You state that you are stately, but I am not a man of state.	8|8	8.5|6	8.5|9	8|5	9|4	8.4|6.4

Pun Word	Torch	Score (GPT-4 and Claude-2)					
Sence 1	The event of something coming in contact with the body.	C&H	CI	O	CS	MLM	Avg.
Sence 2	A suggestion of some quality.						
Pun-GAN	It is a touch with the red sox.	5|4	6|7	7|3	6|5	6|2	6|4.2
Human	The massage which came with the spa treatment was a nice touch.	7.5|6	9|8	7.5|5	8.5|7	8|3	8.1|5.8
PI (Ours)	Potion: A Touch of Color.	7|7	8.5|6	8|7	8.5|6	7.5|4	7.9|6
	When the salt was touchy, his character flaked out.	8.5|8	8|5	9|8	8|6	8.5|7	8.4|6.8

Pun pair	borad-bored	Score (GPT-4 and Claude-2)					
		C&H	CI	O	CS	MLM	Avg.
LCR	your bedding is board.	4|5	6|7	6|5	7|5	5|3	5.6|5
Human	do hotel managers get board with their jobs?	8|7	8|8	7.5|7	8|8	7|5	7.7|7.0
PI (Ours)	For the board game, he had nothing.	7|8	8|9	7|8	8|7	6.5|6	7.3|7.6
	My wife and I don't have much of a board game habit.	7|6	9|8	6.5|6	8|7	6|4	7.3|6.2

Pun pair	rune-ruin	Score (GPT-4 and Claude-2)					
		C&H	CI	O	CS	MLM	Avg.
LCR	there was nothing but desolation and rune.	5|5	7|7	6|4	7|5	6|3	6.2|4.8
Human	the study of ancient symbols will lead you to rune.	6|6	9|8	7|6	8|7	6|4	7.2|6.2
PI (Ours)	The rune shack was ransacked.	7|4	8|9	7|3	7|8	6.5|2	7.1|5.2
	My horse must have run out of runes.	7|7	7|6	7|8	6|5	6.5|6	6.7|6.4

Pun pair	butter – better	Score (GPT-4 and Claude-2)					
		C&H	CI	O	CS	MLM	Avg.
SURGEN	Well, gourmet did it, he thought, it'd butter be right.	7|4	8|7	7|3	7|4	7|3	7.2|4.2
Human	Why did the dairy churn? The less said, the butter...	8|5	8|8	8|5	8|6	7|4	7.8|5.6
PI (Ours)	"It's butter better," Tom Butterer once said.	7|6	7|9	7|7	7|5	7|5	7|6.4
	"It's butter," said Tom butterfully.	8|6	8|8	8|6	8|7	7|4	7.8|6.2

Pun pair	peace – piece	Score (GPT-4 and Claude-2)					
		C&H	CI	O	CS	MLM	Avg.
SURGEN	That's because negotiator got my car back to me in one peace.	8|3	8|6	7|5	7|6	7|4	7.4|4.8
Human	Life is a puzzle; look here for the missing peace.	9|6	9|8	8|7	8|8	8|6	8.4|7
PI (Ours)	When you need to speak with peace, you should always make a point.	7|9	7|8	7|9	7|8	7|7	7|8.2
	You have to make peace with music.	7|7	8|8	8|6	7|7	7|5	7.4|6.6

Pun pair	flour – flower	Score (GPT-4 and Claude-2)					
		C&H	CI	O	CS	MLM	Avg.
SURGEN	Butter want to know who these two girls are, the new members of the holy flour.	7|5	7|8	8|5	7|7	7|4	7.2|5.8
Human	Betty crocker was a flour child.	9|5	9|6	9|5	8|7	8|6	8.6|5.8
PI (Ours)	When the flour blooms she's all over the map.	8|7	8|9	8|8	7|6	7|5	7.6|7
	The flour and flower were all in bloom but not ready for the rain.	8|5	9|9	7|5	7|8	8|6	7.8|6.6

Pun pair	wait – weight	Score (GPT-4 and Claude-2)					
		C&H	CI	O	CS	MLM	Avg.
SURGEN	Even from the outside, I could tell that he'd already lost some wait.	8|6	8|7	7|5	7|5	7|4	7.4|5.4
Human	Patience is a virtue heavy in wait.	9|7	9|8	8|6	8|6	8|5	8.4|6.4
PI (Ours)	Waiting for Bill, I wait only for the weight of my own mind.	8|8	8|7	8|8	8|7	9|7	8.2|7.4
	Waiting for her to die, I felt heavy.	6|5	9|8	7|4	7|6	8|3	7.4|5.2

Table 4: Example outputs of different Show-Case. Including scoring using LLMs