Punish the Pun-ish: Enhancing Text-to-Pun Generation with Synthetic Data from Supervised Fine-Tuned Models

Tomohito Minami, Ryohei Orihara[a], Yasuyuki Tahara[b], Akihiko Ohsuga[c] and Yuichi Sei[d]
The University of Electro-Communications, Chofu, Japan
minami.tomohito@ohsuga.lab.uec.ac.jp, orihara@acm.org, {tahara, ohsuga, seiuny}@uec.ac.jp

Keywords: Natural Language Processing, Natural Language Generation, Language Models, Preference Learning, Humor Generation, Pun Generation.

Abstract: Puns are clever wordplays that exploit sound similarities while contrasting different meanings. Such complex puns remain challenging to create, even with today's advanced large language models. This study focuses on generating Japanese *juxtaposed puns* while preserving the original meaning of input sentences. We propose a novel approach, applying Direct Preference Optimization (DPO) after supervised fine-tuning (SFT) of a pre-trained language model, utilizing synthetic data generated from the SFT model to refine pun generation. Experimental results indicate that our approach yields a marked improvement, evaluated using neural network-based and rule-based metrics designed to measure *pun-ness*, with a 2.3-point increase and a 7.9-point increase, respectively, over the baseline SFT model. These findings suggest that integrating SFT with DPO enhances the model's ability to capture phonetic nuances essential for generating juxtaposed puns.

1 INTRODUCTION

In recent years, natural language generation has made remarkable advances. Large Language Models (LLMs) have achieved significant results in various text generation tasks, such as translation, summarization, and code generation (Zhao et al., 2023). Despite the advances, generating humor remains a challenging task. Although it is reported that ChatGPT (OpenAI, 2022) can produce humorous output, research by Jentzsch et al. shows that novel humor creation remains difficult, as much of LLM's humor relies on existing patterns (Jentzsch and Kersting, 2023).

One well-known form of humor is the pun, which typically involves wordplay based on phonetic similarities between words. For example, "*I scream* every time I see *ice cream*" and "I used to be a banker, but I lost *interest*" demonstrate this wordplay.

Previous research has proposed various approaches to pun generation, including database-driven methods (Araki, 2018) and training a Transformer on pun databases (Hatakeyama and Tokunaga, 2021). Researchers have also tried to align pun generation with human preferences through two-stage tuning using Direct Preference Optimization (DPO) (Chen et al., 2024). However, existing methods insufficiently consider the meaning of the generated puns, focusing mainly on creating puns that incorporate specific input words. For instance, when trying to generate a pun expressing "It won't snow tomorrow," one might expect an output like "I gue*ss, no snow* tomorrow." Yet, generating such meaning-preserved puns remains challenging in previous studies.

In this study, we focus on Japanese *juxtaposed puns*, which present two phonemically similar and semantically different sequences of phonemes that appear together (Yatsu and Araki, 2018). For instance, "*I scream* every time I see *ice cream*" qualifies as a juxtaposed pun since "I scream" and "ice cream" share phonemic similarity and carry different meanings. This form of wordplay is independent of cultural background knowledge and can be evaluated based solely on the text, making it easily applicable across languages. In Japanese, the phrase "布団が吹っ飛んだ" (The futon blew away; *futon ga futton da*) is an example of a pun where the humor comes from the similar sounds in the words. Even people who don't understand Japanese can recognize it as a pun if they hear how it is pronounced.

Pun generation requires balancing phonetic and semantic constraints, making it difficult to produce

[a] https://orcid.org/0000-0002-9039-7704
[b] https://orcid.org/0000-0002-1939-4455
[c] https://orcid.org/0000-0001-6717-7028
[d] https://orcid.org/0000-0002-2552-6717

Minami, T., Orihara, R., Tahara, Y., Ohsuga, A. and Sei, Y.
Punish the Pun-ish: Enhancing Text-to-Pun Generation with Synthetic Data from Supervised Fine-tuned Models.
DOI: 10.5220/0013262900003890
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 1093-1100
ISBN: 978-989-758-737-5; ISSN: 2184-433X

high-quality puns. Furthermore, defining clear rules for evaluating puns is difficult because their humor relies on human subjectivity. To address the challenges, we propose a two-stage procedure. First, we perform *supervised fine-tuning* (SFT) on the LLM to learn basic pun generation skills. Next, we apply *Direct Preference Optimization* (DPO) (Rafailov et al., 2024) to align the model with human preferences by introducing *preference pairs* generated from the SFT model and ground truth puns.

This DPO process allows the model to align more closely with human preferences, enabling it to capture the essential features of puns more effectively. Experimental results show that our approach improves the *pun-ness* metrics, with a 2.3-point increase in neural network-based scores and a 7.9-point rise in rule-based scores over the baseline SFT model.

The key contributions of this study are: (i) Development of an LLM generating *juxtaposed puns* while preserving sentence meaning. (ii) Proposal of a framework for constructing pun-paraphrase pairs from a pun-only dataset. (iii) Demonstration that DPO-based learning with synthetic data improves pun quality over simple supervised fine-tuning.

2 RELATED WORKS

2.1 Pun Generation

Japanese Pun Database. Araki et al. constructed a Japanese pun database containing 67,000 entries collected from web sources (Araki et al., 2018; Araki et al., 2020). Japanese puns typically have two phonemically similar components: a *seed expression* and a *transformed expression*. A seed expression consists of one or more independent morphemes or phrases. In contrast, a transformed expression is a phoneme sequence located elsewhere in the sentence that sounds similar to the seed expression.

The database classifies puns into two categories: juxtaposed and superposed. **Juxtaposed puns**, exemplified by "*I scream* every time I see *ice cream*," explicitly contain both the seed expression (*I scream*) and the transformed expression (*ice cream*) within a sentence. **Superposed puns**, exemplified by "You've got to be *kitten meow*!", rely on the implicit seed expression (*kidding me*), which can be inferred from background knowledge or context.

Pun Generation via GAN and Reinforcement Learning. Luo et al. proposed Pun-GAN (Luo et al., 2019), a system that generates English puns using a Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and reinforcement learning. Pun-GAN's generator generates puns, and the discriminator determines whether the pun is machine-generated. For human-created puns, the discriminator also identifies the meaning of the pun word. Feedback is given as a reward based on two factors: machine-generation detection and the ambiguity of the seed expression's meaning. The generator learns to maximize the reward, and the discriminator learns to detect puns generated by the generator.

Pun Generation via Curriculum Learning. Chen et al. introduced a multi-stage curriculum learning framework to improve LLMs' pun-generation capabilities (Chen et al., 2024). Their method generates humorous sentences from word pairs containing a pun and an alternative word. Learning process through DPO-based optimization: first refining puns' structural features, then improving humor quality. This sequential two-stage approach improves each element progressively. Their evaluation of the method on Chinese and English datasets showed superior performance over existing models.

2.2 Preference Learning

Reinforcement Learning from Human Feedback. Ziegler et al. used reinforcement learning to fine-tune pre-trained language models based on human preferences (Ziegler et al., 2020). This method is known as Reinforcement Learning from Human Feedback (RLHF). In RLHF, a reward model is learned from human evaluations, guiding the language model to maximize this reward. The result is more natural text generation with consistent stylistic elements, including emotional expression and descriptive richness.

Direct Preference Optimization. Rafailov et al. introduced Direct Preference Optimization (DPO) as an alternative to RLHF. Unlike RLHF, DPO directly learns user preferences using classification loss from paired data, avoiding reinforcement learning and its complex training processes. This simplification enables stable and efficient learning. DPO has achieved performance comparable to or better than reinforcement learning methods across tasks such as sentiment adjustment, summarization, and dialogue generation, improving learning efficiency and task outcomes.

3 PROBLEM DEFINITION

Previous studies have focused on generating puns by incorporating specific input words. For example,

given *weak* and *week*, a model might generate the pun "I lift weights only on Saturday and Sunday because Monday to Friday are *weak* days." In contrast, this research tackles generating juxtaposed puns from arbitrary sentences, preserving their original meaning. For instance, given "Seeing ice cream makes me shout", a model generates "*I scream* every time I see *ice cream*." Unlike previous studies, this approach does not depend on explicitly provided pun words; instead, it leverages phonetic wordplay to align with the input's semantics.

This study particularly focuses on generating Japanese juxtaposed puns, which are popular in Japan and have plentiful data available. For example, given the input "布団が飛んでいきました" (futon ga tonde ikimashita; the futon flew away), the model generates the pun "布団が吹っ飛んだ" (*futon* ga *futton* da; the futon blew away).

4 METHOD

4.1 The Pun Paraphrase Dataset

In this study, we explore the task of generating Japanese juxtaposed puns from arbitrary Japanese sentences. To fine-tune a pre-trained language model through supervised learning, a dataset with paired puns and their paraphrased versions is required.

To create the dataset, which we call the *pun paraphrase dataset*, we generated pairs of puns and their paraphrased versions using the pun database (Araki et al., 2018), a resource composed solely of puns. This database contains juxtaposed puns, their structural information, and human evaluations.

4.1.1 Splitting the Pun Dataset

First, we have split the pun dataset into training, validation, and test sets. Random splitting could cause data leakage. In the context of puns, due to common set phrases and popular expressions, similar or identical puns often appear multiple times in the dataset. If such related puns appear across different splits, performance metrics might be inflated, reflecting memorization instead of generalization. To address this, we developed a specialized splitting method.

We have divided the puns in the database into groups. Let $d(S,T)$ denote the edit distance between S and T, and $E(S)$ represent the set of seed and transformed expressions of pun S. For any two juxtaposed puns S and T, we have grouped them if $d(S,T) \leq 4$ and there exist $s \in E(S), t \in E(T)$ such that $d(s,t) \leq 1$. This approach aggregates similar puns.

Next, we have assigned each group to the training, validation, or test set, ensuring that similar puns stayed within one set. This minimizes data leakage. We have split the dataset into training, validation, and test sets in a ratio of 90:5:5, resulting in 58,193 / 3,165 / 3,164 puns, respectively.

4.1.2 Paraphrasing

To paraphrase the puns, we have filtered the database for entries with an average annotator score of 2 or higher on a 5-point scale[1]. This resulted in 62,429 puns for paraphrasing. The entire dataset was also used for training neural network-based pun detection in Section 6.1.

For paraphrase, we have used GPT-4o mini (`gpt-4o-mini-2024-07-18`) (OpenAI, 2024b) and generated 10 paraphrases for each pun, specifying 10 styles: *standard*, *colloquial*, *formal*, *descriptive*, *poetic*, *concise*, *for children*, *exaggerated*, *negative*, and *positive*. The prompt is available online[2] (Prompt A).

This yielded 645,008 pairs of puns and paraphrases. We have selected pairs meeting both criteria:

- Cosine similarity between their text embedding vectors generated by `text-embedding-3-large` (OpenAI, 2024a) is 0.7 or higher.
- Normalized edit distance is 0.5 or higher

To prevent imbalance, we have randomly selected up to three paraphrases per pun from filtered results. This resulted in 172,167 pairs of puns and paraphrases, split into training, validation, and test sets containing 155,779 / 8,161 / 8,227 pairs, respectively.

4.2 Training a Pun Generation Model

We aim to develop a language model that transforms input sentences into juxtaposed puns while preserving their meanings. The model is trained through the following main steps:

1. Supervised fine-tuning (SFT) of a pre-trained language model using the pun paraphrase dataset to enable it to generate pun-style sentences. We call this model the SFT model.
2. Inference on the training and validation sets using the SFT model.
3. Further optimization with Direct Preference Optimization (DPO), using the SFT model outputs

[1]The scoring scale: 5 (very funny), 4 (funny), 3 (average), 2 (unfunny), 1 (very unfunny or not a pun) (Araki et al., 2018).

[2]The prompts used in this study can all be found here: https://link.trpfrog.net/pun-ish

labeled as *dispreferred* and ground truth puns labeled as *preferred*, to enhance generation quality.

While SFT enables language models to generate pun-style text, ensuring high-quality outputs remains challenging due to the difficulty of creating a complex pun structure while preserving meaning. Therefore, we apply DPO using paired SFT outputs and ground truth puns to enhance the model's understanding of pun structure and content.

4.2.1 Fine-Tuning with Pun Paraphrase Dataset

For supervised fine-tuning in pun generation, we use the pun paraphrase dataset created in Section 4.1. By training on pun-paraphrase pairs, the model learns to transform input sentences into puns while maintaining their meaning. The specific prompt and response templates for this training are available online (Prompt B).

4.2.2 Inference with the SFT Model

We apply the SFT model to generate puns from the pun paraphrase dataset's training and validation sets. The generated puns are combined with their corresponding input paraphrases and ground truth puns, forming a triplet. We call this the *pun preference dataset*, used for subsequent preference training.

4.2.3 Preference Training

While SFT enables basic pun-style text generation, achieving high-quality outputs requires additional optimization. Effective pun generation demands not only stylistic transformation, such as replacing the original expressions with conversational ones frequently found in Japanese puns, but also meaning preservation and clever wordplay that reflects the characteristics of puns. We address this challenge through Direct Preference Optimization (DPO).

DPO improves model performance by maximizing the probability gap between *preferred* and *dispreferred* outputs. However, focusing solely on this probability gap can decrease the absolute probability of preferred outputs during training. For example, even if the probability of both the preferred and dispreferred outputs decreases, the DPO loss is minimized as long as the probability gap increases. To address this limitation, we used the APO-zero loss function from Anchored Preference Optimization (APO) (D'Oosterlinck et al., 2024). APO-zero is designed to increase the probability of the preferred output while decreasing the probability of the dispreferred output. It is particularly effective when ground truth data quality exceeds model output quality. In our case, it is effective because the ground truth puns have higher quality than the SFT model-generated puns.

By using DPO with APO-zero, we enhance the model's pun generation capability beyond basic SFT. In this stage, we use the pun preference dataset generated in Section 4.2.2. We label the outputs from the SFT model as dispreferred texts and the original puns from the pun database as preferred texts. This comparison between model-generated puns and human-created ones enables the model to learn subtle nuances, helping it generate high-quality puns with a nuanced understanding of pun structures, rather than simply replicating stylistic elements.

5 EXPERIMENT

Supervised Fine-Tuning. In our experiments, we first fine-tuned the Japanese version of the Gemma 2 2B model (`google/gemma2-2b-jpn-it`) on the pun paraphrase dataset (Section 4.1) for five epochs using LoRA (Hu et al., 2022) with rank $r = 16$, $\alpha = 64$, and dropout $= 0.1$. The batch size was eight, the learning rate was 3×10^{-5} with AdamW, and a cosine annealing scheduler with a 10% warmup ratio was applied. The model from the second epoch, achieving the lowest validation loss, was selected. For inference to create the pun preference dataset, puns were generated with a temperature of 0.8 and top-$p = 0.8$.

Fine-Tuning with DPO. Building on the supervised fine-tuned model, we trained it with DPO for two epochs using LoRA with rank $r = 16$, $\alpha = 64$, and dropout $= 0.1$. The batch size was two, with gradient accumulation steps set to four. The maximum learning rate was 1×10^{-7} with AdamW, and a cosine annealing scheduler was applied, with the first 10% of steps used for warmup. For DPO, we used APO-zero as the loss type, with a beta value of 0.5.

Text Generation. We used top-p sampling ($p = 0.8$) with a temperature of 0.8 to generate puns. For comparison, GPT-4o (`gpt-4o-2024-11-20`) (OpenAI, 2024c) was employed under identical settings.

6 EVALUATION

This study automatically evaluated the generated puns using several metrics.

Edit Distance. Edit distance, a metric for measuring string similarity, was used to evaluate the similar-

ity between generated and reference puns. We also calculated it for the Romanized forms of the puns, where Japanese text is transliterated into Latin characters (e.g., "ありがとう" (thank you) becomes "*arigatou*"). Since Romanization reflects the phonetic features of Japanese, this evaluates phonetic similarity.

Semantic Similarity. To evaluate the semantic similarity between the input sentence and generated pun, we used OpenAI's `text-embedding-3-large` embedding model (OpenAI, 2024a). Cosine similarity between the embeddings of the input and the generated pun served as the semantic similarity metric.

Pun-ness. We evaluated the pun-ness of the generated text using a neural network-based pun detector, a rule-based pun detector, and a pun DB-based metric. Details are provided in Section 6.1.

LLM-as-a-Judge. To evaluate the quality of generated puns comprehensively, we used LLM as an automated evaluator, comparing our model to baselines on semantic similarity, pun quality, humor, and overall quality. Details are provided in Section 6.2.

6.1 Pun-ness

Neural Network-Based Pun Detection. We used the Convolutional Pun Detection Network (CPDN) (Minami et al., 2023) to detect juxtaposed puns. Juxtaposed puns feature words with similar phonemes. CPDN detects puns by leveraging this characteristic. Table 1 shows the results of the pun classifier. We used the percentage of texts identified as puns by CPDN as the neural network-based pun-ness.

Rule-Based Pun Detection. We used DaaS, a rule-based pun detector (Ritsumeikan University Dajare Club, 2020). DaaS processes text into readings and morphemes, checking for puns via overlapping morphology, exact character matches, or phonetic similarities. We used the percentage of texts identified as puns by DaaS as the rule-based pun-ness.

Pun DB-Based Metrics. We also evaluated pun-ness using the minimum normalized edit distance between the Romanized form of a generated pun and puns in the pun database. Let t_r be the Romanization of text t. The pun DB-based pun-ness in the pun database $\mathcal{D}$ is defined as follows.

$$\text{Pun-ness}_{\mathcal{D}}(t) = \min_{d \in \mathcal{D}} \frac{\text{EditDistance}(t_r, d_r)}{\max(|t_r|, |d_r|)} \quad (1)$$

Table 1: Classification performance of the CPDN and DaaS pun classifier.

	Precision	Recall	F1
CPDN (NN-based)	0.927	0.898	0.913
DaaS (Rule-based)	0.761	0.812	0.786

6.2 LLM-as-a-Judge Evaluation

We employed an automated evaluation method using large language models (LLMs), based on a previous study showing that strong LLMs can approximate human judgments with high agreement rates (Zheng et al., 2024). We used OpenAI's GPT-4o (`gpt-4o-2024-11-20`) as the evaluator with a temperature parameter of 0 to ensure consistency.

We compared outputs from four sources: our proposed model, the base model, the SFT model, and ground truth puns, through pairwise comparisons. Each evaluation involved a randomly sampled tuple (Input, Output A, Output B) from the test set, with 500 tuples per model pair. To avoid *self-enhancement bias*, we excluded GPT-4o outputs. To minimize *position bias*, we alternated output order and recorded *draw* in case of conflicts.

Outputs were evaluated on four criteria: semantic similarity to the input, quality as a juxtaposed pun, humor, and overall quality. The evaluator selected the better output or considered both acceptable. The evaluation prompts are available online (Prompt C).

7 RESULTS AND DISCUSSION

7.1 Quantitative Evaluation

Table 2 shows the results of the quantitative evaluation. Fine-tuning the Japanese Gemma 2B model with our pun paraphrase dataset outperformed GPT-4o across most metrics, except for semantic similarity. It indicates that our dataset, designed for generating puns, helped the model learn pun-ness effectively. In contrast, GPT-4o emphasized semantic similarity over pun-ness, resulting in higher semantic similarity scores. High semantic similarity can limit the flexibility required for puns, as seen in GPT-4o's results. The results suggest that GPT-4o shows lower pun-ness, while our model achieves higher pun-ness with lower semantic similarity than GPT-4o.

Additional training with DPO enables our model to surpass baseline scores on all three pun-ness metrics. We attribute this to DPO's preference learning approach, which captured pun characteristics that supervised fine-tuning alone was unable to capture.

Table 2: Quantitative evaluation results for pun generation. **Bold** indicates the best value, underline the second-best, excluding Input and Ground Truth rows. *Semantic* denotes the semantic similarity between the input sentence and the generated pun. *NN*, *Rule*, and *DB* denote the neural network-based, rule-based, and pun DB-based pun-ness metrics, respectively.

	Edit Distance (↓)		Semantic (↑)	Pun-ness		
	Original	Romanized		NN (↑)	Rule (↑)	DB (↓)
GPT-4o (`gpt-4o-2024-11-20`) @1-shot	0.702	0.610	**0.863**	0.273	0.197	0.562
Base model @1-shot	0.814	0.700	0.664	0.098	0.072	0.576
SFT only @0-shot	**0.575**	**0.465**	0.813	0.753	0.500	0.401
Ours (SFT + DPO) @0-shot	0.604	0.473	0.738	**0.776**	**0.579**	**0.394**
Input (Test set)	0.626	0.545	1.000	0.349	0.209	0.507
Ground Truth (Test set)	0.000	0.000	0.803	0.905	0.826	0.000

Table 3: Comparison results of generated text, judged by GPT-4o. *Win* represents better performance, *Draw* equal performance, and *Lose* worse performance of our method. **Bold** indicates the best value, underline the second-best.

Metric	Ours vs	Win	Draw	Lose
Semantic Similarity	Base Model	0.312	0.184	**0.504**
	SFT only	0.106	0.286	**0.608**
	Ground Truth	0.044	0.208	**0.748**
Juxtaposed Pun Quality	Base Model	**0.898**	0.072	0.030
	SFT only	**0.654**	0.258	0.088
	Ground Truth	0.318	0.264	**0.418**
Humor	Base Model	**0.676**	0.154	0.170
	SFT only	**0.662**	0.220	0.118
	Ground Truth	0.338	0.200	**0.462**
Overall Quality	Base Model	**0.632**	0.218	0.150
	SFT only	**0.652**	0.212	0.136
	Ground Truth	0.274	0.220	**0.506**

Our model's outputs showed lower semantic similarity to the input than the SFT model's outputs. This may be attributed to the preference dataset, where ground truth puns exhibited lower semantic similarity than the SFT model's outputs. As a result, the model guided by DPO learned to favor lower semantic similarity, prioritizing pun characteristics.

7.2 LLM-as-a-Judge Evaluation

Table 3 shows the results of the LLM-as-a-Judge evaluation. In the LLM-as-a-Judge evaluation, our proposed method outperformed baselines in juxtaposed pun quality. While quantitative metrics indicate slight improvement over the baseline SFT model, LLM-as-a-Judge evaluations suggest that our method's puns exhibit markedly more pun-ness given the same input. It shows the effectiveness of our approach.

Furthermore, the proposed method greatly exceeded the baselines in humor and overall quality metrics. This suggests that preference learning through DPO helped capture not only structural elements of pun-ness but also nuances of humor. This result is noteworthy because it not only applies to pun generation but also makes a meaningful contribution to the broader field of humor generation.

Although pun generation remains challenging for LLMs, we used an LLM to evaluate puns. When comparing our model's outputs to human-created ground truth, we observed that *Lose* received the highest evaluation rate. This suggests that GPT-4o can effectively evaluate puns, supporting its use as a metric. While further investigation is required, LLM-based pun evaluation appears to be a viable option.

7.3 Observations from Examples

Table 4 shows examples of generated puns. The example "モヤシ食べちゃった、もうやぁ失点! (*Moyashi* tabechatta, *mou yaa shi*tten!; I ate the bean sprouts, oh no, a point lost!)". This example highlights the model's ability to generate unique puns by leveraging phonetic similarities, such as between "*moyashi*" (bean sprouts) and "*mou yaa shitten*" (oh no, a point lost). Notably, "*mou yaa shitten*" is an uncommon colloquial phrase in Japanese, intentionally chosen here to align with "*moyashi*" and enhance the pun's effect.

While effective, adding "*mata*" (また, again) to "*moyashi tabechatta*" (I ate the bean sprouts) to form "*Mata moyashi tabechatta*" (またモヤシ食べちゃった, I ate the bean sprouts again) could improve semantic similarity without reducing pun quality. This suggests the model has yet to fully balance pun-ness and semantic similarity.

Another example shows that our model produced a better pun structure than the ground truth.

	Neta	dukuri	*ya*	wa	*neta*	shokunin	*ya*	
Ours:	ネタ	作り	屋	は	寝た	職人	や	!
	content or ideas	make	specialist	is	slept	craftsman	a Kansai dialect ending	

	Neta	shokunin	wa	*neta*	shokunin	*desu*	
Ground Truth:	ネタ	職人	は	寝た	職人	です	!
	content or ideas	craftsman	is	slept	craftsman	polite ending	

This output of our model uses "Neta....ya," as a seed expression. In contrast, the ground truth relies only on "Neta."[3] Generally, creating puns with long

[3]"Shokunin" does not meet the criterion of possessing

Table 4: Examples of generated puns. Romanized expressions and English translations were manually annotated. Parentheses ( ) and square brackets [ ] in Romanized expressions indicate the seed and transformed expressions. *Pun?* indicates if the text is a pun, and *Sim.?* indicates if it functions as a paraphrase of the original. ✓ denotes a positive result. Numbers in parentheses in *Pun?* show the phoneme count of the seed (greater is better). △ in *Sim.?* indicates unclear or uncertain similarity. *GT* denotes the ground truth. Note that *Pun?* and *Sim.?* evaluations in this table are based on the authors' subjective judgment.

	Japanese (Model Output)	Romanized Expression	English Translation	Pun?	Sim.?
Input	もう一度モヤシを食べて、また大失敗してしまった！	Mou ichido moyashi o tabete, mata daishippai shite shimatta!	I ate bean sprouts once again and made a big mistake again!		N/A
GPT-4o	モヤシをもう一度食べたら、もやもやっと大失敗しちゃった！	(Moya)shi o mou ichido tabetara, [moya] moya tto dai shippai shichatta!	When I ate bean sprouts once more, I ended up making a big mistake in a *moyamoya* way! [*1]	✓(4)	✓
Base Model	もう一回、モヤシを食べて、その失敗を大いに笑って過ごそう！	Mou ikkai, moyashi o tabete, sono shippai o ooi ni waratte sugosou!	Let's eat bean sprouts once more and spend our time greatly laughing at that failure!		△
SFT Model	モヤシをまた食べて、もやっしい!	(Moyashi) wo mata tabete, [moyashii]!	I ate bean sprouts again, it's *moyaccy*! [*2]	✓(7)	
Ours	モヤシ食べちゃった、もうやぁ失点!	(Moyashi) tabechatta, [mou yaa shi]tten!	I ate bean sprouts, oh no, a point lost!	✓(7)	△
GT	また、モヤシ食って、またもや、しくった。	(Mata, moyashi ku)tte, [mata moya, shiku]tta.	I ate bean sprouts again, and once more, I failed.	✓(13)	✓
Input	ムスカ大佐のムースは、まるで石のように固い。	Musuka taisa no muusu wa, marude ishi no you ni katai.	Colonel Muska's mousse is as hard as a stone.		N/A
GPT-4o	ムスカ大佐のムースか？それはもう、石のムースかもね！	(Musuka) taisa no muusu ka? Sore wa mou, ishi no [muusu ka]mo ne!	Colonel Muska's mousse? It might be a stone mousse!	✓(6)	△
Base Model	ムスカ大佐のムースは、まるで石のように堅い！	Musuka taisa no muusu wa, marude ishi no you ni katai!	Colonel Muska's mousse is as hard as a stone!		✓
SFT Model	ムスカ大佐のムース固め	(Musuka) taisa no [muusu ka]tame	Colonel Muska's mousse hardening.	✓(6)	✓
Ours	ムスカ大佐のムースか	(Musuka) taisa no [muusu ka]	Colonel Muska's mousse?	✓(6)	△
GT	ムスカ大佐のムース硬いさ。	(Musuka taisa) no [muusu katai sa].	Colonel Muska's mousse is hard.	✓(11)	✓
Input	ネタを作る職人は、実は寝ている職人だ。	Neta o tsukuru shokunin wa, jitsu wa nete iru shokunin da.	A craftsman who creates ideas is, in fact, a craftsman who sleeps.		N/A
GPT-4o	ネタを作る職人は、実は「寝た」を作る職人だ。	(Neta) o tsukuru shokunin wa, jitsu wa [neta] o tsukuru shokunin da.	A craftsman who creates ideas is, in reality, a craftsman who creates "sleep."	✓(4)	△
Base Model	了解！	Ryoukai!	Understood!		
SFT Model	ネタ作りが得意な人は、寝た職人	(Neta) dukuri ga tokui na hito wa, [neta] shokunin	Those skilled at creating ideas are sleeping craftsmen.	✓(4)	△
Ours	ネタ作り屋は寝た職人や!	(Neta) dukuri(ya) wa [neta] shokunin [ya]!	An idea-maker is a sleeping craftsman!	✓(6)	✓
GT	ネタ職人は、寝た職人です	(Neta) shokunin wa, [neta] shokunin desu	An idea-maker is a sleeping craftsman!	✓(4)	✓

*__Note 1:__ In Japan, *moyamoya* is an onomatopoeic term for a vague feeling of unease or frustration. __Note 2:__ *moyashii* is not a valid word.

seed and transformed expressions is more challenging than with shorter ones. This example suggests that our approach could help future language models create more complex puns, possibly even more sophisticated than humans.

8 CONCLUSION

In this study, we proposed a method for generating juxtaposed puns by applying Direct Preference Optimization (DPO) following supervised fine-tuning (SFT). We constructed a pun paraphrase dataset from an existing pun database, enabling the model to learn pun characteristics more effectively. Using this dataset, our fine-tuned Japanese Gemma 2B model outperformed GPT-4o in pun-ness metrics, even without applying DPO. The subsequent DPO training, which used both the SFT model's outputs and ground truth, enhanced pun quality. Specifically, we observed a 2.3-point increase in the neural network-based pun-ness score and a 7.9-point increase in the rule-based score compared to the baseline SFT model. Evaluations conducted by GPT-4o also confirmed that our proposed method outperformed other approaches in humor and overall pun quality.

Future research could refine the pun paraphrase dataset and leverage multi-stage training to better balance pun-ness and semantic similarity. Methods such as reinforcement learning, which treat pun-ness and semantic similarity as dual objectives, have the potential to enhance performance systematically. However, multi-objective optimization is a challenging problem, including in the context of reinforcement learning, requiring continuous and iterative efforts to achieve a reasonable balance. Conducting comprehensive human evaluations is likely to validate humor quality further and reveal nuanced linguistic aspects. Extending this framework to diverse linguistic and cultural contexts is critical for assessing its generalizability beyond Japanese. Additionally, our method's simplicity suggests applicability to other forms of wordplay, from varied humor to poetic expressions. A broader investigation into these areas could reveal how much this approach generalizes beyond puns, contributing to a deeper understanding of the creativity of LLMs.

a distinct meaning. Therefore, it cannot be classified as a seed and transformed expression in this context.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Numbers JP22K12157, JP23K28377, JP24H00714.

This work was conducted using the computers at the Artificial Intelligence eXploration Research Center (AIX) at the University of Electro-Communications.

We acknowledge the assistance for the GPT-4o (OpenAI, 2024c), OpenAI o1-preview (OpenAI, 2024e), OpenAI o1 (OpenAI, 2024d) and Anthropic Claude 3.5 Sonnet (Anthropic, 2024) were used for proofreading, which was further reviewed and revised by the authors.

The pun database used in this work was developed under JSPS KAKENHI Grant-in-Aid for Scientific Research (C) Grant Number 17K00294. We would like to express our gratitude to Professor Kenji Araki of the Language Media Laboratory, Division of Media and Network Technologies, Faculty of Information Science and Technology, Hokkaido University, for providing the pun database.

REFERENCES

Anthropic (2024). Introducing claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. (Accessed on 11/07/2024).

Araki, K. (2018). Performance evaluation of pun generation system using pun database in japanese. Proceedings SIG-LSE-B703-8, The Japanese Society for Artificial Intelligence, 2nd Workshops (in Japanese).

Araki, K., Sayama, K., Uchida, Y., and Yatsu, M. (2018). Expansion and analysis of a fashionable database. JSAI Type 2 Study Group Language Engineering Study Group Material (SIG-LSE-B803-1) (in Japanese), pages 1–15.

Araki, K., Uchida, Y., Sayama, K., and Tazu, M. (2020). Pun database. http://arakilab.media.eng.hokudai.ac.jp/~araki/dajare_eng.htm. (Accessed on 11/05/2024).

Chen, Y., Yang, C., Hu, T., Chen, X., Lan, M., Cai, L., Zhuang, X., Lin, X., Lu, X., and Zhou, A. (2024). Are U a joke master? pun generation via multi-stage curriculum learning towards a humor LLM. In *Findings of the ACL 2024*.

D'Oosterlinck, K., Xu, W., Develder, C., Demeester, T., Singh, A., Potts, C., Kiela, D., and Mehri, S. (2024). Anchored preference optimization and contrastive revisions: Addressing underspecification in alignment. https://arxiv.org/abs/2408.06266.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3.

Hatakeyama, K. and Tokunaga, T. (2021). Automatic generation of japanese juxtaposition puns using transformer models. *27th Annual Meeting of the Language Processing Society of Japan (in Japanese)*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *ICLR*.

Jentzsch, S. and Kersting, K. (2023). ChatGPT is fun, but it is not funny! humor is still challenging large language models. In *Proceedings WASSA*, pages 325–340.

Luo, F., Li, S., Yang, P., Li, L., Chang, B., Sui, Z., and Sun, X. (2019). Pun-gan: Generative adversarial network for pun generation. In *Proceedings of the 2019 Conference on EMNLP-IJCNLP*, pages 3388–3393.

Minami, T., Sei, Y., Tahara, Y., and Osuga, A. (2023). An investigation of pun generation models using paraphrasing with deep reinforcement learning. *15th Forum on Data Engineering and Information Management (DBSJ 21th Annual Meeting) (in Japanese)*.

OpenAI (2022). ChatGPT: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/. (Accessed on 02/05/2024).

OpenAI (2024a). Embeddings - OpenAI API. https://platform.openai.com/docs/guides/embeddings. (Accessed on 02/04/2024).

OpenAI (2024b). GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. (Accessed on 08/25/2024).

OpenAI (2024c). Hello GPT-4o. https://openai.com/index/hello-gpt-4o/. (Accessed on 11/07/2024).

OpenAI (2024d). Introducing OpenAI o1. https://openai.com/o1/. (Accessed on 01/04/2025).

OpenAI (2024e). Introducing OpenAI o1-preview. https://openai.com/index/introducing-openai-o1-preview/. (Accessed on 11/07/2024).

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model. https://arxiv.org/abs/2305.18290.

Ritsumeikan University Dajare Club (2020). rits-dajare/daas: Dajare as a service（japanese pun detection / evaluation engine). https://github.com/rits-dajare/daas. (Accessed on 10/31/2024).

Yatsu, M. and Araki, K. (2018). Comparison of pun detection methods using Japanese pun corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Zhao, W. X. et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. (2024). Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on NeurIPS*.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. (2020). Fine-tuning language models from human preferences. https://arxiv.org/abs/1909.08593.