"A good pun is its own *reword*": Can Large Language Models Understand Puns?

Zhijun Xu[♠], **Siyu Yuan**[♠], **Lingjie Chen**[♠], **Deqing Yang**[♠*]
[♠]School of Data Science, Fudan University
{zjxu23,syyuan21,ljchen21}@m.fudan.edu.cn
yangdeqing@fudan.edu.cn

Abstract

As one of the common rhetorical devices, puns play a vital role in linguistic study, including the comprehensive analysis of linguistic humor. Although large language models (LLMs) have been widely explored on various tasks of natural language understanding and generation, their ability to understand puns has not been systematically studied, limiting the utilization of LLMs in creative writing and humor creation. In this paper, we leverage three popular tasks, *i.e.*, *pun recognition, pun explanation*, and *pun generation*, to systematically evaluate LLMs' capability of understanding puns. In addition to the evaluation metrics adopted by prior research, we introduce some new evaluation methods and metrics that are better suited to the in-context learning paradigm of LLMs. These new metrics offer a more rigorous assessment of an LLM's capability to understand puns and align more closely with human cognition. Our research findings reveal the "lazy pun generation" pattern and identify the primary challenges in understanding puns with LLMs. The code is available at https://github.com/Zhijun-Xu/PunEval.

1 Introduction

Pun, as a form of wordplay, cleverly exploits double or multiple meanings of words (Miller et al., 2017). For example, for a pun sentence, "A good pun is its own reword", it plays on the similar sounds of "reword" and "reward", suggesting that the intrinsic value or reward of a good pun lies in its clever use of language or its inventive rephrasing. In most cases, the use of puns can produce humorous effects, as it creates a lexical-semantic ambiguity (Kao et al., 2016) and a context-shift surprise (He et al., 2019). Compared to other forms of humor, such as jokes (Dynel, 2009) and comedies (Stott, 2014), puns are appropriate for linguistic humor study as they have a more precise definition and a relatively fixed structure (Hempelmann, 2008; Attardo, 2018).

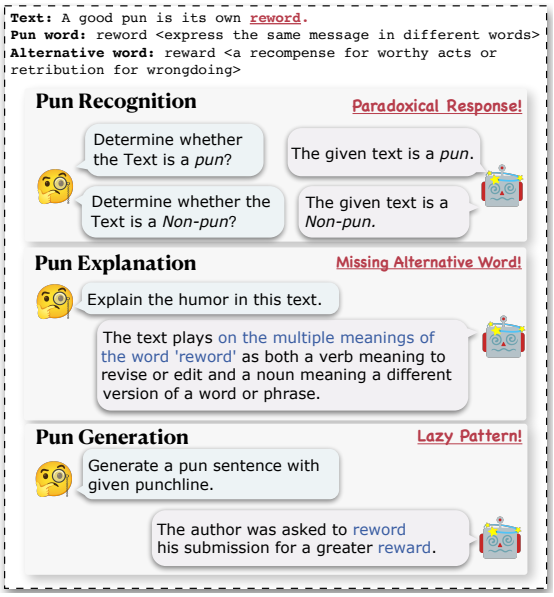

Figure 1: Toy examples of achieving three representative tasks related to pun understanding with LLMs, including pun recognition, explanation and generation. We explore the primary difficulties (*e.g.*, paradoxical response, missing alternative word and lazy pattern) in these tasks.

Previous research on pun exploration primarily concentrated on developing specific language models or complex frameworks to recognize (Zou and Lu, 2019; Zhou et al., 2020), explain (Sun et al., 2022a), or generate (Mittal et al., 2022; Tian et al., 2022) puns. With the advancement of large language models (LLMs), recent studies have explored using LLMs for detecting jokes (Gupta et al., 2021; Baranov et al., 2023) and identifying humor in images (Hessel et al., 2023) and videos (Ko et al., 2023). Exploring LLMs' comprehension of puns could further enhance their values on creative text creation and humor generation. Unfortunately, there are still no studies evaluating LLMs' capability of understanding puns systematically. There-

* Corresponding author.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 11766–11782
November 12-16, 2024

fore, in this paper, we aim to systematically evaluate the capabilities of LLMs on pun understanding. As illustrated in Figure 1, to provide comprehensive assessments, we focus on three tasks from previous work, *i.e.*, pun recognition, pun explanation, and pun generation. To adapt these tasks to the in-context learning (ICL) paradigm of LLMs, we develop some new methods and metrics to ensure rigorous evaluation. For pun recognition, we create dual-biased prompts to gauge the confidence level of LLMs' responses. These prompts explicitly incorporate the terms "pun" or "non-pun" to interfere with the model's judgment. For pun explanation, we employ both a fine-grained punchline check and a coarse-grained pairwise comparison. These methods help identify LLMs' shortcomings and assess the overall quality of LLMs' explanations. For pun generation, we introduce two novel settings, *i.e.* free and constrained generation, which demonstrate the LLMs' ability to create puns under varying conditions. Moreover, we introduce an *Overlap* metric to measure the originality of the puns generated by LLMs.

Our research has demonstrated that most LLMs are easily influenced by prompt bias in recognizing puns. They also struggle to explain puns which are based on phonetic similarities. In addition, we observe that LLMs often resort to a low-quality and incorrect pattern in pun generation, separating the double meanings instead of combining them. We term this pattern as "lazy pun generation". Despite all these issues, some powerful LLMs still exhibit impressive performance across the three tasks. Specifically, LLMs are competitive with humans in pun explanation and surpass the state-of-the-art models in pun generation. The main contributions of this paper are summarized as follows:

- To the best of our knowledge, our work is the first to systematically evaluate LLMs' capabilities of pun understanding.
- We propose several novel evaluation methods and metrics, including dual-biased prompted asking, punchline check, and overlap indicator for assessing the originality of pun generation. Compared to previous work, our evaluation methods and metrics better adapt to the ICL paradigm of LLMs.
- Through extensive experiments with various LLMs under different pun settings, we provide a detailed and in-depth analysis of the results. Our findings highlight the primary difficulties LLMs face in pun understanding and offer insights that could benefit future research in this area.

2 Related Work

Studies on Puns Puns, recognized as a significant linguistic art form, have garnered attention in AI research (Xiu et al., 2017; Doogan et al., 2017; Yu et al., 2018),. Previous work mainly collects various types of puns (Miller et al., 2017) from literature and the Internet and proposes diverse tasks to evaluate the pun understanding capabilities of LMs. These tasks can be divided into three categories: *1) pun recognition* (Diao et al., 2018; Zou and Lu, 2019; Zhou et al., 2020), which involves the detection of puns and localization of pun words. *2) pun explanation* (Sun et al., 2022a), which clarifies why the puns are funny by natural language explanations. *3) pun generation*, which mainly requests small LMs to either rewrite retrieved sentences into puns (He et al., 2019; Yu et al., 2020) or create puns more flexibly using acquired context words (Mittal et al., 2022; Tian et al., 2022; Sun et al., 2022b). For evaluation metrics, some work analyses pun from multiple quantifiable dimensions like ambiguity and distinctiveness (Kao et al., 2016), as well as surprise and unusualness (He et al., 2019). However, these studies mostly focus on training small models in pun tasks. Our research is the first to systematically evaluate the capabilities of LLMs to recognize, explain, and generate puns.

LLMs for Humors With vastly improved understanding and creativity, LLMs not only excel in traditional humor tasks such as detection and rating (Gupta et al., 2021; Baranov et al., 2023; Choi et al., 2023) but also demonstrate exciting potential in humor explanation and generation (Jentzsch and Kersting, 2023; Zhong et al., 2023). Some works aid LLMs in joke generation with humor algorithms (Toplyn, 2023) or feedback-driven techniques (Ravi et al., 2024), while others focus on comprehending and explaining punchlines in images (Hessel et al., 2023) or videos (Ko et al., 2023). Our work is the first to focus on pun understanding, a vital part of the humor.

3 Preliminaries

In this paper, we focus on two primary types of puns: *homographic pun* (hom-pun) and *heterographic pun* (het-pun) (Miller et al., 2017).

- **Hom-Pun**: Hom-puns play on the dual meaning of homographs (Attardo, 2009), referring to the words that have different meanings but share the same spelling. For example, the hom-pun "*Pick (Pick) your friends, but not to pieces*" utilizes the dual entendre of the word "*pick*". The first part "*Pick your friends*" suggests choosing or selecting friends. However, combined with the second part "*but not to pieces*", it evokes the phrase "*pick someone to pieces*", meaning to criticize or find fault with someone. This pun leads to an unexpected twist and creates humor.
- **Het-Pun**: Het-puns leverage the double meaning of paronyms or homophones (Attardo, 2009), both of which are similar-sounding words but with different meanings. Take the het-pun "*Life is a puzzle, look here for the missing peace (piece)*" as an example. The word "*peace*" typically refers to tranquility or serenity in life. Meanwhile, it can be easily recognized as the homophone "*piece*", as in a puzzle piece. This play on "*peace*" and "*piece*" delivers a humorous dual entendre.

In the above two examples, the underlined parts represent the pun-alternative word pair (He et al., 2019), with the alternative word in (parentheses). For hom-puns, the pun word w_p and the alternative word w_a are identical. For het-puns, these two words have a similar pronunciation, but only the former appears in the sentence. Both w_p and w_a have their respective meanings: pun sense S_p and alternative sense S_a, which are supported by the clever use of contextual words C_w. In the first instance, the C_w are "*friend*" and "*to piece*", and in the second example, are "*life*" and "*puzzle*". Following the notation of Sun et al. (2022b), we refer to w_p, w_a, S_p, and S_a together as the *pun pair*, denoted as $P_p =< w_p, w_a, S_p, S_a >$.

4 Probing Protocol

In this section, we design an evaluation protocol consisting of three progressive tasks to assess whether LLMs can understand puns well.

4.1 Task Formulation

Task 1: Pun Recognition This task requires the LLM to determine the corresponding category $C \in \{\text{pun}, \text{non-pun}\}$ for a given text T, as shown in the following two examples.

> **Input Text**: Pick your friends, but not to pieces.
> **Model Output**: The given text is a pun.
> **Input Text**: A man's home is his castle.
> **Model Output**: The given text is a non-pun.

Task 2: Pun Explanation This task asks the LLM to provide a natural language explanation E for a given pun text T_p, by explicitly clarifying each element of the pun pair and the humor they express. Here is an example:

> **Input Text**: Life is a puzzle, look here for the missing peace.
> **Model Output**: The text uses the homophones "piece" and "peace". "Piece" is expected in a puzzle context, but "peace" is used, shifting the meaning to tranquility. Thus it delivers a sense of humor.

Task 3: Pun Generation This task requires the LLM to generate a pun text T_p based on the input. We explore two types of inputs in our settings. Both types accept a pun pair P_p as the basic input, but one can freely use context, while the other must utilize the given contextual words C_w. In the following two examples, senses S_p and S_a are enclosed with "<>":

> **Pun Pair** P_p: peace <freedom from disputes>; piece <separate part of a whole>
> **Model Output**: When the pie was divided, everyone had a peace.
> **Pun Pair** P_p: peace <freedom from disputes>; piece <separate part of a whole>
> **Contextual Words** C_w: life, puzzle
> **Model Output**: In the puzzle of life, finding peace is difficult.

4.2 Task Implementation

We design specific prompts for LLMs to test their inherent abilities on these three tasks.[1]

- For **pun recognition**, we focus on the model's accuracy and confidence in its response. Therefore, we craft two slightly biased instructions (one leaning towards pun and the other non-pun) in the prompt. We also incorporate the definition of puns and several examples into the prompt to assess their impact.
- For **pun explanation**, we introduce the Chain-of-Thought (CoT) technique (Wei et al., 2022) in the recognition prompt, which requires the LLM to provide the reason before making a decision. The "reason" part is directly collected as the corresponding explanation.
- For **pun generation**, we employ two prompts with different requirements. In the free mode, LLM can freely choose its context based on the given P_p. In the restricted mode, LLM needs to leverage the words from C_w as much as possible. This enables us to evaluate the LLM's capacity to generate puns freely and under constraints.

[1]All prompts for three pun-related tasks are available at Appendix B.

Data Split	Examples			Test Data		
	hom	het	non	hom	het	non
Hom-Dataset	10	0	10	810	0	633
Het-Dataset	0	10	10	0	647	499

Table 1: Dataset statistics. We use "hom", "het", and "non" to represent hom-puns, het-puns, and non-puns.

4.3 Dataset Construction

The dataset used in our evaluations integrates the Semeval-2017-Task-7 dataset (Miller et al., 2017) with the ExPun dataset (Sun et al., 2022a). The former is a widely used open-source pun dataset, while the latter augments the former with detailed crowdsourced annotations. Since these two datasets are not perfectly aligned and some data in ExPun lack explanations for puns, we conduct a review and filtered out some of the data. Through this process, we ensure that each pun entry includes the pun text, pun pair, human explanation, and keyword set, whereas each non-pun entry contains only non-pun text.[2] The keyword set here serves as the contextual words C_w for generating puns since it usually provides a proper context without hindering the model's generation. We divide the entire dataset into two parts: the hom-dataset and the het-dataset, and select a small number of samples as the demonstration examples in prompts, as shown in Table 1.

4.4 Model Selection

To assess the pun understanding level of LLMs with varying parameter sizes and capabilities, we selected eight well-known LLMs from two categories for our experiments. The first category includes open-source 7B models, such as Llama2-7B-Chat (Touvron et al., 2023), Mistral-7B (Jiang et al., 2023), Vicuna-7B (Zheng et al., 2024), and OpenChat-7B (Wang et al., 2024). The second category consists of closed-source models with larger parameter scales, like Gemini-Pro (Google, 2023), GPT-3.5-Turbo (OpenAI, 2023a), Claude-3-Opus (Anthropic, 2024) and GPT-4-Turbo (OpenAI, 2023b). All of them are generative text models endowing with in-context learning and instruction-following abilities.

[2]We selected the longest explanation and the most extensive set of keywords in ExPun, expecting them to be more informative.

4.5 Evaluation Metrics

Metrics for Recognition We measure the accuracy and confidence of LLMs on pun recognition through the following three indicators. *1)* **True Positive Rate (TPR)** (Yerushalmy, 1947) indicates the ratio of puns correctly identified. *2)* **True Negative Rate (TNR)** is the ratio of non-puns accurately recognized. *3)* **Cohen's Kappa (κ)** (Cohen, 1960) measures the agreement between two sets of biased recognitions. Moreover, we compute the **variations (Δ)** in TPR and TNR when the prompt leans towards non-pun compared to pun, as they reflect the model's inconsistency intuitively.

Metrics for Explanation Considering the labor-intensive and time-consuming nature of manually evaluating pun explanation, we combine manual assessment with automatic evaluation according to the following two methods. *1)* A small-scale, fine-grained *punchline check*: We randomly select 100 hom-puns and 100 het-puns and employ three annotators to assess the quality of their explanations.[3] For each sample, we ask annotators to check whether elements of the pun pair $P_p =< w_p, w_a, S_p, S_a >$ are correctly mentioned in the explanation. Their annotations demonstrate a high level of agreement (with Fleiss's $\kappa = 0.87$), highlighting the reliability of this method. In cases of disagreement, we adopt the majority view. Then, we compute the average mentioned ratio (denoted as **Average Mention Ratio**) of w_p, w_a, S_p, and S_a as indicators. *2)* A large-scale coarse-grained *pairwise comparison*: We instruct GPT-4 (OpenAI, 2023c) to choose the winner between the human explanation and the model explanation (allowing for a tie), and then calculate the **Win Rate**, **Tie Rate**, and **Loss Rate** of each LLM. This kind of approach is widely used for evaluation (Li et al., 2024; Yuan et al., 2024; Qin et al., 2024). It is worth noting that GPT-4 achieves a high level of consistency with our annotators, showing an accuracy of 88.3% on the sampled data.

Metrics for Generation The metrics used for pun generation in our study consist of two main dimensions: *1)* automatic indicators, which are primarily based on word probability modeling, like **Ambiguity (A)**, **Distinctiveness (D)** (Kao et al., 2016), and **Surprise (S)** (He et al., 2019).[4] We

[3]More information about our annotators can be found in Appendix A.

[4]The formula for calculating these metrics and the details of their implementation are available at Appendix C.

Model	Homographic Pun					Heterographic Pun				
	TPR	Δ_{TPR}	TNR	Δ_{TNR}	κ	TPR	Δ_{TPR}	TNR	Δ_{TNR}	κ
Basic Prompt (with only Instruction and Test Data)										
Llama2-7B-Chat	0.993	-0.128	0.049	+0.294	0.148	0.985	-0.083	0.042	+0.323	0.173
Vicuna-7B	0.984	-0.299	0.028	+0.376	0.077	**0.997**	-0.195	0.024	+0.419	0.055
Mistral-7B	0.867	-0.533	0.208	+0.540	0.156	0.873	-0.442	0.202	+0.585	0.175
OpenChat-7B	0.948	-0.073	0.368	+0.120	0.722	0.930	-0.068	0.379	+0.120	0.742
Gemini-Pro	**0.998**	-0.048	0.166	+0.506	0.287	0.983	-0.133	0.192	+0.467	0.296
GPT-3.5-Turbo	0.990	-0.137	0.224	+0.510	0.291	0.977	-0.148	0.263	+0.467	0.342
Claude-3-Opus	0.989	-0.011	0.624	+0.109	0.867	0.969	-0.037	0.613	+0.096	0.839
GPT-4-Turbo	0.988	**-0.003**	**0.630**	**+0.054**	**0.894**	0.960	**-0.020**	**0.621**	**+0.048**	**0.884**
Enhanced Prompt (with Additional Pun Definition and 6 Examples)										
Llama2-7B-Chat	0.738	+0.123	0.306	-0.071	0.309	0.770	+0.153	0.501	-0.313	0.208
Vicuna-7B	0.986	**-0.001**	0.112	+0.016	0.726	0.985	**+0.000**	0.283	+0.044	0.842
Mistral-7B	0.569	-0.181	0.798	+0.076	0.696	0.553	-0.158	**0.894**	+0.064	0.722
OpenChat-7B	0.890	-0.063	0.556	+0.107	0.816	0.873	-0.060	0.667	+0.048	0.881
Gemini-Pro	**0.998**	-0.058	0.460	+0.422	0.519	0.982	-0.097	0.499	+0.349	0.555
GPT-3.5-Turbo	0.974	-0.036	0.611	+0.137	0.811	0.935	-0.056	0.699	+0.106	0.814
Claude-3-Opus	0.982	-0.005	**0.806**	+0.041	0.953	**0.991**	-0.003	0.750	+0.070	0.929
GPT-4-Turbo	0.988	**-0.001**	0.758	**+0.010**	**0.962**	0.961	+0.008	0.796	**-0.006**	**0.959**

Table 2: Results of two biased pun recognition. Apart from TPR, TNR, and κ, we also compute the variations (Δ) in TPR and TNR when the prompt bias shifts from pun to non-pun. These variations are similarly marked based on their absolute values. The best results (smallest variations) are **bolded**, and the second-best results are underlined.

also examine the inclusion rates of the pun and the contextual word in the generation, denoted as **One-pun-word Incorporation Rate** ($1w_p$) and **Contextual Word Incorporation Rate** (Sun et al., 2022b). *2)* manual indicators, which include **Success Rate** and **Funniness Rating** of human puns and LLM-generated puns. We ask our annotators to identify whether a pun text is successful and rate its funniness on a scale from 1 to 5. These annotations are performed on the same subset (100 hom-puns and 100 het-puns) of our dataset.

5 Results and Analysis

5.1 *Can LLMs Distinguish Between Puns and Non-puns?*

We design two types of prompts for pun recognition: The first type is the *basic prompt*, which only includes test data and biased instructions. The second type is the *enhanced prompt*, which adds to the basic prompt with the definition of puns and some examples (3 puns and 3 non-puns).[5]

As shown in Table 2, we can find that: *1)* Almost all tested LLMs are influenced by the bias in the prompt, leading to results that tend to align with this bias. Some models, such as Vicuna-7B, Mistral-7B, Gemini-Pro, and GPT-3.5-Turbo, show significant fluctuations in their responses, indicating their lack of confidence in their answers. *2)* Adding a definition and examples as additional information significantly improves the consistency between LLMs' two responses. It also enhances the models' accuracy in recognizing non-puns. *3)* The TNR metric is generally lower than the TPR. This discrepancy arises because non-puns in our dataset are mostly non-pun jokes and proverbs. They are somewhat similar to puns. *4)* There is no obvious difference in recognizing hom-puns and het-puns.

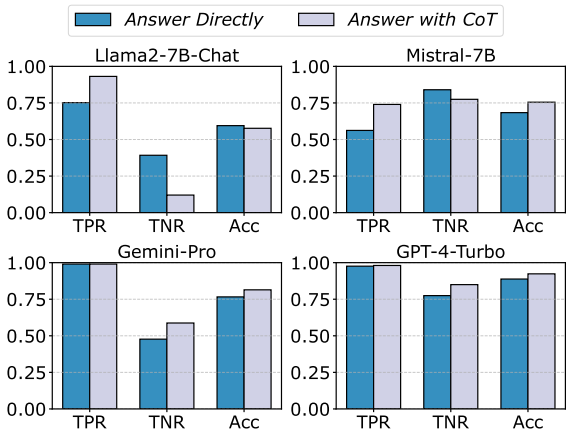

Figure 2: The performance of four selected LLMs in recognizing puns via direct answers and CoT responses. The Acc metric represents the overall accuracy.

[5]To explore the respective roles of definition and examples, we conduct ablation studies on the GPT series, with the results presented in Appendix D.

Model	Homographic Pun				Heterographic Pun			
	w_p	w_a	S_p	S_a	w_p	w_a	S_p	S_a
Llama2-7B-Chat	0.63	0.63	0.45	0.42	0.69	0.11	0.47	0.13
Vicuna-7B	0.71	0.71	0.64	0.59	0.85	0.21	0.81	0.29
Mistral-7B	0.78	0.78	0.73	0.68	0.69	0.22	0.68	0.22
OpenChat-7B	0.81	0.81	0.72	0.71	0.77	0.28	0.74	0.33
Gemini-Pro	0.92	0.92	0.87	0.81	0.89	0.42	0.83	0.42
GPT-3.5-Turbo	0.88	0.88	0.81	0.81	0.91	0.55	0.82	0.57
Claude-3-Opus	0.96	0.96	0.95	0.92	0.95	0.84	**0.94**	0.78
GPT-4-Turbo	**0.98**	**0.98**	**0.96**	0.93	0.96	0.90	0.93	0.85
Human	0.95	0.95	0.95	**0.95**	**0.97**	**0.97**	**0.94**	**0.93**

Table 3: Results of punchline check for pun explanations. We represent the average mention ratio of the pun pair elements in explanations with the corresponding symbols. The top outcomes are **bolded** and the second best are underlined.

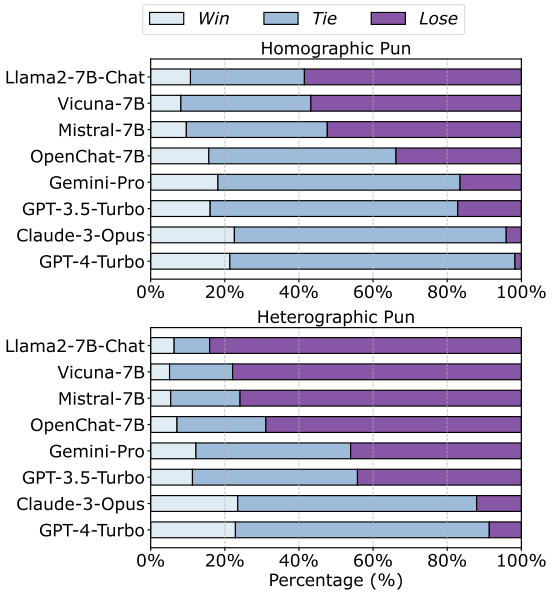

Figure 3: Results of pairwise comparison for pun explanations.

This may reveal that LLMs capture the core feature (*i.e.*, dual meanings) of puns and use it as the main criterion for judgment. *5)* GPT-4-Turbo and Claude-3-Opus demonstrate exceptional performance, exhibiting satisfactory pun recognition capabilities.

CoT Prompting Although we primarily use CoT to obtain explanations of puns from LLMs, it also offers an opportunity to explore its impact on the pun recognition task. We differentiate between two response methods based on the enhanced prompt: answering directly and answering with CoT, while keeping the prompt's bias towards pun. Then, we select four models and chart their performance in Figure 2. It is observable that, except for LLama2-7B-Chat, the remaining three LLMs showed an overall improvement in accuracy after using CoT. Notably, Gemini-Pro and GPT-4-Turbo's weak spots in recognizing non-pun text are compensated for through CoT response, showcasing a stronger ability to distinguish between puns and non-puns.

5.2 *Can LLMs Explain the Humor in Puns?*

The humor in puns mainly stems from exploiting double entendre. Thus, explaining the humor in a pun is akin to identifying its dual meanings or, more precisely, the corresponding pun pair.

We present the results of the punchline check in Table 3. This evaluation shows that: *1)* Most LLMs accurately identify the pun words w_p in both hom-puns and het-puns, which is fundamental to explaining puns. *2)* Except for GPT-4-Turbo and Claude-3-Opus, the remaining LLMs struggle to identify alternative words w_a and alternative sense S_a in het-puns. This challenge arises because w_a in het-puns does not directly appear in the text but relies on evocation through context and similar pronunciation to w_p.

Unlike the detail-oriented punchline check, pairwise comparison focuses on the overall quality of explanations. Its results, illustrated in Figure 3, indicate that: *1)* LLMs generally perform worse at explaining het-puns than hom-puns, aligning with the findings in the punchline check. Based on the results of pun recognition, we infer that alternative words do not affect pun recognition but are crucial for correctly explaining puns. *2)* The explanations by GPT-4-Turbo and Claude-3-Opus often approach or even surpass those by humans. We find that LLMs consistently use a general-to-specific structure in their explanations, whereas human explanations tend to be more casual.[6] This aspect gives the two models an edge in comparison.

Error Types in Explanation LLMs tend to make various mistakes when explaining puns, and we categorize the primary errors as follows: *1) Misclassify pun as non-pun*, which means the model fails to detect the double meaning. *2) Incorrect pun word identification*, which means the model fails to find the correct w_p. *3) Incorrect alternative word identification*, a mistake only made in the explanation of het-puns, which means the model fails to evoke the correct w_a. *4) Misinterpret het-*

[6]The structure of the pun explanation is further discussed in Appendix E.1.

Model	Homographic Pun						Heterographic Pun					
	A	**D**	**S**	$1w_p$	Success	Funny	**A**	**D**	**S**	$1w_p$	Success	Funny
Generated non-pun	0.195	0.037	-0.640	0.983	0.010	1.042	0.113	0.071	-0.734	0.978	0.010	1.014
Pun Generation with only Pun Pair												
Llama2-7B-Chat	0.206	0.033	-0.130	0.425	0.060	1.071	0.168	0.155	-0.029	0.145	0.040	1.042
Vicuna-7B	0.223	0.062	-0.249	0.690	0.120	1.216	0.211	0.088	-0.272	0.377	0.050	1.128
Mistral-7B	0.193	0.072	-0.239	0.583	0.170	1.321	0.211	0.151	-0.156	0.343	0.130	1.336
OpenChat-7B	0.208	0.058	-0.168	0.549	0.200	1.261	0.207	0.136	-0.261	0.271	0.080	1.128
Gemini-Pro	0.222	0.038	-0.203	0.680	0.320	1.699	**0.241**	0.072	-0.076	0.383	0.150	1.336
GPT-3.5-Turbo	0.220	0.064	-0.233	0.714	0.420	1.367	0.223	0.073	0.072	0.521	0.290	1.306
Claude-3-Opus	0.211	**0.073**	-0.150	**0.893**	0.540	2.050	0.200	**0.208**	0.096	**0.915**	0.470	1.931
GPT-4-Turbo	**0.225**	0.047	**-0.027**	0.890	**0.600**	**2.016**	0.221	0.098	**0.121**	0.847	**0.510**	**1.948**
Pun Generation with Pun Pair and Relevant Contextual Words												
Llama2-7B-Chat	0.205	0.107	-0.093	0.605	0.340	1.602	0.180	**0.235**	-0.066	0.352	0.220	1.413
Vicuna-7B	0.199	0.077	-0.181	0.782	0.300	1.650	0.182	0.238	0.015	0.453	0.210	1.459
Mistral-7B	0.186	**0.115**	-0.201	0.616	0.280	1.618	0.176	0.213	0.108	0.373	0.220	1.506
OpenChat-7B	0.196	0.091	-0.133	0.636	0.370	1.715	0.166	**0.235**	0.013	0.352	0.240	1.522
Gemini-Pro	0.221	0.079	-0.200	0.689	0.440	1.880	0.198	0.149	0.142	0.581	0.330	1.731
GPT-3.5-Turbo	0.217	0.079	**-0.076**	0.856	0.550	2.137	**0.216**	0.163	0.205	0.543	0.320	1.699
Claude-3-Opus	**0.237**	0.081	-0.131	**0.907**	0.650	2.438	0.206	0.185	0.275	**0.849**	**0.610**	**2.348**
GPT-4-Turbo	0.217	0.082	-0.217	0.880	**0.670**	**2.584**	0.199	0.168	**0.285**	0.794	0.600	**2.348**
Human pun	0.225	0.129	-0.069	0.990	0.860	3.268	0.185	0.256	0.323	0.985	0.840	3.229

Table 4: Results of pun generation. We abbreviate the metrics Ambiguity, Distinctiveness, Surprise, and One-pun-word Incorporation Rate as "A", "D", "S" and "$1w_p$", respectively. For each generation method, the best results appear in **bold** and the second best are underlined.

pun as hom-pun, which means the model wrongly classifies the pun's genre. *5) Lack of meaning analysis*, which means the model points out w_p and w_a but skips explaining the dual meanings. *6)* Fabricating non-existent meanings, which means the model invents meanings for w_p or w_a that do not exist. We provide a case for each type of error in Appendix E.2 to help readers understand them. We believe addressing these errors is key to enabling LLMs to generate better explanations of puns.

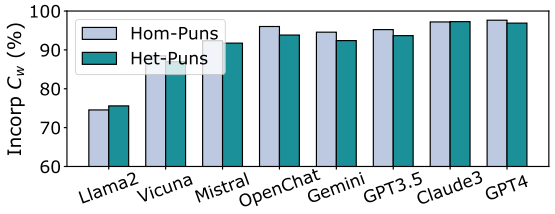

Figure 4: Contextual word incorporation rate of different LLMs in constrained pun generation.

5.3 *Are LLMs Capable of Generating Puns?*

To answer this question, we first ask GPT-3.5-Turbo to generate non-puns containing the same pun words w_p as human puns, to serve as a baseline. Then, we request all tested LLMs to generate puns under two different inputs mentioned in § 4.1.

From Figure 4, we can see that with the exception of LLama2-7B-Chat, all other LLMs can easily accomplish the task of constrained generation. They are notably efficient at incorporating nearly all contextual words C_w in the generated sentences. Other metrics are presented in Table 4. Our analysis reveals that: *1)* All LLMs demonstrate a noticeably weaker ability to generate het-puns than hom-puns, indicating that het-pun generation is a more challenging task. *2)* Since all C_w are derived from human puns, we believe LLMs can grasp the intrinsic relationship between these words and the given pun pair, thereby improving the quality and success rate of the generated puns. This suggests that providing good context helps in pun generation. *3)* Most LLMs, especially the 7B models, tend to include multiple w_p when generating puns. This phenomenon is rarely seen in human puns, which usually leads to the failure of pun generation. *4)* GPT-4-Turbo and Claude-3-Opus achieve impressive success in generating puns and rival the traditional SOTA methods, which has a success rate of 56% for hom-puns and 47% for het-puns (Tian et al., 2022). However, the puns they generate are still not as funny as those created by humans.

Lazy Pun Generation Samples
/ Pun Pair */* dock <deprive someone of benefits, as a penalty> dock: <come into dock> */* Human Pun */* When longshoremen show up late for work they get docked. */* LLM Generation */* The sailor's pay was docked after he struggled to dock on time.
/ Pun Pair */* two <the cardinal number that is the sum of one and one> too <to a degree exceeding normal or proper limits> */* Human Pun */* My friend gave me a book about puns for my birthday and I loved it. It was two meaningful. */* LLM Generation */* I tried to make puns about numbers, but two were too much to handle.

Table 5: Examples of LLMs' lazy pun generation pattern. We underline the w_p and w_a in human puns and LLM-generated puns.

"Lazy Pun Generation" Pattern No matter how much the prompt emphasizes that only one w_p should be used, most LLMs frequently generate text containing two or even more w_p (and w_a for het-puns), as shown in Table 5. We refer to this stubborn pattern as *lazy pun generation*, and classify pun sentences produced in this pattern as unsuccessful. We attribute this pattern to two main reasons. Firstly, including multiple w_p allows for expressing double meanings at different parts of the sentence, making the construction relatively simple. Secondly, the current definitions of puns do not explicitly limit the number of w_p and w_a used. Avoiding w_a in het-puns and adopting a single w_p is an unwritten rule that most human-crafted puns follow, but LLMs often ignore. Since adding corresponding restrictions in the prompt can slightly alleviate this issue, we believe it would be more helpful for the LLM to learn this explicitly through definitions or cases during training.

Copying or Originality? LLMs are trained on vast amounts of text. It's essential to ascertain whether they merely reproduce existing puns or genuinely create new ones. To assess this, we developed an *Overlap* metric to measure the similarity between puns created by models and those by humans. The metric's computation involves three steps. First, we identify the lemma word sets in puns generated by LLMs and humans, labeled as Pun_{LLM} and Pun_{human}. Next, we eliminate the words w_p, w_a, and C_w provided in the prompt, resulting in refined sets $\tilde{Pun}_{LLM}$ and $\tilde{Pun}_{human}$. Finally, we compute the overlap ratio as the size of the intersection over the size of the union of these sets, as in the formula:

$$Overlap = \frac{|\tilde{Pun}_{LLM} \cap \tilde{Pun}_{human}|}{|\tilde{Pun}_{LLM} \cup \tilde{Pun}_{human}|}$$

We establish a coarse criteria for originality as an overlap < 0.5, thereby defining the "***Strict Success***" of pun generation, which combines success with originality. Figure 5 shows that: *1)* When given only pun pair P_p, LLMs rarely copy human puns, relying probably on self-creation. *2)* When given additional C_w, the likelihood of LLMs reproducing human puns increases slightly, leading to a decrease in strict success. We also find that the larger the LLM, the more prone it is to do this, suggesting that their stronger memory of the corpus adversely affects the generation of creative puns.

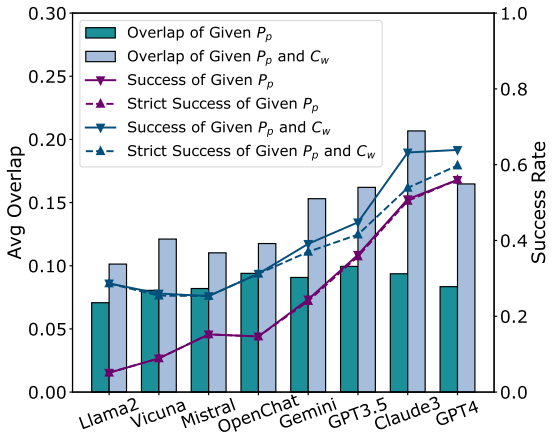

Figure 5: Average overlap, success, and strict success of two methods for generating puns.

6 Conclusion

In this paper, we examine the ability of large language models (LLMs) to understand puns. We employ three tasks: pun recognition, pun explanation, and pun generation, and develop various metrics to systematically assess the capabilities of LLMs in these areas. Experiments indicate that although LLMs perform satisfactorily in recognizing and explaining puns, there is still room for improvement in their ability to generate creative and humorous puns. We also suggest that het-pun explanation and generation are more difficult than those of hom-pun. We believe our evaluation methods and findings will contribute to advancing research on pun understanding.

Limitations

Although we utilize the most widely used pun dataset currently available to evaluate the pun-understanding ability of LLMs, our pun texts are all in English. The ability of LLMs to understand puns can vary across different languages, and puns in languages other than English may have different definitions, structures, or purposes. Such a limitation highlights the potential for future work to generalize to puns in other languages.

In addition, given that LLMs have massive training data, most of which are not publicly available, it is possible that LLMs just copy puns that are not present in our dataset. Thus, our *Overlap* metric is not a precise measurement but only roughly indicates the extent of LLMs' plagiarism when generating puns. Since exploring originality is intriguing, we eagerly hope for future work to develop more accurate indicators.

Another limitation of our work stems from potential biases in the evaluation process. Evaluating the quality of a pun explanation and the success of a generated pun involves human annotator judgments. Preferences vary among individuals: some may prefer detailed explanations, while others might seek clarity and brevity. Moreover, a pun that amuses one person may offend another. Future studies can consider designing more appropriate evaluation metrics.

Ethics Statement

We acknowledge that all authors are informed about and adhere to the ACL Code of Ethics and the Code of Conduct.

Use of Human Annotations Our institution recruited annotators to implement the annotations of pun evaluation. We ensure the privacy rights of the annotators are respected during the annotation process. The annotators receive compensation exceeding the local minimum wage and have consented to the use of pun data generated by them for research purposes. Appendix A provides further details on the annotations.

Risks The pun datasets in our experiment are sourced from publicly available sources. However, we cannot guarantee that they are devoid of socially harmful or toxic language. Furthermore, evaluating the data quality of pun explanation and generation is based on common sense, which can vary among individuals from diverse backgrounds. We use ChatGPT to correct grammatical errors in this paper.

Acknowledgement

We thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the Chinese NSF Major Research Plan (No.92270121).

References

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Salvatore Attardo. 2009. *Linguistic theories of humor*, chapter 3. Walter de Gruyter.

Salvatore Attardo. 2018. Universals in puns and humorous wordplay. *Cultures and traditions of wordplay and wordplay research*, pages 89–110.

Alexander Baranov, Vladimir Kniazhevsky, and Pavel Braslavski. 2023. You told me that joke twice: A systematic investigation of transferability and robustness of humor detection models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13701–13715, Singapore. Association for Computational Linguistics.

Minje Choi, Jiaxin Pei, Sagar Kumar, Chang Shu, and David Jurgens. 2023. Do LLMs understand social knowledge? evaluating the sociability of large language models with SocKET benchmark. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11370–11403, Singapore. Association for Computational Linguistics.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Yufeng Diao, Hongfei Lin, Di Wu, Liang Yang, Kan Xu, Zhihao Yang, Jian Wang, Shaowu Zhang, Bo Xu, and Dongyu Zhang. 2018. WECA: A WordNet-encoded collocation-attention network for homographic pun recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2507–2516, Brussels, Belgium. Association for Computational Linguistics.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at Semeval-2017 task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 103–108, Vancouver, Canada. Association for Computational Linguistics.

Marta Dynel. 2009. Beyond a joke: Types of conversational humour. *Language and linguistics compass*, 3(5):1284–1299.

Gemini Team Google. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Aishwarya Gupta, Avik Pal, Bholeshwar Khurana, Lakshay Tyagi, and Ashutosh Modi. 2021. Humor@IITK at SemEval-2021 task 7: Large language models for quantifying humor and offensiveness. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 290–296, Online. Association for Computational Linguistics.

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. 2008. Computational humor: Beyond the pun? *The Primer of Humor Research. Humor Research*, 8:333–360.

Jack Hessel, Ana Marasovic, Jena D. Hwang, Lillian Lee, Jeff Da, Rowan Zellers, Robert Mankoff, and Yejin Choi. 2023. Do androids laugh at electric sheep? humor "understanding" benchmarks from the new yorker caption contest. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–714, Toronto, Canada. Association for Computational Linguistics.

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.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Dayoon Ko, Sangho Lee, and Gunhee Kim. 2023. Can language models laugh at YouTube short-form videos? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2897–2916, Singapore. Association for Computational Linguistics.

Shuang Li, Jiangjie Chen, Siyu Yuan, Xinyi Wu, Hao Yang, Shimin Tao, and Yanghua Xiao. 2024. Translate meanings, not just words: Idiomkb's role in optimizing idiomatic translation with language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18554–18563.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

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.

OpenAI. 2023a. Gpt-3.5-turbo.

OpenAI. 2023b. Gpt-4 and gpt-4-turbo.

OpenAI. 2023c. Gpt-4 technical report.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Sahithya Ravi, Patrick Huber, Akshat Shrivastava, Aditya Sagar, Ahmed Aly, Vered Shwartz, and Arash Einolghozati. 2024. Small but funny: A feedback-driven approach to humor distillation. *arXiv preprint arXiv:2402.18113*.

Andrew Stott. 2014. *Comedy*. 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.

Joe Toplyn. 2023. Witscript 3: A hybrid ai system for improvising jokes in a conversation. *arXiv preprint arXiv:2301.02695*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2024. Openchat: Advancing open-source language models with mixed-quality data. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Yuhuan Xiu, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 task 7: Using supervised and unsupervised methods to detect and locate English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 453–456, Vancouver, Canada. Association for Computational Linguistics.

Jacob Yerushalmy. 1947. Statistical problems in assessing methods of medical diagnosis, with special reference to x-ray techniques. *Public Health Reports (1896-1970)*, 62(40):1432–1449.

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.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Ren Kan, Dongsheng Li, and Deqing Yang. 2024. Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Shanshan Zhong, Zhongzhan Huang, Shanghua Gao, Wushao Wen, Liang Lin, Marinka Zitnik, and Pan Zhou. 2023. Let's think outside the box: Exploring leap-of-thought in large language models with creative humor generation. *arXiv preprint arXiv:2312.02439*.

Yichao Zhou, Jyun-Yu Jiang, Jieyu Zhao, Kai-Wei Chang, and Wei Wang. 2020. "the boating store had its best sail ever": Pronunciation-attentive contextualized pun recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 813–822, Online. Association for Computational Linguistics.

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

A Crowd-sourcing

We have recruited a team of three undergraduates who majored in English. They are very familiar with puns and are specifically trained for our evaluation work. We pay each of them $9/h, exceeding the local minimum wage. The screenshots of the instructions and annotation interface are shown in Figure 6, 7, 8.

B Details of Prompts

B.1 Prompt for Pun Recognition and Explanation

When recognizing different types of puns, we will provide accordingly 6 examples. The explanations used in experiments are collected from enhanced prompts for best performance. We set the temperature parameter to 0 to minimize the impact of the model's inherent randomness on recognition and explanation tasks.

Pun Recognition and Explanation

/ Definition */*
Puns are a form of wordplay exploiting different meanings of a word or similar-sounding words, while non-puns are jokes or statements that don't rely on such linguistic ambiguities.

/ Instruction */*
Determine whether the given Text is a **pun/non-pun.** Give your reasons first, then make your final decision clearly. You should either say "The given text is a pun" or say "The given text is a non-pun". You must output the current status in a parsable JSON format. An example output looks like {"Reason": "XXX", "Choice": "The given text is a XXX"}

/ 6 Examples */*
Text: Driving on so many turnpikes was taking its toll.
Output:{"Reason": "The text is using the word 'toll' in a double entendre. It refers both to the physical tolls paid on turnpikes and to 'taking its toll' as in having a negative effect or cost.", "Choice": "The Given test is a pun."}
Text: Nothing ventured, nothing gained.
Output:{"Reason": "The given text is a proverb that expresses a general truth or piece of advice and does not exploit different meanings of a word or similar-sounding words.", "Choice": "The given text is a non-pun."}
.....

/ Test Data /*
Text: I wanted to have dinner at a native American-themed restaurant, but I didn't have reservations.
Output:

Table 6: Prompt for pun recognition and explanation. Red Text denotes the Chain of Thought (CoT) module. We will select a single bias indicated by **bold text** at a time.

B.2 Prompt for Pun Generation

In pun generation tasks, we will provide 3 examples in the prompt and test the effect of contextual words on the final generation's quality. Here, the temperature parameter is set to 0.7, which strikes a balance between stimulating the model's creativity and preventing it from going off the rails.

Pun Generation

/ Definition */*
Puns are a form of wordplay exploiting different meanings of a word or similar-sounding words, while non-puns are jokes or statements that don't rely on such linguistic ambiguities.

/ Instruction */*
Below is a keyword, two of its meanings and a set of contextual words. Please generate a pun sentence with a punchline on the keyword that conveys both given meanings simultaneously and using all the contextual words. Except for the keyword, the pun sentence must not utilize any words from either of the two meanings. Besides, once a keyword is used, it's strictly prohibited to use it again in the latter half of the sentence. You must output the current status in a parsable JSON format. An example output looks like: {"Sentence": "XXX"}

/ 3 Examples */*
Keyword: toll
Meaning 1: toll <a fee levied for the use of roads or bridges (used for maintenance)>
Meaning 2: toll <value measured by what must be given or done or undergone to obtain something>
Contextual Words: Driving, many, turnpikes, taking its toll
Output:
{"Sentence": "{"Driving on so many turnpikes was taking its toll."}"}
.....

/ Test Data */*
Keyword: bore
Meaning 1: <Make a hole, especially with a pointed power or hand tool>
Meaning 2: <A carpenter sat on his drill and was bored to tears.>
Contextual Words: carpenter, sat, drill, bored to tears
Output:

Table 7: Prompt for pun generation. Red texts denotes the addition of contextual words.

B.3 Prompt for Non-pun Generation

We use GPT3.5-Turbo to generate non-puns as lower-bound references for the evaluation metric. This task is relatively simple so we don't provide examples. The prompt is presented in Table 8.

B.4 Prompt for Pairwise Comparison

During the preliminary experiments of pairwise comparison, we provide GPT-4 with three examples for reference. However, we later noticed that the model's performance is similar with both 0-shot

and 3-shot settings. Considering that not providing examples could significantly save on token usage, we ultimately opt for the 0-shot approach. The prompt is placed in Table 9.

Non-pun Generation

/ Definition */*
Puns are a form of wordplay exploiting different meanings of a word or similar-sounding words, while non-puns are jokes or statements that don't rely on such linguistic ambiguities.

/ Instruction */*
Below is a keyword and one of its meanings. Please generate a non-pun sentence with the keyword that conveys the given meaning. You must output the current status in a parsable JSON format. An example output looks like: {"Sentence": "XXX"}

/ Test Data */*
Keyword: thick
Meaning: <having a short and solid form or stature>
Output:

Table 8: Prompt for non-pun generation.

Pairwise Comparison

/ Definition */*
Puns are a form of wordplay exploiting different meanings of a word or similar-sounding words.

/ Instruction */*
Below is a pun text, gold meanings of pun, and two corresponding explanations. Please carefully judge which explanation is of better quality. A good explanation should point out the correct pun word and analyze the multiple meanings of the pun or similar-sounding words in detail appropriately while avoiding unnecessary or incorrect interpretations. You must choose from one of the three answers: "Explanation 1 is much better", "Explanation 2 is much better", "I'm not sure which would be better.". You must output the current status in a parsable JSON format. An example output looks like: {"Choice": "XXX"}

/ Text Data */*
Pun Text: Have another soft drink, Tom coaxed.
Gold Meanings of Pun:
1. coax < influence or urge by gentle urging, caressing, or flattering >
2. coke < Coca Cola is a trademarked cola >
Explanation 1: This is a pun on how "coaxed" sounds like "Coke" which is a brand of soft drink.
Explanation 2: The text plays on the double meaning of the word 'coaxed'. "Coaxed" can mean persuading someone to do something, but it can also refer to mixing or stirring a drink. This creates a humorous double meaning.
Output:

Table 9: Prompt for pairwise comparison.

B.5 Prompt for Finding Synonyms

For assessing ambiguity, distinctiveness, surprise, and unusualness, synonyms play a crucial role in the calculations, as detailed in Appendix C.2. So we design a prompt to find synonyms for both the pun words and alternative words in hom-puns. We use GPT-4 to complete this work.

Finding Synonyms

/ Instruction */*
Below is a pun text, one keyword, and its two meanings. The keyword is the pun in the text, which can be interpreted in two meanings. Please find two different synonyms for the keyword, each corresponding to one of the meanings. The synonyms should be able to replace the keyword in the text seamlessly to remove ambiguity, while ideally being a simple word. You must output the current status in a parsable JSON format. An example output looks like: {'Synonym 1 for Meaning 1': 'XXX', 'Synonym 2 for Meaning 2': 'XXX'}

/ 6 Examples */*
Text: Driving on so many turnpikes was taking its toll.
Keyword: toll
Meaning 1: < a fee levied for the use of roads or bridges (used for maintenance) >
Meaning 2: < value measured by what must be given or done or undergone to obtain something >
Output:
{"Synonym 1 for Meaning 1": "fee", "Synonym 2 for Meaning 2": "impact"}
.....

/ Test Data */*
Text: A boy told his parents he wanted to raise goats for a living, but he was only kidding.
Keyword: kid
Meaning 1: < tell false information to for fun >
Meaning 2: < young goat >
Output:

Table 10: Prompt for finding synonyms.

C Details of A, D, S, and U Metrics

C.1 Formulas

Ambiguity & Distinctiveness (Kao et al., 2016) Ambiguity measures the extent to which the sentence supports both pun sense and alternative sense. It's quantified by the entropy of $P(m|\mathbf{w})$, where m is either the pun word w_p or the alternative word w_a.

$$P(m|\mathbf{w}) = \sum_{\mathbf{f}} \left(P(m)P(\mathbf{f}) \prod_i P(w_i|m, f_i) \right)$$

Distinctiveness is indicative of how distinctive the meanings $m1$ (w_p) and $m2$ (w_a) are, based on the supporting subsets of words in the sentence and it's calculated by KL divergence.

$$D_{KL}(F1||F2) + D_{KL}(F2||F1)$$

Variables:

- m: Pun word or alternative word.
- $\mathbf{w}$: Context.
- $\mathbf{f}$: Indicate whether a certain word is related to the topic.
- $F1$, $F2$: Distributions of focus sets given sentence topics $m1$ and $m2$, respectively.
- D_{KL}: Symmetrized Kullback-Leibler divergence score representing the distinctiveness between $F1$ and $F2$.

Surprise & Unusualness (He et al., 2019) Surprise in puns arises from the unexpected presence of the pun word over an anticipated one within a sentence, generating humor. It's quantified by S_{ratio}.

$$S(c) = -\log\left(\frac{p(w_p|c)}{p(w_a|c)}\right)$$
$$S_{\text{local}} = S(x_{p-d:p-1}, x_{p+1:p+d}),$$
$$S_{\text{global}} = S(x_{1:p-1}, x_{p+1:n}),$$
$$S_{\text{ratio}} = \begin{cases} -1, & \text{if } S_{\text{local}} < 0 \text{ or } S_{\text{global}} < 0, \\ \frac{S_{\text{local}}}{S_{\text{global}}}, & \text{otherwise.} \end{cases}$$

Unusualness attends to the pun's anomalous nature, and it's quantified by:

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

Variables are as follows:

- w_p: Pun word.
- w_a: Expected alternative word.
- c: Context.
- S_{local}: Local surprisal.
- S_{global}: Global surprisal.

C.2 Implementation

Similar to previous papers, we apply a SkipGram model (Mikolov et al., 2013) to evaluate **A**mbiguity and **D**istinctiveness, and use an LM pre-trained on WikiText (Merity et al., 2016) to evaluate **S**urprise. Given that these models had limited vocabularies, we only calculate metrics for generations with words within the model's lexicon. Additionally, we exclude the top 2% of extreme values in **S** calculations to prevent distortion of results due to near-zero denominators.

These metrics based on word probability modeling require the pun words and alternative words to differ, which is not a problem for het-puns but poses a challenge for hom-puns. To address this issue, we adopt the idea of Tian et al. (2022). This involves finding synonyms for both w_p and w_a according to the double meanings in hom-puns and then using these synonyms as substitutes for calculating the metrics. The prompt for synonym generation can be referenced in Table 10.

Prompt Type	TPR	Δ_{TPR}	TNR	Δ_{TNR}	κ
GPT-3.5-Turbo					
Basic	**0.990**	-0.137	0.224	+0.510	0.291
Basic+D	0.974	-0.137	0.488	+0.360	0.511
Basic+E	0.984	**-0.012**	0.469	**+0.134**	**0.817**
Basic+D+E	0.974	-0.036	**0.611**	+0.137	0.811
GPT-4-Turbo					
Basic	**0.988**	-0.003	0.630	+0.054	0.894
Basic+D	0.972	+0.004	**0.840**	**-0.003**	0.949
Basic+E	0.986	**-0.001**	0.698	**+0.003**	0.952
Basic+D+E	**0.988**	**-0.001**	0.758	+0.010	**0.962**

Table 11: Results of ablation study for prompt modules in homographic pun recognition. We use "D" and "E" to represent Definition and Example separately. The best results (smallest variations) are **bolded**, and the second-best results are underlined.

D Ablation Study for Prompt Modules in Pun Recognition

In addition to the basic and enhanced prompts, we also design two semi-enhanced prompts: basic+definition and basic+examples. We conduct experiments on GPT-3.5-Turbo and GPT-4-Turbo to verify the effect of different prompt modules on pun recognition. Since the results for hom-pun and het-pun are similar, we only include the hom-pun results in Table 11. From this table, we can learn that both pun definition and ICL examples aid in recognizing puns. However, for GPT-4-Turbo, providing only the pun definition improves TNP more effectively than giving both. It seems that GPT-4-Turbo, as a top-tier LLM, can sufficiently learn and distinguish between puns and non-puns from definitions alone, while additional examples may interfere with its judgment.

E Case Study of LLM-Generated Explanations

E.1 Text Structure

When interpreting puns, LLMs typically start by identifying the w_p and w_a and then analyze the dual meanings in the surrounding context. This clear main-and-subordinate structure is not very common in human explanations, which makes them less intuitive than those by models, as shown in Table 12.

Pun Explanation Samples
/ Pun Text */* Actors are cast through their good and bad stages. */* Human Explanation */* This sentence is a pun. It's saying that actors are cast through good and bad acting. The word used is stages instead of acting. A stage is a raised platform that actors act on. */* LLM Explanation */* The text plays on the double meaning of the word 'stages'. It refers to both the phases of an actor's career (good and bad times) and the physical stages (platforms) in theaters where actors perform.
/ Pun Text */* Is that a pure rump roast? Nothing butt. */* Human Explanation */* This is a joke on the fact that rump roasts come from the butt of the animal, and the speaker answering the question here is confirming that the product is nothing but pure rump roast roast. */* LLM Explanation */* The text plays on the homophones 'butt' and 'but', using 'butt' in the context of a rump roast (which refers to the rear or buttocks of the animal) and 'but' as a conjunction typically used to introduce a phrase or clause contrasting with what has already been mentioned. This creates a humorous double meaning.

Table 12: Comparison between human pun explanations and LLM-generated pun explanations. We underline the segments that explicitly point out the w_p and w_a.

E.2 Error Types

In our analysis, we've identified recurring error patterns in the output of LLMs and will examine these trends in the forthcoming case studies. All results below are acquired from GPT-3.5's result, which we find struck a balance between generation quality and noticeability of erroneous patterns.

Misclassify Pun as Non-pun For certain common puns, LLM tends to classify them as proverbs instead of puns. This shows LLM's comprehension of puns is superimposed by its understanding of certain common expressions. An example can be found in Table 13.

Misclassify Pun as Non-pun
/ Pun Text */* He who lends to the poor gets interest from god. */* Right explanation of Human */* This is a pun about earning interest, as in interest rates on money and fascination, from god. Helping a poor person is considered a good gesture in the eyes of religion. */* Wrong Explanation of LLM */* The text is a proverb conveying a moral lesson and does not exploit different meanings of a word or similar-sounding words for humorous effect.

Table 13: LLM misclassifies pun as non-pun in explanation.

Incorrect Pun Word Identification For certain het-puns, LLM can't correctly identify the pun word because the relationship between the pun word and alternative words is a bit far. Like in this example (Table 14), we need to associate "slow" with "sloe" using phonetic similarity and then associate "sloe" with "blackthorn" by semantic similarity. These types of puns are challenging for current LLMs.

Incorrect Pun Word Identification
/ Pun Text */* 'This looks like the fruit of the blackthorn', said Tom slowly. */* Right explanation of Human */* This is a play on words. The word "slowly" means at a slow speed but a "sloe" is the fruit of a blackthorn.. */* Wrong Explanation of LLM */* The text plays on the double meaning of 'blackthorn', which refers to both the fruit of the blackthorn plant and the color black. This creates a humorous double meaning.

Table 14: LLM incorrectly pinpoints pun word in explanation.

Incorrect Alternative Word Identification In some het puns, LLM's explanation shows a strange pattern: it can't identify the correct alternative word, but it can give out the right alternative meaning. Like in this example (Table 15), it doesn't pinpoint the pun word: "canvass" but the explanation includes its meaning: "the act of going door-to-door to promote a product or service". For these cases, we think LLM truly understands both meanings of the sentence. However, its inability to wield phonetic similarity leads the failure in alternative word identification.

Incorrect Alternative Word Identification
/ Pun Text */* In his latest promotional campaign, the manager of City Tent and Awning told his crew to canvas the entire area. */* Right explanation of Human */* "Canvas" can both mean a noun that refers to a type of rough, durable cloth or fabric commonly used in tents and outdoor coverings such as awnings, and a verb that means searching or covering a given area, usually in the context of personal outreach for a given goal or mission. The play on words here is that the manager of a business selling tents and awnings, commonly made of canvas, is telling his employees to help promote the business by *canvassing* the area to that end. */* Wrong Explanation of LLM */* The text plays on the double meaning of 'canvas'. In the context of a promotional campaign, 'canvas' refers to the act of going door-to-door to promote a product or service. However, 'canvas' is also a material used for making tents and awnings, which relates to the manager's business. This creates a humorous double meaning.

Table 15: LLM incorrectly pinpoints alternative word in explanation.

Misinterpret Het as Hom A frequently appearing pattern in het-puns explanations. These explanations include expressions such as: "use the similar sounding of", and "plays on the homophones of". Demonstrating that LLM doesn't have a firm grasp of the definition of het-pun and hom-pun. An example is presented in Table 16.

Misinterpret Het as Hom
/ Pun Text */* "I've been listening to the Brandenburg Concertos", Tom barked */* Right explanation of Human */* 'Bark' sounds like Bach, who is the composer of Brandenburg Concertos. Using 'bark' creates a humorous effect */* Wrong Explanation of LLM */* The text plays on the homophones 'barked' and 'barked', using 'barked' in the context of a dog making a sound and 'barked' as in speaking loudly and aggressively. This creates a humorous double meaning.

Table 16: LLM misinterprets het-pun as hom-pun in explanation.

Lack of Meaning Analysis For certain common puns, especially het puns, LLM will only pinpoint the pun word and alternative words without an explanation of their respective meanings, like in Table 17.

Fabricating Non-existent Meanings After LLM wrongly identifies a pun word or alternative word, they will continue to explain the pun based on the previous incorrect conclusion. Just like the following example (Table 18) shows, after "plays on the double meaning of splice", LLM generated the nonexistent meaning of "splice" as "adding toppings to a pizza" to fit in the explanation.

Lack of Meaning Analysis
/ Pun Text */* All students in Helsinki go to Finnish school. */* Right explanation of Human */* This is a pun on 'finishing school' which is a private school where girls are prepared for entry into fashionable society, but 'Finnish' is used in its place because it sounds slightly similar and since Helsinki is the capital of Finland, any school there is a Finnish school. */* Wrong Explanation of LLM */* The text plays on the similarity in sound between 'Finnish' and 'finish', creating a humorous double meaning. It exploits the similar-sounding words for a humorous effect.

Table 17: LLM doesn't explain any meanings of pun/alternative words in explanation.

Fabricating Non-existent Meanings
/ Pun Text */* Did you hear about the film editor at Hollywood who had a part-time job at a pizza parlor? He used to ask customers whether they wanted one splice or two. */* Right explanation of Human */* "Splice" sounds like "slice" and "slice" is a word often associated with pizza. "Splice" is a verb that means to join or connect. In film, a splice is when a film has to be taped together because of breakage. This is often seen during the editing process. The joke is about a film editor's part-time job at a pizza parlor. */* Wrong Explanation of LLM */* The text plays on the double meaning of 'splice', which can refer to both editing film and adding toppings to a pizza. This creates a humorous play on words.

Table 18: LLM makes up non-existent meanings in explanation.

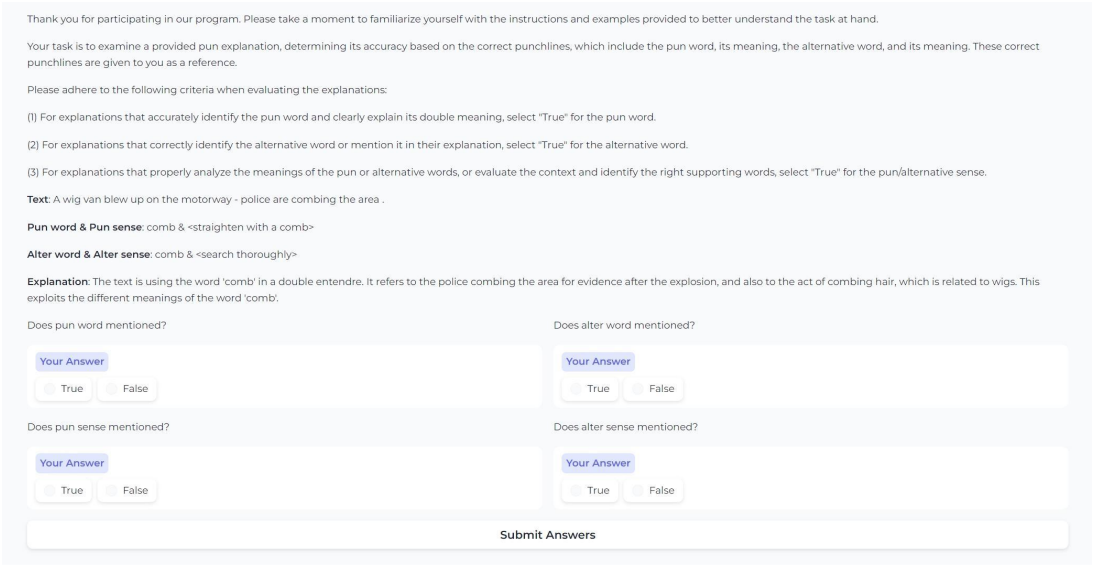

Figure 6: The screenshot of punchline check annotation.

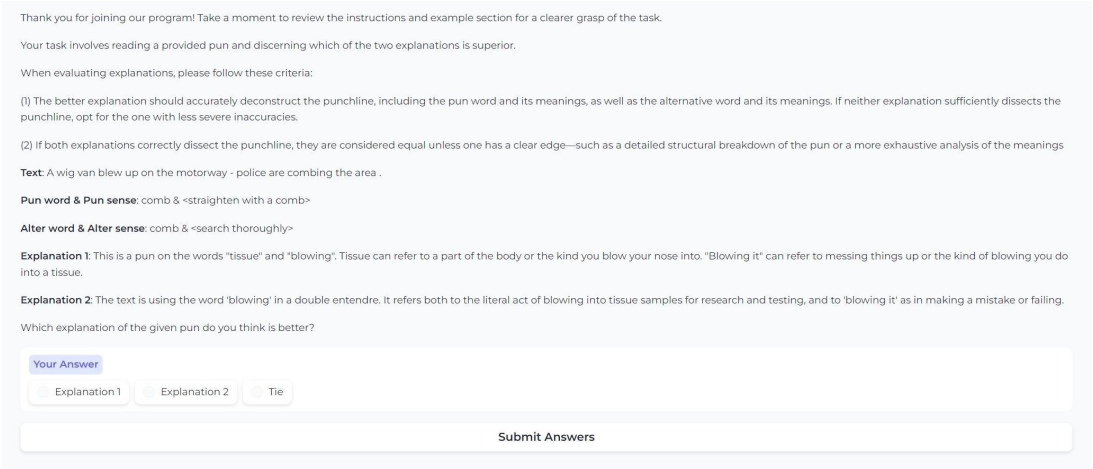

Figure 7: The screenshot of pairwise comparison annotation.

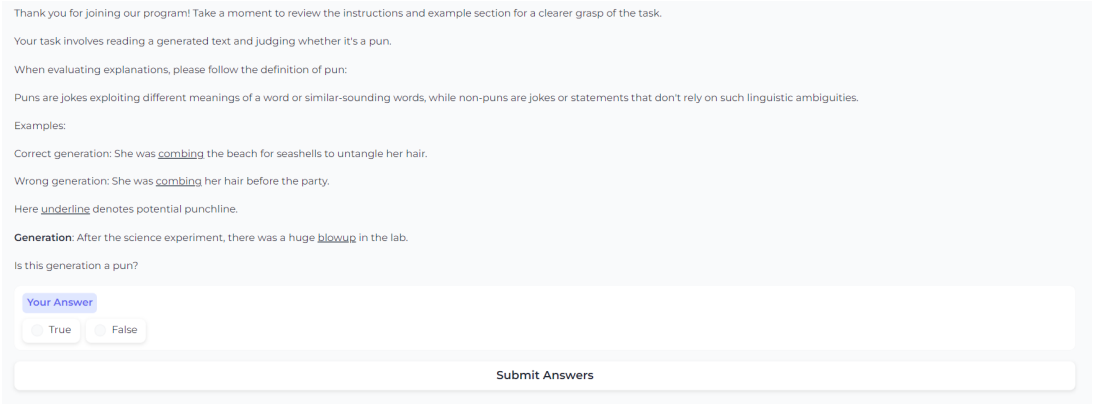

Figure 8: The screenshot of generation success annotation.