Pun Unintended: LLMs and the Illusion of Humor Understanding

Alessandro Zangari ♠ **Matteo Marcuzzo** ♠ **Andrea Albarelli** ♠
Mohammad Taher Pilehvar ♢ **Jose Camacho-Collados** ♢
♠Dept of Environmental Sciences, Informatics and Statistics, Ca' Foscari University of Venice
alessandro.zangari@unive.it, matteo.marcuzzo@unive.it, albarelli@unive.it
♢School of Computer Science and Informatics, Cardiff University
pilehvarmt@cardiff.ac.uk, camachocolladosj@cardiff.ac.uk

Abstract

Puns are a form of humorous wordplay that exploits polysemy and phonetic similarity. While LLMs have shown promise in detecting puns, we show in this paper that their understanding often remains shallow, lacking the nuanced grasp typical of human interpretation. By systematically analyzing and reformulating existing pun benchmarks, we demonstrate how subtle changes in puns are sufficient to mislead LLMs. Our contributions include comprehensive and nuanced pun detection benchmarks, human evaluation across recent LLMs, and an analysis of the robustness challenges these models face in processing puns.

1 Introduction

Understanding linguistic nuances, such as hedges, idiomatic phrases, figures of speech, and metaphors, is crucial for effective communication (Boisson et al., 2024; Qamar et al., 2025). This requires deep contextual and cultural awareness, presenting ongoing challenges for LLMs that struggle with subtle, multi-layered language (Liu et al., 2023; Ghosh and Srivastava, 2022; Zhang et al., 2024). One notable example of nuanced language is the *pun* (paronomasia), a form of wordplay generating rhetorical, often humorous, effects through polysemy and phonetic similarity. Prominent in literature, poetry, and advertising (Miller and Gurevych, 2015), puns depend on the intuitive recognition of dual meanings (literal vs. figurative), creating the characteristic *pun effect* (Brown, 1956; Attardo, 2017). The ability to automatically recognize puns is relevant for digital humanities and NLP tasks, such as sentiment analysis, and machine translation, which struggle with the ambiguity and non-literal nature of puns (Miller and Turković, 2016). In comparison to other rhetorical devices, such as sarcasm, metaphors, or jokes, puns are structurally simpler (Raskin, 2008; Miller et al., 2017), and easier to formalize (Sun et al., 2022b).

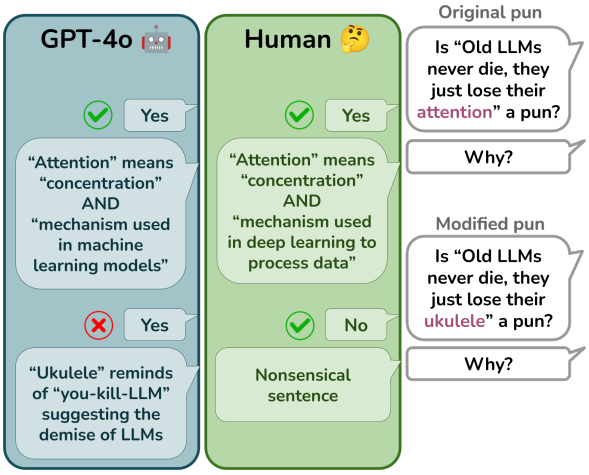

Figure 1: Example pun detection and explanation by GPT-4o vs. human. Subtle modifications to the pun (replacing the polysemous word, *attention*, with a random one, *ukulele*) are often sufficient to mislead LLMs.

These qualities make them a good candidate for assessing linguistic understanding in LLMs.

While recent studies have examined pun generation (Mittal et al., 2022; Tian et al., 2022), detection, and explanation (Sun et al., 2022a; Xu et al., 2024; Miller et al., 2017), this line of research faces two key limitations. First, most studies have relied on a single dataset from SemEval (Miller et al., 2017), where shallow cues and data leakage may inflate performance (Ermakova et al., 2023a). Second, evaluations are typically focused on binary classification or hard-to-assess free-text rationales. A systematic analysis of more structured rationales would help clarify the capabilities and limitations of LLMs in interpreting rhetorical ambiguity (Kim, 2025; Wiegreffe et al., 2021).

Motivated by these limitations, we introduce two new collections of annotated short texts, PunnyPattern and PunBreak, designed specifically to evaluate the robustness of LLMs in pun detection (see Fig. 1). These collections move beyond simple detection, using targeted substitutions and common

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 27925–27960
November 4-9, 2025

language patterns to probe whether models can accurately recognize puns' context and structure, or if they rely on memorization and superficial cues. We publicly release both datasets.[1]

We evaluate 7 open- and closed-weights LLMs on both identifying puns and providing supporting *rationales*, then analyze the quality of these rationales, and report their impact on detection performance. In contrast to earlier studies (Sun et al., 2022a; Xu et al., 2024), we design prompts that elicit semi-structured rationales, enabling more systematic evaluation via automatic metrics and detailed manual analysis guided by a new annotation protocol. More specifically, we organize our discourse around the following research questions:

- RQ1: How well can LLMs detect puns on new and existing datasets?
- RQ2: How robust are LLMs at detecting puns?
- RQ3: To what extent can LLMs explain puns?

Our results show that most LLMs superficially associate puns with common language patterns, which makes it difficult for them to distinguish genuine puns from structurally similar sentences that contain no pun. Automatic and manual error analyses further reveal that the models struggle to handle context and phonological properties effectively.

2 Related Work

SemEval-2017 Task 7 (Miller et al., 2017) targeted pun detection, location, and interpretation tasks by releasing a dataset of annotated puns and non-puns. The participating systems were mostly based on traditional NLP techniques, with leading systems employing heuristics based on positional and semantic features, such as n-grams enhanced by Word2Vec embeddings (Mikolov et al., 2013) and phonetic distance measurements. The reported results showed strong performance in binary detection, moderate in pun location, but notably weak in correct sense association. SemEval-2021 (Meaney et al., 2021) later expanded this dataset by adding new jokes, although without additional annotations.

Building upon this foundation, the JOKER workshop (Ermakova et al., 2023b) introduced multilingual datasets for pun detection, location, interpretation, and translation, encompassing English, French, and Spanish. The JOKER corpus (Ermakova et al., 2023a) aimed to address earlier criticisms of imbalance and reliance on superficial cues by providing 3,506 annotated short jokes in English and French, alongside 1,700 negative examples generated via word substitutions. Despite utilizing advanced LLMs such as GPT-3 (Brown et al., 2020) and BLOOM (Workshop et al., 2023), a custom-trained T5 model (Raffel et al., 2020) consistently outperformed others on all tasks, albeit with limited success (detection F1 < 0.6). Other studies have further explored non-English puns (Gameiro et al., 2025; Chen et al., 2023; Dsilva, 2024).

Complementing these efforts, Sun et al. (2022a) investigated pun explanation by enhancing training data with rationales generated by T5 models. They measured explanation quality using simulatability, defined as the accuracy difference when explanations are provided as additional input vs. when they are not. They found that high-quality human-annotated rationales can improve model performance, but automatically generating such explanations with language models remains challenging. In a related work, Sun et al. (2022b) focus on pun generation by retrieving context and pun words. They propose a system that first retrieves a suitable pair of words, along with their senses, from given contextual words, and then employs a T5 model to generate a pun. However, the success rate of the generated puns was reported to be significantly lower than that achieved by humans.

To the best of our knowledge, Xu et al. (2024) is the only study to evaluate recent LLMs on pun detection, explanation, and generation. They refined the SemEval dataset by removing duplicates and adding human-annotated explanations. While the study included a manual evaluation of a small set of generated explanations, it lacks a comprehensive error analysis and detailed evaluation guidelines. Their results indicate that, although LLMs perform reasonably well on binary pun detection, they struggle with pun generation and explanation, often relying on memorization or biased shortcuts. Building on this work, our study conducts a robustness analysis targeting specific biases and examines common errors in model-generated rationales.

3 Experimental Methodology

The primary pun-related tasks previously discussed in the NLP literature relate to detection, location, and interpretation. *Pun detection* is a binary classification task, categorizing texts as either containing a pun or not. *Pun location* involves identifying the specific pair of words responsible

[1] https://github.com/alezanga/punintended

for creating the double entendre, while *pun interpretation* requires associating each identified pun word with its correct meaning or sense (Ermakova et al., 2023b; Jain et al., 2019).

Preliminaries. Structurally, a pun is composed of a *pun word* (w_p), an *alternative word* (w_a) and their respective *senses* s_p and s_a (Sun et al., 2022b). For brevity, we will refer to (w_p, w_a) as the *pun pair*. Following previous literature (Miller et al., 2017; Xu et al., 2024), we focus on *heterographic* and *homographic* puns.

In *heterographic* puns (het-puns), only w_p appears in the sentence, while w_a and its sense are evoked through the context. For example, the het-pun "*I bought a boat because it was for sail (sale)*" creates a double meaning by playing on the homophones *sail* (to navigate on a boat) and *sale* (an occasion when items are sold). In contrast, *homographic* puns (hom-puns) feature a polysemous w_p, resulting in $w_p = w_a$. The hom-pun "*I was wondering why the ball was getting bigger; then it hit (hit) me*" plays on the two senses of *hit* (to be physically struck) and *hit* (to suddenly realize).

The term *rationale* has been used in the literature to denote various types of evidence justifying a decision, particularly in the form of natural language explanations (Herrewijnen et al., 2024). This is consistent with its usage in prior research on puns (Xu et al., 2024; Sun et al., 2022a), although we do not limit it to unstructured or free-text explanations.

3.1 Datasets

Existing collections of puns annotated with the structure just described are limited. The SemEval 2017 dataset (Miller et al., 2017) provides 4,030 samples encompassing heterographic and homographic puns. Later research (Sun et al., 2022a; Xu et al., 2024) refined this dataset by adding sentiment annotations and removing incorrect examples. The English part of the JOKER corpus (Ermakova et al., 2023a,b) extends SemEval and adds 632 new English puns collected from `PunOfTheDay.com`. By removing or replacing a single contextual word, these puns have been altered to create an equal number of non-puns. Both datasets contain pun words and senses annotations.

We utilize the dataset proposed by Xu et al. (2024), which is the most refined iteration of the SemEval 2017 dataset, along with a subset of the JOKER dataset. In the former, we corrected several typographical errors and 13 incorrect annotations, resulting in 2,589 samples. We split the dataset into training (1,071 samples), testing (1,341), and validation (177) sets, ensuring that no w_p and w_a in the training set appear in the test and validation sets. We refer to this as the *PunEval* dataset. Unlike the SemEval dataset, the full *JOKER* corpus is not publicly available and was released only as part of the JOKER CLEF workshop (Ermakova et al., 2023b). However, the authors kindly agreed to share the complete dataset with us upon request. Since most examples in this dataset originate from the SemEval dataset, we retained the subset of 632 puns and 632 non-puns created by replacing a single word in the original pun. Statistics for all datasets are in Appendix §B.

3.2 Comparison models

We benchmark five recent instruction-tuned LLMs on pun understanding, including both open- and closed-weights models: GPT-4o-2024-08-06 (OpenAI) (OpenAI et al., 2024), Qwen2.5-72B (Alibaba) (Yang et al., 2025), Llama3.3-70B (Meta) (Grattafiori et al., 2024), Gemini2.0-Flash (Google DeepMind) (Anil et al., 2025), and Mistral3-24B (Mistral AI) (Jiang et al., 2023). We refer to these models as follows: GPT-4o, Qwen2.5, Llama3.3, Gemini2.0, and Mistral3. Additionally, we employed two reasoning models from the DeepSeek family: DeepSeek-R1 (R1) and DeepSeek-R1-Distill-Llama-70B (R1-D) (DeepSeek-AI et al., 2025), the latter being a distilled version of the full R1, based on Llama-3.3-70B-Instruct.

3.3 Prompts

Unlike in Xu et al. (2024), in this work we leverage the pun structure described in §3 and instruct the model to respond in a semi-structured format, facilitating parsing and the evaluation of the rationales.

We adopt the following response format that is compatible with the rationale: `(yes|no) [<`w_p`> <`w_a`> [<`s_p`> <`s_a`>]]`. In the configurations without rationales, the answer would simply be *yes* or *no* for the pun detection task. The content within the angular brackets represents the rationale (or justification) for the answer. Inspired by the success of CoT prompting (Kojima et al., 2022; Wei et al., 2022), we explore an alternative prompting strategy to elicit reasoning from non-reasoning LLMs by asking them to *think before answering*. Unlike previous cases where responses were restricted to a structured format, the reasoning prompts allow the model to

provide free-text reasoning before delivering the final answer in the same structured format. In what follows, we describe the prompt configurations used. Validation experiments with alternative prompts, along with the exact prompts we adopted, are detailed in Appendix §A.1.

Zero-Shot. (0S) This prompt asks to provide a "Yes" or "No" answer on whether the text contains a pun, with no formal definition of a pun.
Few-Shot. (FS) A definition of pun is provided, including the concepts of w_p, w_a, s_p, and s_a, followed by six examples (three puns, three non-puns) drawn from the same set used by Xu et al. (2024). Examples are held constant across experiments.
Words. (W) Same as *Few-Shot* prompt, but requires reporting the pun pair as justification.
Words+Senses. (W+S) This prompt mirrors the *Words* prompt, but additionally requires reporting s_p and s_a after the pun pair.
Reasoning. (R+) This configuration uses the same prompts, but allows the generation of arbitrary text before the final answer. This prompt is applied only to the five non-reasoning LLMs.

4 RQ1: Can LLMs Detect Puns?

The first research question investigates LLMs' overall performance in binary pun detection. We thus evaluate all comparison LLMs described in §3.2 using the prompting strategies outlined in §3.3.

4.1 Data

In addition to the PunEval and JOKER datasets described in §3.1, we annotate a new collection of 128 puns from various sources, including websites[2], personal recollections, and original creations. The main reason for creating this new dataset was to avoid any potential contamination from existing datasets. Each pun is rephrased into a non-pun, similar to the JOKER dataset, but without the requirement of replacing a single word, resulting in a total of 256 instances. We use Mistral3 for this rephrasing process, and all generated samples are manually inspected and modified as needed. We refer to this as the *Newly Annotated Puns* (NAP) dataset. A sample pun and its corresponding non-pun are shown in Example 1, while general NAP statistics are included in Appendix §B. Finally, we also annotate each pun with its pun pair (w_p, w_a) and respective senses (s_p, s_a), using short definitions from online dictionaries[3]. This process is consistent with the methods employed in the SemEval and PunEval datasets, and these annotations will be used to address RQ3.

Original Pun
What did the buffalo say to his son? Bison (*Bye son*).
Rephrased pun
What did the buffalo ask his son? I do not know.

Example 1: A pun and non-pun from the NAP dataset.

4.2 Results

The average F1-score for these datasets across 3 runs is shown in Table 1. Additional results can be found in the Appendix (see Table 7 and Fig. 9). All LLMs achieved F1-scores around 0.8 on the NAP dataset with the best prompting strategy. GPT-4o performed the best, while Mistral had the lowest scores. Notably, Mistral only performed well in the zero-shot setting and struggled to learn from additional context in the prompt.

Results on the JOKER dataset show a similar trend but are slightly more challenging for all models. This is likely due to its method of creating non-puns by replacing a single word, resulting in more "adversarial" examples. In contrast, in the NAP dataset, we rephrased puns into non-puns, potentially replacing or removing multiple words.

Aside from Mistral, which consistently performs poorly on our new datasets, the W/W+S prompts — which require LLMs to justify each decision with a rationale — achieve the highest performance, with an average improvement of +3% in F1-score compared to the few-shot configuration (more details in Appendix §C). This finding aligns with previous research (Sun et al., 2022a) that shows explanations can enhance pun detection performance. However, the reasoning prompt did not yield significant improvements over the rationale-augmented prompts (W and W+S) on the NAP and JOKER datasets. Overall, the strong performance suggests that LLMs can understand puns to some extent and benefit from jointly detecting and explaining.

Embedding Space Analysis. In addition to the comparison LLMs, we also include a RoBERTa-large encoder model (Liu et al., 2019) that has been

[2] https://parade.com/1024249/marynliles/funny-puns

[3] https://dictionary.cambridge.org/dictionary, https://www.merriam-webster.com

M	Prompt	F1-score (%)		
		NAP	JOKER	PunEval
Gemini2.0	0S	73.3 ± 0.0	71.2 ± 0.0	85.7 ± 0.2
	FS	74.9 ± 0.0	74.0 ± 0.1	89.3 ± 0.1
	W	76.2 ± 0.4	74.4 ± 0.1	88.3 ± 0.1
	W+S	**77.9** ± 0.3	74.4 ± 0.0	89.1 ± 0.2
	R+FS	70.9 ± 0.6	71.1 ± 0.2	82.7 ± 0.3
	R+W	76.6 ± 0.0	74.4 ± 0.0	**90.6** ± 0.1
	R+W+S	75.4 ± 0.2	**74.6** ± 0.1	89.5 ± 0.1
GPT-4o	0S	78.4 ± 0.1	77.2 ± 0.0	89.7 ± 0.1
	FS	81.0 ± 0.7	79.7 ± 0.1	92.8 ± 0.2
	W	**86.9** ± 0.8	**83.3** ± 0.4	87.3 ± 0.9
	W+S	85.0 ± 0.8	82.6 ± 0.1	88.9 ± 0.9
	R+FS	82.9 ± 0.0	80.5 ± 0.3	92.3 ± 0.1
	R+W	84.0 ± 1.1	81.1 ± 0.2	92.6 ± 0.4
	R+W+S	82.9 ± 0.8	81.3 ± 0.3	**93.0** ± 0.3
Llama3.3 (70B)	0S	79.6 ± 1.5	77.1 ± 0.8	84.0 ± 0.4
	FS	81.0 ± 0.7	77.6 ± 0.1	84.4 ± 0.4
	W	83.4 ± 0.9	**79.2** ± 0.2	**87.2** ± 0.2
	W+S	**83.6** ± 0.9	77.9 ± 0.6	86.4 ± 0.3
	R+FS	79.2 ± 0.2	75.2 ± 0.1	83.8 ± 0.2
	R+W	78.1 ± 0.8	76.4 ± 0.1	85.1 ± 0.5
	R+W+S	78.3 ± 1.0	76.8 ± 0.2	84.5 ± 0.3
Mistral3 (24B)	0S	**75.0** ± 0.5	**75.3** ± 0.2	80.8 ± 0.2
	FS	69.5 ± 2.2	66.2 ± 1.2	74.6 ± 1.9
	W	69.0 ± 1.7	69.3 ± 0.4	78.4 ± 0.5
	W+S	68.5 ± 0.9	68.5 ± 0.4	74.9 ± 0.9
	R+FS	73.6 ± 1.1	70.9 ± 0.1	82.4 ± 1.3
	R+W	70.7 ± 0.3	69.0 ± 0.4	84.2 ± 0.4
	R+W+S	70.9 ± 0.2	70.1 ± 0.3	**84.5** ± 0.2
Qwen2.5 (72B)	0S	71.3 ± 0.4	73.6 ± 0.2	83.6 ± 0.1
	FS	78.6 ± 0.4	76.4 ± 0.7	87.2 ± 0.2
	W	**81.1** ± 0.0	77.1 ± 0.4	86.8 ± 0.4
	W+S	80.7 ± 0.6	76.8 ± 0.8	87.3 ± 0.1
	R+FS	80.2 ± 0.7	**77.5** ± 0.6	88.5 ± 0.6
	R+W	77.2 ± 0.2	75.0 ± 0.1	**89.4** ± 0.5
	R+W+S	77.0 ± 1.1	75.8 ± 0.6	88.7 ± 0.1
R1-D (70B)	0S	74.8 ± 0.5	75.3 ± 0.6	86.7 ± 0.3
	FS	76.3 ± 2.7	76.2 ± 0.5	87.0 ± 0.3
	W	**79.8** ± 1.4	78.1 ± 0.9	86.8 ± 0.4
	W+S	78.7 ± 1.2	**78.7** ± 0.6	**88.3** ± 0.6
R1 (671B)	0S	74.2 ± 0.4	75.5 ± 0.5	88.5 ± 0.2
	FS	76.6 ± 0.5	78.4 ± 0.4	90.8 ± 0.2
	W	77.7 ± 0.4	77.7 ± 0.1	90.9 ± 0.1
	W+S	**79.1** ± 0.8	**79.2** ± 0.0	**91.3** ± 0.4
RoBERTa-L		64.0 ± 0.1	65.7 ± 0.1	92.5 ± 0.3

Table 1: Pun detection F1-score ± STD over 3 runs. Top results per model are in bold.

fine-tuned on the PunEval training split. The significant performance difference observed between PunEval (92.5) and the other datasets (64.0 in NAP and 65.7 in JOKER) prompted an investigation to understand the reasons behind this disparity. We analyzed the embedding distribution of the RoBERTa model using t-SNE (van der Maaten and Hinton, 2008) to visualize the embeddings in a 2D space (Fig. 2, see details in Appendix §A). The visualizations reveal poor separation between subtly altered positive and negative examples in the NAP and JOKER datasets (Fig. 2, right). In contrast, PunEval embeddings show better separation even before fine-tuning (Fig. 2, center), with clusters often corresponding to recurring templates. For instance, the top-left cluster from the middle image represents puns structured as "Old [...] never die, they just", a common pattern for creating puns (e.g., "*Old bankers never die, they just lose interest*"). This suggests that dataset artifacts or structural similarities in PunEval may facilitate shortcut learning.

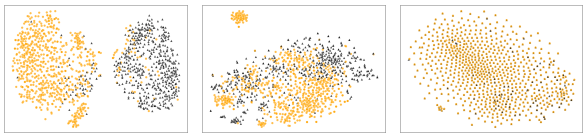

Figure 2: Separation of PunEval embeddings for the RoBERTa model: after fine-tuning (left), before fine-tuning (middle), and NAP+ JOKER separation before fine-tuning (right). Orange dots represent puns, while black triangles represent non-puns. Notably, PunEval puns exhibit greater separability in the embedding space compared to NAP/JOKER, even before fine-tuning.

5 RQ2: How Robust Are LLMs at Detecting Puns?

Given the analysis in the previous section, one may wonder whether the performance gap between the PunEval dataset and the other two datasets is partly due to LLMs recognizing superficial cues or overfitting to specific pun structures without fully understanding their meanings. This observation prompts us to investigate the LLMs' robustness to common pun language patterns and simple pun alterations.

5.1 Pun Language Patterns

The PunEval dataset contains language patterns frequently used to create puns. Examples include "*Old [...] never die, they just [...]*" and "*Tom*" (a common name in English jokes), which may serve as shortcuts for recognizing puns.

To analyze the impact of these recurrent patterns on model performance, we extracted bag-of-n-gram features and trained a logistic regressor for pun detection on the PunEval training set. By examining the top-20 expressions based on learned coefficients and frequency, we manually identified six patterns that significantly correlate with the presence of a pun. These patterns, along with ex-

Original pun
I used to be a comedian, but my life became a joke (*joke*).
Rephrased pun
I used to be a comedian, but my life became chaotic.

Example 2: A pun with a common pun language pattern and a derived non-pun.

amples and their frequencies, are listed in Tables 8 and 9 in the Appendix. While their presence has been noted previously (Ermakova et al., 2023a), no detailed analysis of their effects has been provided. In the PunEval test split, 171 out of 173 samples that contain these patterns are puns. Excluding them results in a 2-3% drop in F1-score and a 2-6% drop in precision across all models compared to the full test split, confirming that the presence of these recognizable patterns can inflate performance. To further investigate this issue, we created a new benchmark to test robustness against this bias.

5.1.1 Data

For each of the six extracted patterns, we sampled 100 puns from the PunEval test split that contain the same expression, supplementing with new examples collected from the Internet or generated using GPT-4o (verified manually) as needed to complete the set. More details on this collection are reported in Appendix §B.1. We also generated an equal number of verified non-pun sentences using GPT-4o (with human oversight), ensuring they contain the same expression. The resulting dataset, referred to as the *PunnyPattern* dataset, contains 1,200 samples, two of which are presented in Example 2.

5.1.2 Results

Table 2 compares the average performance metrics over 3 runs between the entire PunnyPattern dataset and the PunEval dataset. More detailed results for each language pattern can be found in the Appendix (see Table 10 and Fig. 11). Overall, there is an average drop of 16-23% in precision and 4-13% in F1 on the PunnyPattern dataset across all models. Additionally, we observe a sharp imbalance of low precision and high recall for some of the patterns (see Fig. 11), which indicates LLMs are prone to identifying puns whenever they observe a typical pun-like pattern. These results suggest that most LLMs tend to process certain patterns somewhat superficially, lacking understanding of the underlying principles of well-crafted puns.

Model	**F1**	**Precision**	**Recall**
Gemini2.0	76.9 (-12.2)	66.7 (-20.9)	91.6 (+0.9)
GPT-4o	**83.1** (-4.2)	**79.7** (-18.2)	88.1 (+9.4)
Llama3.3	81.0 (-6.2)	71.5 (-16.4)	94.2 (+7.7)
Mistral3	77.4 (-3.4)	72.1 (-14.8)	87.5 (+11.9)
Qwen2.5	79.7 (-7.1)	74.1 (-19.3)	87.5 (+6.5)
R1-D	78.3 (-10.0)	66.9 (-18.7)	94.8 (+3.7)
R1	81.1 (-10.2)	69.5 (-15.9)	**98.3** (+0.3)

Table 2: F1-score, precision, and recall (%) on the best prompt on PunnyPattern and the absolute performance difference with the PunEval dataset.

5.2 Pun Alterations

Motivated by our previous results, we investigate whether LLMs exhibit robustness to simple alterations of puns designed to ruin them. For humans, the pun effect relies on the perception of two well-supported senses that create ambiguity and contrast. Therefore, replacing the pun word (w_p) with a lexical unit that fails to convey this duplicity would result in the loss of such effect, and the failure of the intended pun.

5.2.1 Data

To systematically investigate LLMs' understanding of this phenomenon, we replace the pun word w_p with multiple alternative expressions and measure their ability to classify "ruined" puns. To this end, we randomly select 100 heterographic and 100 homographic puns from the NAP and PunEval datasets and modify each pun with four different substitutions:

- `Pun syn`: a synonym or hypernym of the pun word w_p, different in phonetics and orthography.
- `Alt syn`: as above, but for the alternative word w_a.
- `Homophone`: a nonsensical expression phonetically similar to w_p.
- `Random`: a nonsensical random word.

Additionally, we include 100 randomly generated sentences (non-puns) as a control group, resulting in a total of 1,100 examples, which we refer to as the *PunBreak* dataset. A complete set of substitutions is shown in Example 3, and more details on the generation procedure are in Appendix §B.2. Note that while some non-sensical replacements may be humorous due to the absurdity of the resulting sentences, they do not represent valid puns.

Original pun
Long fairy tales have a tendency to dragon (*drag on*).
`Pun syn` substitution
Long fairy tales have a tendency to wyvern.
`Alt syn` substitution
Long fairy tales have a tendency to prolong.
`Homophone` substitution
Long fairy tales have a tendency to brag gone.
`Random` substitution
Long fairy tales have a tendency to tick.

Example 3: Example substitutions from the PunBreak.

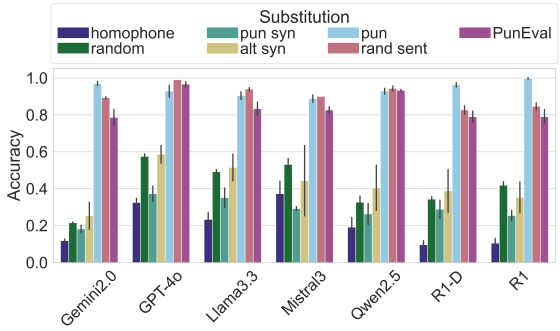

Figure 3: Binary accuracy of comparison LLMs with their best prompt on the *PunBreak* dataset, with error bars representing the standard deviation over 3 runs.

5.2.2 Results

The binary accuracy of each LLM for every substitution subset, using each model's best prompt from the RQ1 experiments, is shown in Fig. 3. The "pun" subset contains exactly 200 puns (100 het-puns and 100 hom-puns); every other subset contains exactly 200 non-puns representing the various substitutions. Results from single prompts are reported in Appendix §C.4. The "rand sent" results are obtained from a control of 100 randomly generated non-puns to detect any unseen bias toward the pun class; scores exceed 0.8, indicating no such bias in the random sentences.

All models exhibit a bias toward labeling altered examples as puns, reflected in low accuracy across all substitution categories, which contain only negative examples (non-puns). In particular, GPT-4o, the best performer overall, reaches a low of 0.33 on the `homophone` subset and a high of 0.59 on `alt syn`. Conversely, all models achieve accuracy of around 0.8 or higher on the negative examples from PunEval and the control group (`rand sent`) (see Table 11 for details). The `homophone` subset is the most challenging overall, indicating that phonetic similarity between w_p and its replacement degrades the LLMs' ability to distinguish them in context. This may also be influenced by orthographic similarity, as similar phonetics often imply similar orthography (e.g., *stock*/*stork* or *dragon*/*brag gone* in Example 3). Pun-word synonym substitutions (`pun syn`) also generally challenge the models, suggesting they may overlook the effect of the replaced word when it is semantically similar to a more fitting alternative. This suggests that LLMs might be actively "fixing" the pun in their reasoning process, thereby rectifying the alteration. The `random` and `alt syn` sets show slightly higher accuracy overall, further indicating that semantic or phonetic similarity to valid puns misleads predictions more than loosely related substitutions. These results suggest that LLMs fail to recognize contexts that should not trigger the *pun effect* in cases where the target sentence has a typical pun structure.

6 RQ3: To What Extent can LLMs Explain Puns?

Our rationale-augmented prompts achieve the best performance in our previous experiments, notably reducing bias on the PunnyPattern dataset with a 9.8% increase in precision (see §C.1 in the Appendix for details on rationale impact). In this section, we systematically examine the quality of the generated rationales to gain insights into LLMs' understanding of puns.

6.1 Keyword Evaluation

Our goal is to measure the overlap between human-provided semi-structured rationales and LLM-generated rationales, while also accounting for prediction accuracy. To this end, we quantify the alignment between the *LLM-predicted pun pair* (w'_p, w'_a) and the true pun/alternative word pair (w_p, w_a) using the *pun pair agreement* (PPA) metric. Each prediction is scored: 2 if both w_p and w_a are correctly generated, 1 if only one is correct, and 0 if neither is correct. Predictions with incorrect labels receive a score of 0, since no rationales are generated for non-pun predictions. For further details on the evaluation process, see §A.2.

Results. Table 3 reports the average PPA over three runs, representing the mean number of words correctly identified in generated rationales for pun pairs on a 0–2 scale. GPT-4o is the top performer, with ∼1.5 out of 2 correctly predicted words per

Model	NAP		JOKER		PunEval	
	W	W+S	W	W+S	W	W+S
Gemini2.0	1.2	1.2	1.1	1.1	1.5	1.5
GPT-4o	**1.5**	**1.5**	**1.5**	**1.4**	**1.6**	1.6
Llama3.3	1.4	1.3	1.3	1.2	1.5	1.5
Mistral3	0.8	0.7	0.8	0.7	1.2	0.9
Qwen2.5	1.2	1.2	1.2	1.2	1.5	1.5
R1-D	1.3	1.2	1.2	1.2	1.5	1.6
R1	1.3	1.2	1.2	1.3	**1.6**	**1.7**

Table 3: Average Pun Pair Agreement (PPA), in [0–2], for each prompt setting (W, W+S) across datasets. Standard deviation is 0.0 for all measurements, and the best results are marked in bold.

pun across all three datasets. It is followed by Llama3.3 and DeepSeek-R1, whose performance is noticeably lower. Mistral has the lowest scores, trailing the next-lowest model, Gemini2, by ∼33%. Among the 70B models, Llama3.3 performs best on average. We also observed no significant difference between the two prompts tested. Consistent with our earlier results, scores on the PunEval datasets are generally higher than on JOKER and NAP, where performance drops by approximately 20% and peaks at 1.5 out of 2. Furthermore, a PPA evaluation restricted to true positives (see Appendix Table 12) shows that most models, except Qwen and Mistral, can effectively justify their correct answers. In this specific evaluation, GPT-4o and DeepSeek-R1 emerge as the top performers by a wide margin. These findings motivate our investigation of the main error types in model reasoning, which is addressed in the next subsection.

6.2 Error Analysis: Manual Evaluation

The PPA metric measures the ability to match the original pun pair but does not indicate *why* predictions were wrong, nor does it evaluate the complete rationale obtained with the more expressive W+S prompt. Upon inspection of the produced rationales, we identify four main categories of mistakes frequently made by our LLMs regarding the predicted pun pair (w'_p, w'_a) and the associated pair of senses (s'_p, s'_a):

- **Word-sense association** (`word-sense` error): incorrect association between word and sense (i.e., w'_p–s'_p or w'_a–s'_a).
- **Missing context** (`context`): the interpretation of w'_p as s'_p or w'_a as s'_a is not supported by the context.
- **Pun pair** (`pun pair`): w'_p and w'_a are not sufficiently similar in phonetic or orthographic terms to create wordplay (this only applies to het-puns).
- **Senses similarity** (`sense sim`): senses s'_p and s'_a are too similar. Puns work when polysemous words carry well-separated senses that create contrast.

These types of mistakes also highlight different weaknesses. A word-sense mistake can be considered a hallucination (e.g., "crustacean" interpreted as "a species of bird") while a pun pair mistake (as in Fig. 1) indicates difficulty in grasping the phonological properties of words.

To precisely characterize the model's mistakes in the categories above, we conducted an error analysis with five native English speakers specifically hired for this task. We randomly sampled 80 incorrect answers from each of the three top-performing LLMs, according to the PPA evaluation: GPT-4o, Llama3.3, and DeepSeek-R1. For each of the sampled answers, we designed a set of questions to determine the presence of each category of mistake. For example, the first two questions assess whether s'_p (the LLM-predicted pun sense) and s'_a (the predicted alternative sense) are legitimate interpretations of w'_p (predicted pun word) and w'_a (predicted alternative word), respectively. Annotators could answer with "Yes", "No", or "Maybe" if uncertain. If any of these questions is answered with "No" for a given sample, that sample is counted as a `word-sense` association error. Note that if both senses are incorrect, it still counts as a single error, ensuring a more comparable error count across all mistake categories. The full evaluation procedure, guidelines, and questionnaire are detailed in §A.3 of the Appendix. Table 13 shows the error-analysis results for six example samples.

Error Breakdown by Category			
Error Category	**GPT-4o**	**Llama3.3**	**R1**
`context`	66	**49**	50
`pun pair`	50	39	**27**
`word-sense`	9	17	**8**
`sense sim`	**2**	3	**2**
Aggregated statistics			
Total errors	128	111	**87**
Avg. errors	1.6	1.4	**1.2**

Table 4: Error statistics by category and overall on 80 misclassified samples from the PunBreak dataset. Note that there may be more than one error per sample.

Results. Table 4 reports the results of the manual evaluation for the three analyzed LLMs. Our analysis reveals that, in general, the most frequent errors involve: (i) missing the required context to support both s_p and s_a (`context`) and (ii) selecting unsuitable word pairs that do not fit into a pun (`pun pair`). The latter category only applies when $w'_a! = w'_p$, and typically results from forcing words that are not phonetically compatible to create wordplay. A third fairly common mistake arises from incorrectly pairing words with their meanings (`word-sense`). Llama3.3, the smallest among the three LLMs, tends to struggle the most in this area, producing more sense hallucinations. In contrast, GPT-4o and R1 tend to assess the senses correctly, even when some words in the pun pair are incorrectly identified. These error patterns suggest a lack of understanding of the mechanisms of puns, particularly the inability to distinguish inappropriate contexts and difficulties in handling wordplay based on phonetic or orthographic similarity. Among the three evaluated models, DeepSeek-R1 proved to be the most precise, exhibiting the lowest number of errors in its responses.

6.3 Discussion

Our findings align with prior work showing that LLMs struggle with humor, particularly when it depends on nuanced context or roleplay (Quan et al., 2025; Mirowski et al., 2024). These studies note that safety constraints (e.g., the "Harmless, Helpful, Honest" criteria) can cause LLMs to misinterpret relational context in potentially offensive situations. These safety mechanisms bias models toward a safer, "washed-out" form of humor, stripping away the surprise and edge on which many jokes depend. Puns, in contrast, are rich in surprise, subtlety, and timing; they are also deeply rooted in human emotions and experiences, qualities that current LLMs cannot fully tap into.

LLMs also exhibit a form of "regressive sycophancy" — a tendency to agree with a user prompt even when incorrect — particularly in ambiguous or sensitive contexts (Malmqvist, 2025; Cheng et al., 2025). We suspect that this tendency causes the models to produce false positives by forcing inputs into a "pun" template. Such behavior is a known artifact of human-preference alignment (post-training), which can bias models toward agreeable responses at the expense of contextual and factual accuracy (Sharma et al., 2025).

Finally, the frequent `pun pair` errors point to weaknesses in assessing phonetic and orthographic similarity. These issues are likely tied to tokenization, which can mask morphological elements critical to wordplay, and to limited phonological modeling (Liao and Shi, 2025). The composition of pre-training corpora and data cleaning practices may also deprive models of the informal, creative language where puns are common. Because developers rarely disclose detailed preprocessing and corpus information, it is difficult to determine how these practices affect fluency with such language.

Improving an AI's grasp of puns and humor in general will likely require deeper human-machine collaboration and greater social awareness. Prior work suggests that human alignment of LLMs should shift from a single global standard to a more granular, audience-aware approach (Quan et al., 2025; Mirowski et al., 2024). This would help models to adapt to specific audiences and contexts, adjusting appropriateness and fairness criteria accordingly.

7 Conclusion

Results in pun detection benchmarks often show a high performance by current LLMs. However, it remains unclear whether this high performance comes from true understanding or other data-specific factors. To assess robustness, we challenged state-of-the-art LLMs with two new annotated datasets that we publicly released for future research: PunnyPattern, which collects examples with language patterns common in English puns, and PunBreak, containing sentences subtly modified to mimic puns. Although LLMs can detect puns on existing benchmarks (> 0.83 average accuracy), accuracy drops substantially on our more challenging datasets: -15% on PunnyPattern, and -50% on PunBreak, indicating limited understanding of pun mechanisms and over-reliance on similarity with known puns. We also used LLMs to explain puns by generating semi-structured rationales and evaluated those explanations with automatic and manual assessments. Only three models correctly identified the pun-related words in more than 70% of cases. Moreover, they all struggled to recognize when contexts and word choice legitimately supported wordplay. This research underscores the need for more rigorous evaluations of LLMs' performance on ambiguous-language tasks, particularly those requiring subtle language understanding, such as pun detection and generation.

Limitations

A limitation of our work is that, despite our efforts to create high-quality data, the newly contributed datasets have not been comprehensively validated by language experts. Our manual error analysis indicates the possibility of incorrect annotations or "accidental" puns, stemming from the inherently subjective nature of pun perception. Discussions with annotators also revealed that the definition of a pun should be more formally characterized; we therefore provided many examples before annotation and corrected some samples based on their feedback. Despite these efforts, our manual error analysis remains limited by the subjectivity of several questions.

Additionally, there is the possibility that most LLMs have previously encountered many examples from our datasets, as puns often exist in various versions and can be easily found online. Consequently, we may not have fully disentangled this effect from our bias measurements on the PunBreak and, in particular, the PunnyPattern dataset, the latter of which includes many examples sourced from PunEval. We believe that further tests — such as injecting the identified language patterns into arbitrary sentences — could help clarify the impact of model bias on performance.

Future work should investigate the severity and correctability of these biases. As reviewers suggested, it remains to be tested whether in-context learning with targeted negative examples, such as those from our new datasets, can mitigate the models' tendencies to classify most examples as puns. Moreover, the fact that our evaluation was conducted solely on English puns further limits the scope of the conclusions that can be drawn from these results.

Finally, our prompt design was tailored for Llama 3.1, and we utilized the same prompt for all LLMs. We acknowledge that custom-tailored prompts for each model are likely to impact performance. During this work, several new LLMs were published, and while we tested some of them, we did not have the opportunity to extensively optimize prompts for each one.

Ethics Statement

All puns used in this study were either sourced from previous work publicly released for research purposes, devised by the authors, generated using LLMs, or obtained from other open sources. In all instances, proper citations are provided to acknowledge the original authors. All collected puns are used solely for research purposes, and we do not claim authorship over any of them. As per the authors' request, the JOKER dataset was not released and was used exclusively for this study.

The pun datasets do not contain any personal information and do not refer to specific individuals; however, some may be considered offensive by certain audiences. While we have limited the number of offensive puns in our new datasets, such content may still exist in other datasets utilized in this work.

Human judges were hired for their participation in the evaluation process. Each annotator responded to 7-9 questions, which included multiple-choice answers and a Likert scale for two questions. Annotators had the option to select a specific answer or skip questions if they did not possess sufficient knowledge to respond. They were compensated 15-20£ per hour, which is well above the minimum wage in the country where the annotations took place. The annotators, who were students from diverse backgrounds — including Computer Science, Theology, and Literature — were interviewed by the authors in a virtual meeting following the initial pilot study.

Acknowledgments

We thank Felix Chadwick-Smith, Fayeja Farhana, Dhruvil Raval, Kayleigh Shier, and Hannah Trotman for their work as annotators on the manual error analysis of rationales. We also thank the entire Cardiff NLP group for their guidance on the project and for feedback on an early draft of the paper. Finally, we are grateful to Dr. Tristan Miller and co-authors for kindly allowing us to use the JOKER dataset in our evaluation.

This study was funded by the European Union - NextGenerationEU, in the framework of the iNEST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS_00000043 – CUP H43C22000540006). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.

Jose Camacho-Collados was supported by a UKRI Future Leaders Fellowship.

References

Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, and Anja Hauth et al. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Salvatore Attardo, editor. 2017. *The Routledge Handbook of Language and Humor*. Routledge, New York.

Joanne Boisson, Asahi Ushio, Hsuvas Borkakoty, Kiamehr Rezaee, Dimosthenis Antypas, Zara Siddique, Nina White, and Jose Camacho-Collados. 2024. How are metaphors processed by language models? the case of analogies. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pages 365–387, Miami, FL, USA. Association for Computational Linguistics.

James Brown. 1956. Eight types of puns. *PMLA*, 71(1):14–26.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Yuyan Chen, Zhixu Li, Jiaqing Liang, Yanghua Xiao, Bang Liu, and Yunwen Chen. 2023. Can pre-trained language models understand chinese humor? In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, WSDM '23, page 465–480, New York, NY, USA. Association for Computing Machinery.

Myra Cheng, Sunny Yu, Cinoo Lee, Pranav Khadpe, Lujain Ibrahim, and Dan Jurafsky. 2025. Social sycophancy: A broader understanding of llm sycophancy. *Preprint*, arXiv:2505.13995.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, and Qihao Zhu et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ryan Rony Dsilva. 2024. From sentence embeddings to large language models to detect and understand wordplay. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 205–214, Cham. Springer Nature Switzerland.

Liana Ermakova, Anne-Gwenn Bosser, Adam Jatowt, and Tristan Miller. 2023a. The joker corpus: English-french parallel data for multilingual wordplay recognition. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pages 2796–2806, New York, NY, USA. Association for Computing Machinery.

Liana Ermakova, Tristan Miller, Anne-Gwenn Bosser, Victor Manuel Palma Preciado, Grigori Sidorov, and Adam Jatowt. 2023b. Overview of joker – clef-2023 track on automatic wordplay analysis. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 397–415, Cham. Springer Nature Switzerland.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Patrícia Gameiro, Marcio Lima Inácio, Hugo Gonçalo Oliveira, and Ana Alves. 2025. Sequence labeling for pun location and detection in portuguese. In *Progress in Artificial Intelligence*, pages 254–266, Cham. Springer Nature Switzerland.

Sayan Ghosh and Shashank Srivastava. 2022. ePiC: Employing proverbs in context as a benchmark for abstract language understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3989–4004, Dublin, Ireland. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, and Akhil Mathur et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

John Healy and Leland McInnes. 2024. Uniform manifold approximation and projection. *Nature Reviews Methods Primers*, 4(1):82.

Elize Herrewijnen, Dong Nguyen, Floris Bex, and Kees van Deemter. 2024. Human-annotated rationales and explainable text classification: a survey. *Frontiers in Artificial Intelligence*, 7. Publisher: Frontiers.

Aditi Jain, Pratishtha Yadav, and Hira Javed. 2019. Equivoque: Detection and interpretation of english puns. In *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, pages 262–265.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Hazel H. Kim. 2025. How ambiguous are the rationales for natural language reasoning? a simple approach to handling rationale uncertainty. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10047–10053, Abu Dhabi, UAE. Association for Computational Linguistics.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Disen Liao and Freda Shi. 2025. How tokenization limits phonological knowledge representation in language models and how to improve them. In *Tokenization Workshop*.

Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah Smith, and Yejin Choi. 2023. We're afraid language models aren't modeling ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 790–807, Singapore. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Lars Malmqvist. 2025. Sycophancy in large language models: Causes and mitigations. In *Intelligent Computing*, pages 61–74, Cham. Springer Nature Switzerland.

J. A. Meaney, Steven Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 task 7: HaHackathon, detecting and rating humor and offense. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 105–119, Online. Association for Computational Linguistics.

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

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

Tristan Miller, Christian 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.

Tristan Miller and Mladen Turković. 2016. Towards the automatic detection and identification of english puns. *The European Journal of Humour Research*, 4(1):59–75.

Piotr Mirowski, Juliette Love, Kory Mathewson, and Shakir Mohamed. 2024. A robot walks into a bar: Can language models serve as creativity supporttools for comedy? an evaluation of llms' humour alignment with comedians. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, page 1622–1636, New York, NY, USA. Association for Computing Machinery.

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, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, and AJ Ostrow et al. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Md. Tauseef Qamar, Shahab Saquib Sohail, Gunjan Ansari, and Chandni Saxena. 2025. The language of nuance: Exploring the limits of large language models in handling ambiguity. In *Big Data Analytics in Astronomy, Science, and Engineering*, pages 180–192, Cham. Springer Nature Switzerland.

Kexin Quan, Pavithra Ramakrishnan, and Jessie Chin. 2025. Can ai take a joke—or make one? a study of humor generation and recognition in llms. In *Proceedings of the 2025 Conference on Creativity and Cognition*, C&C '25, pages 431–437, New York, NY, USA. Association for Computing Machinery.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Victor Raskin, editor. 2008. *The Primer of Humor Research*. De Gruyter Mouton, Berlin, New York.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R. Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R. Johnston, Shauna Kravec, Timothy Maxwell, Sam McCandlish, Kamal Ndousse, Oliver Rausch, Nicholas Schiefer, Da Yan, Miranda Zhang, and Ethan Perez. 2025. Towards understanding sycophancy in language models. *Preprint*, arXiv:2310.13548.

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.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

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 *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Sarah Wiegreffe, Ana Marasović, and Noah A. Smith. 2021. Measuring association between labels and free-text rationales. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10266–10284, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, and Daniel Hesslow et al. 2023. Bloom: A 176b-parameter open-access multilingual language model. *Preprint*, arXiv:2211.05100.

Zhijun Xu, Siyu Yuan, Lingjie Chen, and Deqing Yang. 2024. "a good pun is its own reword": Can large language models understand puns? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11766–11782, Miami, Florida, USA. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, and Chengyuan Li et al. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. 2024. CLAMBER: A benchmark of identifying and clarifying ambiguous information needs in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10746–10766, Bangkok, Thailand. Association for Computational Linguistics.

A Experimental Details

Throughout all experiments, the temperature was set to 0 to ensure more deterministic outputs, while the beam search decoding strategy was used for models that support it (beam size = 2). RoBERTa-large[4] was trained on the PunEval training split for max 4 epochs with early stopping on the validation set, based on F1-score. We use a batch size of 32 and a learning rate of $1.5e^{-5}$.

For testing, we utilized hosted instances of Llama 3.3-70B, Mistral, DeepSeek-R1-DistiLlama, and Gemini 2.0-Flash through OpenRouter[5]. The RoBERTa model was fine-tuned on a local server using the latest dumps available on Hugging Face.

For the embedding space analysis, we averaged token embeddings to obtain a sentence-level representation for each sample, preferring this strategy over `CLS` token embeddings, as the `CLS` token is not used during RoBERTa pre-training. Clusters representing patterns were identified using DBSCAN (Ester et al., 1996) with UMAP (Healy and McInnes, 2024) for dimensionality reduction. We employed k-nearest neighbor (kNN) to determine the optimal *eps* hyperparameter.

A.1 Prompts

The final prompts utilized are reported below. Table 5 presents a comparison of performance with other prompting strategies based on the validation split of the PunEval dataset using GPT-4o, the best-performing LLM. The +ZC prompts refer to those used with the Zero-shot CoT strategy. We edited the original prompt by removing all examples and instructing the models to "think step by step." We first asked them to produce reasoning and then to output the final answer. Since the performance was very similar to the reasoning (+R) prompt, we preferred the latter, as it allowed for a single prompt call instead of two. The RATIONALE and BASE prompts refer to the initial prompts that utilized JSON formatting for output. This strategy did not work well for smaller models, leading to the adoption of our custom prompting format, showcased in Figs. 7,6,5,4.

A.2 Keywords Evaluation

The output of the automatic evaluation indicates how many correct words the model produces in the rationale. During the evaluation process, both pun pairs are tokenized, lemmatized, and stripped of punctuation, with the overlap between each pair serving as the scoring metric. A score of 2 is assigned only if both w'_p and w'_a correctly match the true pun pair. A score of 1 is given if only one is correct, while a score of 0 is assigned if both are incorrect. This evaluation is conducted both with and without lemmatization, retaining the maximum score to mitigate issues arising from incorrect lemmatization. Pun words and alternative words are lemmatized using SpaCy, specifically the `en_core_web_lg` model[6].

Prompt	F1-score
Final prompts	
0S	90.2
FS	**95.6**
W	93.9
W+S	89.7
JSON prompts	
JSON-W+S	**87.1**
JSON-0S	85.0
Reasoning and Zero-shot CoT	
FS+R	**95.1**
W+R	93.3
W+S+R	93.3
FS+ZC	91.4
W+ZC	88.3
W+S+ZC	87.7

Table 5: F1-score of GPT-4o over different versions of prompts tried.

A.3 Error Analysis

For our error analysis, we sample 80 examples from the set of false positives identified in the results on the PunBreak dataset (which has the worst overall performance), sampling from results using the W+S prompt, which provides the most expressive rationale. Given that we have 200 examples for each of the four substitution categories, this represents 10% of the dataset size.

We hired five native English speakers from various backgrounds based on their expertise in corpus annotation. Specifically, the group consists of three men and two women who are students at our university, ranging from undergraduate to postgraduate levels.

The annotators were asked to answer 7-9 multiple-choice questions assessing the reasonableness of the models' responses for each selected

[4] https://huggingface.co/FacebookAI/roberta-large

[5] https://openrouter.ai

[6] https://spacy.io/models/en

sample. We conducted an initial round of annotation on a small subset to validate agreement among the annotators and gather feedback. The pilot study resulted in moderate inter-annotator agreement, with Fleiss's $\kappa = 0.41$).

We use Google Forms for collecting answers, and we show a screenshot of all questions in Fig. 16, 17, and 18. Each annotator reviewed a batch of 40 examples and was asked to annotate according to a set of guidelines that clarified the nature of the task and provided examples. The full guidelines, along with the questions, are provided at the end of this Appendix. Note that after the pilot study, we added two additional questions to assess the presence of "accidental" puns. However, we ultimately excluded these from the final study because the results were inconsistent among reviewers, which led us to conclude that the questions were not properly formulated and produced overly subjective responses that did not provide reliable evidence.

A.3.1 Results processing

We identify each question with a letter from A to I (see Fig. 16). Let $a(\ell)$ be the answer to the question identified by $\ell \in \{A, B, ..., I\}$. All questions have four possible answers: "Yes", "No", "Maybe" and "I don't understand". The exceptions are questions C and H, where respondents must express their rating on a Likert scale from 0 to 5. Although we received only one answer to question I, several annotators rated a small number of examples as possible puns, despite the low agreement on this question observed during the pilot test.

In our analysis, we adopt a conservative approach to detecting errors based on the annotators' responses, ensuring that errors are recorded only when the annotators express clear certainty. Hence, for answers on a qualitative scale, we assume "No" $= 0$ and all other answers are coded with 1. In this case, each error category is assigned to samples when specific conditions are met:

- **Word-sense association** (`word-sense`): $a(A)$ AND $a(B)$ (i.e., at least one of the senses must be clearly wrong);
- **Missing context** (`context`): $a(D)$ AND $a(E)$;
- **Pun pair** (`pun pair`): $a(F)$ OR $a(G)$ (i.e., the pun words must be both phonetically and orthographically diverse);
- **Senses similarity** (`sense sim`): $a(C) \leq 1$.

Note that additional error categories may be applicable. In some cases, annotators, despite finding no errors according to our guidelines, marked the example as a non-pun. This observation highlights that our guidelines may not be exhaustive, in part due to the inherent subjectivity of pun interpretation.

B Dataset information

Aggregated statistics for all datasets are presented in Table 6. In the PunEval dataset, we initially identified annotation issues by locating puns where the annotated w_p did not appear in the text. We detected 13 such issues, along with several typos, and addressed them by re-annotating the examples. The PunEval dataset also includes a list of contextual words that support both senses and a human explanation. Additionally, we removed contextual words that did not appear in the text (consistency errors). While the contextual words and human explanations are not utilized in this study, we have incorporated all these refinements in the published data splits.

B.1 PunnyPattern

To select common pun patterns for the PunnyPattern, we use scikit-learn[7] to train a logistic regressor (LR) on the PunEval training split and then infer the target labels on the testing split, using default parameters. The LR coefficients with higher absolute values are deemed more significant for the outcome. We train four separate LRs with n-gram features ($n \in \{1-4\}$), imposing a minimum frequency threshold for each feature (10 for unigrams and 3 for the others). In addition, we compute the percentage difference between the occurrences of each feature in puns and non-puns, as we seek features that effectively differentiate the two sets. The final feature score is calculated by multiplying this percentage difference by the absolute value of the corresponding coefficient, and the top 20 features are then manually reviewed.

We observed that n-grams belonging to the same pattern often appeared as separate features (e.g., both "*she was only*" and "*daughter*" are part of the pattern "*she was only a [...] daughter but*"). Consequently, we retained the 6 most representative patterns for each feature. Some features were discarded because they did not represent a genuine pattern (for example, the expression "*lot of*" ap-

[7] https://scikit-learn.org/stable/index.html

W+S *(system)*

/ Definition */*
Puns are a type of wordplay that use words with multiple meanings or similar-sounding words to create humor by juxtaposing these different meanings. Non-puns are jokes or statements that do not rely on this kind of wordplay. A pun is created by a pair of words or short expressions, referred to as "w_p" (the pun word) and "w_a" (the alternative word), which together create a humorous effect. Note that "w_p" and "w_a" must be the minimal text spans that create the pun. Depending on the type of pun, either "w_p" equals "w_a", or only "w_p" appears in the text, with "w_a" being evoked by the context. Each of these expressions, "w_p" and "w_a", carries its own meanings, denoted as "s_p" and "s_a" respectively, and these meanings are supported by a set of contextual words.
/ Instruction */*
You are a helpful assistant tasked with analyzing texts to determine if they contain a pun or not. You must first answer with 'yes' if the given text is a pun and 'no' if it is a non-pun. If you think it is a pun, you must also justify your answer by providing "w_p" and "w_a", along with their meanings "s_p" and "s_a".

W+S *(user)*

You must answer with 'yes' if the given text is a pun and 'no' if it is a non-pun.
If you think it is a pun, you must also justify your answer by providing the words, or short expressions, "w_p" and "w_a", along with their meanings "s_p" and "s_a". Note that "w_p" and "w_a" must be the minimal text spans that create the pun, or empty strings if the text is a non-pun. "s_p" and "s_a" must contain short definitions of "w_p" and "w_a" that match their meanings in the context of the sentence, or empty strings if the text is a non-pun. Please provide your answer in one line using the following formats: 'yes <w_p> <w_a> <s_p> <s_a>' for puns and 'no <> <> <> <>' for non-puns. Do not add any additional text or characters.
/ Examples */*
Text: A carpenter sat on his drill and was bored to tears.
Output: yes <bored> <bored> <make a hole, especially with a pointed power or hand tool> <cause to be bored>
Text: Don't kill the goose that lays the golden eggs?
Output: no <> <> <> <>
Text: I scream, you scream, we all scream for ice cream!
Output: yes <ice cream> <I scream> <a dessert made from frozen sweetened cream, usually flavored> <to utter a long loud piercing cry, as from pain or joy>
Text: He's dead Jim. Grab his tricorder. I'll get his wallet!
Output: no <> <> <> <>
Text: The whistling fisherman was always out of tuna.
Output: yes <tuna> <tune> <any very large marine food and game fish of the genus Thunnus> <the property of producing accurately a note of a given pitch>
Text: Better go about than fall into the ditch.
Output: no <> <> <> <>
Text: **INPUT TEXT** Output:

Figure 4: W+S system and user prompts.

Dataset	Avg. Statistics		Labels Count			
	N. chars	N. words	Heterographic puns	Homographic puns	Non-puns	Total samples
PunEval (train)	58	13	279	304	488	1071
PunEval (test)	58	13	312	461	568	1341
PunEval (val)	59	13	56	46	75	177
NAP	67	16	64	64	128	256
JOKER	64	14	381	251	632	1264
PunnyPattern	64	15	212	388	600	1200
PunBreak	66	15	100	100	900	1100

Table 6: Dataset statistics. Datasets in bold represent the new ones introduced in this work. *N. chars* and *N. words* denote the average number of characters and words per sample, respectively.

W *(system)*

/ Definition */*
Puns are a type of wordplay that use words with multiple meanings or similar-sounding words to create humor by juxtaposing these different meanings. Non-puns are jokes or statements that do not rely on this kind of wordplay. A pun is created by a pair of words or short expressions, referred to as "w_p" (the pun word) and "w_a" (the alternative word), which together create a humorous effect. Note that "w_p" and "w_a" must be the minimal text spans that create the pun. Depending on the type of pun, either "w_p" equals "w_a", or only "w_p" appears in the text, with "w_a" being evoked by the context. Each of these expressions, "w_p" and "w_a", carries its own meanings, denoted as "s_p" and "s_a" respectively, and these meanings are supported by a set of contextual words.
/ Instruction */*
You are a helpful assistant tasked with analyzing texts to determine if they contain a pun or not. You must first answer with 'yes' if the given text is a pun and 'no' if it is a non-pun. If you think it is a pun, you must also justify your answer by providing "w_p" and "w_a".

W+S *(user)*

You must answer with 'yes' if the given text is a pun and 'no' if it is a non-pun. If you think it is a pun, you must also justify your answer by providing the words, or short expressions, "w_p" and "w_a". Note that "w_p" and "w_a" must be the minimal text spans that create the pun, or empty strings if the text is a non-pun. Please provide your answer in one line using the following formats: 'yes <w_p> <w_a>' for puns and 'no <> <>' for non-puns. Do not add any additional text or characters.
/ Examples */*
Text: A carpenter sat on his drill and was bored to tears.
Output: yes <bored> <bored>
Text: Don't kill the goose that lays the golden eggs?
Output: no <> <>
Text: He's dead Jim. Grab his tricorder. I'll get his wallet!
Output: no <> <>
Text: The whistling fisherman was always out of tuna.
Output: yes <tuna> <tune>
Text: Better go about than fall into the ditch.
Output: no <> <>
Text: **INPUT TEXT** Output:

Figure 5: W system and user prompts.

FS *(system)*

/ Definition */*
Puns are a type of wordplay that use words with multiple meanings or similar-sounding words to create humor by juxtaposing these different meanings. Non-puns are jokes or statements that do not rely on this kind of wordplay. A pun is created by a pair of words or short expressions, referred to as "w_p" (the pun word) and "w_a" (the alternative word), which together create a humorous effect. Note that "w_p" and "w_a" must be the minimal text spans that create the pun. Depending on the type of pun, either "w_p" equals "w_a", or only "w_p" appears in the text, with "w_a" being evoked by the context. Each of these expressions, "w_p" and "w_a", carries its own meanings, denoted as "s_p" and "s_a" respectively, and these meanings are supported by a set of contextual words.
/ Instruction */*
You are a helpful assistant tasked with analyzing texts to determine if they contain a pun or not. You must answer only with 'yes' if the given text is a pun and 'no' if it is a non-pun.

FS *(user)*

You must answer only with 'yes' if the given text is a pun and 'no' if it is a non-pun. Do not add any additional text or characters.
/ Examples */*
Text: A carpenter sat on his drill and was bored to tears.
Output: yes
Text: Don't kill the goose that lays the golden eggs?
Output: no
Text: I scream, you scream, we all scream for ice cream!
Output: yes
Text: He's dead Jim. Grab his tricorder. I'll get his wallet!
Output: no
Text: The whistling fisherman was always out of tuna.
Output: yes
Text: Better go about than fall into the ditch.
Output: no
Text: **INPUT TEXT** Output:

Figure 6: FS system and user prompts.

0S *(system)*

You are a helpful assistant tasked with analyzing texts to determine if they contain a pun or not. You must answer only with 'yes' if the given text is a pun and 'no' if it is a non-pun.

0S *(user)*

You must answer only with 'yes' if the given text is a pun and 'no' if it is a non-pun. Do not add any additional text or characters.
Text: **INPUT TEXT** Output:

Figure 7: 0S system and user prompts.

peared only 8 times and likely reflects a spurious correlation rather than a common pun construction).

After selecting the patterns, we sampled all puns containing these language patterns from the PunEval dataset. Missing examples were collected online and generated via GPT-4o using the prompt shown in Fig. 8, with the temperature set to 1.0. All generated examples were manually reviewed, and those that were poorly crafted were either revised or discarded. We queried the model multiple times until we obtained 100 puns for each pattern.

B.2 PunBreak

To create the PunBreak, we selected 100 heterographic puns and 100 homographic puns from the NAP and PunEval test split. We generated alternatives using GPT-4o-mini for `homophone` replacements and employed Google Translate to ensure that the substituted words sounded similar to the originals. For `random` replacements, we compiled a list of random words and replaced them in the dataset, ensuring that these words did not fit contextually within the sentences. To generate synonyms (`alt syn` and `pun syn`), we utilized online thesauruses and dictionaries to find synonyms for each word; when synonyms were unavailable, we opted for a hypernym instead. We do our best to ensure that none of these substitutions result in accidental puns. For example, consider the pun "*«Consult an investment broker»was Tom's stock (stock) answer*", which plays on *stock* as both "*commonly used or brought forward*" (s_p) and "*capital raised by a corporation representing ownership interest*" (s_a). For the synonym substitutions, we replaced "*stock*" with "*routine*", which is roughly equivalent in context, and *equity*, respectively. We purposely avoided using "*default*" in the second substitution, as it carries an economic connotation. As a nonsensical but phonetically similar expression, we used "*stork*" (a bird) and "*yawn*" as a random replacement.

C Additional Results

Table 7 reports recall over all prompts and the dataset, divided by heterographic and homographic puns.

C.1 Effect of rationale

Fig. 9 highlights the impact of rationale-augmented prompts for all models on the NAP and JOKER datasets. Fig. 12 compares the impact of the best rationale-augmented prompt (W or W+S) on pun detection performance over the PunnyPattern dataset, excluding the `tom` and `when` patterns that do not consistently bias the tested LLMs. Fig. 13 shows the same results on the PunBreak, when considering only the four substitution sets (800 negative examples). The best prompt used is the one highlighted in Table 10, except for Mistral3, where we select the W prompt as it appears to work better overall. In the violin plots, the right side (light blue) represents the results distributions over the three runs with the rationale-augmented prompt, while the left side (red) represents performance using the FS prompt — typically the best prompt without rationales. Note that in all cases except Mistral, incorporating rationalization reduces the number of false positives, thereby lowering recall and improving precision, countering the bias effect.

C.2 RQ1

Fig. 9 shows the difference in performance for each prompt used. Fig. 10 shows the same effect over the PunnyPattern dataset. This shows that, overall, the W or W+S prompt improves performance over the simpler prompts.

C.3 RQ2: Pun Language Patterns

Pattern identification. Table 8 lists the six language patterns identified as commonly used to create puns. Table 9 summarizes the statistics for these patterns in our datasets. We omit data for the PunnyPattern dataset, as all examples in it are designed to contain a language pattern.

Performance. Table 10 compares the results obtained using each prompt for PunEval and PunnyPattern. Note that the best prompts (in bold) are the same as those that performed best in our RQ1 (Table 1). Fig. 11 visualizes the performance differences for each set of expressions containing language patterns. We interpret bias as being present in models with very high recall but low precision, indicating they tend to classify all instances containing the pattern as puns.

We observe a sharp imbalance of low precision and high recall for the `never_die`, `used`, and `doctor` patterns (Fig. 11), indicating that these patterns tend to mislead LLMs into identifying puns. In contrast, the `tom` and `when` patterns do not appear to consistently affect them. While all LLMs tend to produce many false positives with the `used`,

Non-puns generation

/ Definition */*
Puns are a form of wordplay that exploits multiple meanings of a word or similar-sounding words to produce a humorous effect. Puns typically involve a clever juxtaposition of words or phrases, where the humor arises from the interplay of their meanings or sounds. In contrast, non-puns are jokes or statements that do not rely on such linguistic ambiguities.
/ Instructions */*
You are a creative linguist who thinks out-of-the-box. You receive a JSON-formatted string in the format "expression": "EXPR1 [X] EXPR2", "count": C where [X] represents a missing expression, and you must generate C pairs of distinct short paragraphs where: - the first text in the pair must be a short paragraph containing a pun and respect the template given in "expression"; - the second text in the pair must be a rephrased version of the first one, that removes the pun (i.e., a non-pun) but keeps a similar meaning. You can add words before or after the template, but you must ensure that each generated paragraph contains the given template with proper substitutions. You must format each pair in JSON format: "pun": "[generated pun]", "non-pun": "[generated non-pun]". You must make sure that field "pun" contains a good pun, and "non-pun" contains no pun at all. You must write each JSON-formatted pair in a new line, respecting the JSONL format. Do not add any superfluous text and report a JSON-formatted string in each line.
/ Examples */*
INPUT: "expression": "Old [X] never die, they just", "count": 3
OUTPUT: "pun": "Old wizards never die, they just improve their spelling.", "non-pun": "Old wizards never die, they just improve their pronunciation." "pun": "Old squirrels never die, they just go nuts!", "non-pun": "Old squirrels never die, they just go crazy!" "pun": "Old crabs never die, they just became a little more shellfish!", "non-pun": "Old crabs never die, they just became a little more egocentric!"
INPUT: "expression": "is", "count": 2
OUTPUT: "pun": "Denial is a river in Egypt.", "non-pun": "The river Nile is a river in Egypt." "pun": "The cyclist is two tired to win", "non-pun": "The cyclist is too drained to win"
INPUT: "expression": "This [X] always", "count": 2
OUTPUT: "pun": "This vacuum always sucks!", "non-pun": "This vacuum always disappoints!" "pun": "This candy cane always remains in mint condition.", "non-pun": "This candy cane always remains in good condition."
INPUT: **INPUT EXPRESSION**
OUTPUT:

Figure 8: Prompt used to generate puns and non-puns containing language patterns.

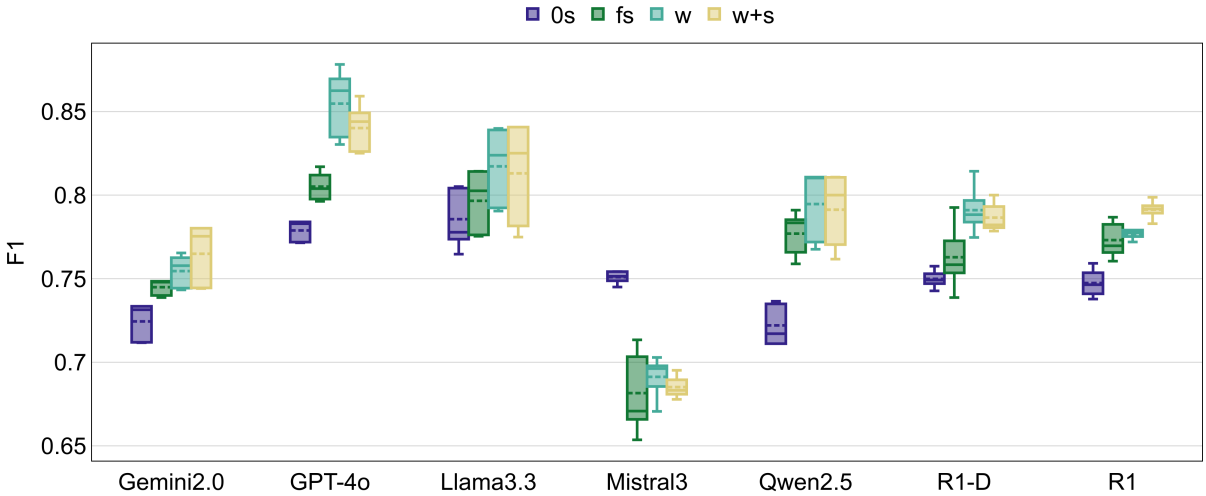

Figure 9: Distribution of F1-score over 3 runs on the NAP and JOKER datasets. The dotted line represents the mean, while the solid line is the median.

Model	Prom.	NAP		JOKER		PunEval		PunnyPattern	
		hom	het	hom	het	hom	het	hom	het
Gemini2.0	0S	100.0 ±0.0	100.0 ±0.0	96.0 ±0.0	95.9 ±0.2	97.3 ±0.1	92.3 ±0.0	97.2 ±3.6	95.4 ±6.0
	FS	96.9 ±0.0	98.4 ±0.0	92.4 ±0.0	93.6 ±0.2	95.3 ±0.1	88.3 ±0.2	97.1 ±3.1	96.2 ±5.7
	W	94.3 ±0.9	97.9 ±0.9	90.4 ±0.0	94.0 ±0.0	94.3 ±0.1	86.9 ±0.0	92.0 ±8.7	95.3 ±7.6
	W+S	98.4 ±0.0	98.4 ±0.0	87.7 ±0.0	91.9 ±0.0	94.0 ±0.1	85.7 ±0.4	91.5 ±8.0	91.4 ±10.3
GPT-4o	0S	99.0 ±0.9	100.0 ±0.0	95.2 ±0.0	96.3 ±0.0	96.4 ±0.3	93.0 ±0.6	98.2 ±2.3	98.3 ±2.3
	FS	97.4 ±0.9	98.4 ±0.0	93.6 ±0.0	94.6 ±0.2	94.2 ±0.3	88.9 ±0.4	96.9 ±3.5	96.9 ±2.9
	W	84.9 ±2.4	98.4 ±0.0	79.9 ±0.3	92.7 ±0.4	77.7 ±3.0	80.3 ±0.7	86.2 ±8.3	91.3 ±9.8
	W+S	87.5 ±1.6	97.9 ±0.9	85.9 ±0.3	93.0 ±0.2	81.1 ±1.9	82.8 ±1.4	88.2 ±7.2	94.7 ±6.1
Llama3.3	0S	96.9 ±0.0	99.5 ±0.9	92.8 ±0.0	94.2 ±0.4	94.6 ±1.3	86.7 ±2.5	97.3 ±3.8	94.4 ±6.7
	FS	98.4 ±0.0	97.4 ±2.4	92.6 ±0.3	93.3 ±0.6	94.9 ±0.3	88.1 ±1.1	97.0 ±4.2	96.2 ±4.7
	W	90.1 ±0.9	93.2 ±1.8	87.9 ±0.3	90.0 ±0.0	89.4 ±0.6	82.2 ±0.7	95.1 ±4.3	92.7 ±5.1
	W+S	96.9 ±0.0	97.4 ±0.9	91.8 ±0.3	92.0 ±0.6	92.6 ±0.7	83.0 ±0.6	93.7 ±7.5	93.9 ±7.7
Mistral3	0S	93.2 ±0.9	80.7 ±0.9	79.5 ±0.3	86.0 ±0.6	80.3 ±1.2	68.7 ±0.8	87.8 ±12.4	87.9 ±16.3
	FS	88.0 ±3.3	87.0 ±9.2	84.7 ±1.4	86.6 ±4.5	81.1 ±1.1	81.8 ±4.0	88.3 ±3.9	86.2 ±5.3
	W	88.0 ±3.9	96.9 ±3.1	91.4 ±0.8	94.6 ±0.6	84.0 ±1.2	82.5 ±0.5	92.2 ±6.0	90.8 ±8.2
	W+S	95.3 ±1.6	96.4 ±2.4	95.6 ±0.6	97.6 ±0.4	93.2 ±1.2	92.1 ±0.8	97.6 ±1.6	97.5 ±4.1
Qwen2.5	0S	100.0 ±0.0	100.0 ±0.0	92.8 ±0.6	96.5 ±0.2	96.1 ±0.4	88.6 ±0.4	97.1 ±4.2	95.2 ±11.2
	FS	95.8 ±0.9	96.4 ±0.9	84.5 ±1.7	91.3 ±0.0	87.6 ±0.1	78.5 ±0.3	92.2 ±7.5	91.6 ±13.1
	W	94.8 ±0.9	93.8 ±1.6	80.7 ±0.8	86.5 ±0.2	85.0 ±0.6	75.1 ±0.7	88.1 ±11.8	87.6 ±17.0
	W+S	94.3 ±0.9	94.3 ±0.9	82.1 ±1.7	87.8 ±0.2	87.8 ±0.8	75.8 ±1.0	88.5 ±12.3	88.4 ±16.3
R1-D	0S	100.0 ±0.0	99.0 ±0.9	97.8 ±0.3	98.3 ±0.2	96.8 ±0.3	96.6 ±0.4	98.4 ±1.9	99.0 ±2.1
	FS	98.4 ±1.6	96.9 ±2.7	96.4 ±1.7	96.8 ±0.4	96.8 ±0.2	93.3 ±1.2	97.0 ±3.4	98.0 ±3.3
	W	93.7 ±2.7	99.0 ±0.9	91.8 ±0.3	96.6 ±1.1	91.9 ±1.1	91.8 ±1.0	92.6 ±3.5	97.0 ±3.6
	W+S	94.8 ±0.9	99.0 ±1.8	92.6 ±2.0	94.4 ±0.2	93.2 ±1.0	87.9 ±1.0	93.9 ±5.2	95.9 ±4.8
R1	0S	100.0 ±0.0	100.0 ±0.0	99.0 ±0.3	99.0 ±0.4	99.3 ±0.1	99.2 ±0.2	99.7 ±0.5	99.0 ±1.6
	FS	99.5 ±0.9	99.0 ±0.9	97.8 ±0.3	97.9 ±0.4	98.9 ±0.1	98.0 ±0.5	98.8 ±0.9	98.3 ±1.6
	W	98.4 ±0.0	100.0 ±0.0	98.4 ±0.0	98.6 ±0.2	98.8 ±0.3	98.5 ±0.2	99.2 ±1.1	99.1 ±1.1
	W+S	99.0 ±0.9	100.0 ±0.0	97.0 ±0.9	97.9 ±0.4	98.4 ±0.1	97.4 ±0.9	98.3 ±1.2	98.7 ±1.7

Table 7: Recall (puns) on the subsets of heterographic (het) and homographic (hom) puns for each dataset.

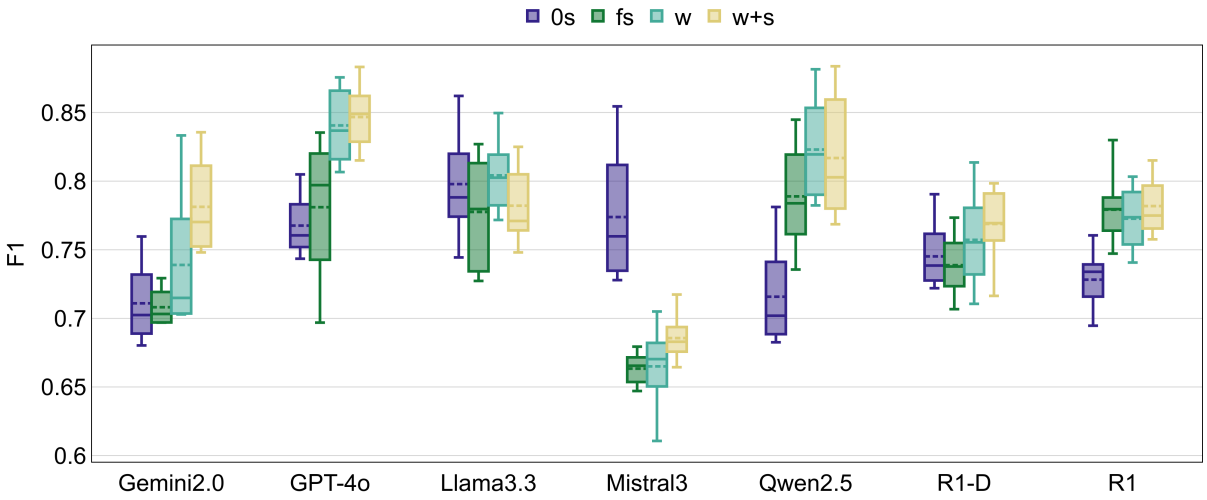

Figure 10: Distribution of F1-score over 3 runs on the PunnyPattern. The dotted line represents the mean, while the solid line is the median.

Structure	Short name	Example
Old [...] never die [...] they	`never_die`	OLD BANKERS never die they just lose interest
Tom	`tom`	"I've been to a film festival in Southern France" said Tom cannily
When the	`when`	When the TV repairman got married the reception was excellent
She was only [...] daughter but	`daughter`	She was only a horseman's daughter, but she didn't know how to say neigh
Doctor, Doctor	`doctor`	Doctor, Doctor, I keep thinking I'm a spoon. - Sit there and don't stir
used to [...] but	`used`	I used to be addicted to soap, but I'm clean now

Table 8: Pun language patterns and their occurrences in the PunEval dataset.

Pattern	Occurrences					
	PunEval (train)	PunEval (test)	PunEval (val)	NAP	JOKER	PunBreak
`never_die`	65	62	1	2	12	50
`tom`	61	61	1	0	0	15
`when`	13	20	0	0	22	5
`daughter`	11	19	1	0	0	15
`doctor`	3	7	0	0	0	0
`used`	2	4	1	2	10	10

Table 9: Occurrences of pun language patterns in all the datasets.

`never_die`, and `doctor` patterns, results for the other patterns are more varied. For instance, GPT-4o, Qwen2.5, and Mistral3 are more resilient to the `daughter` pattern compared to the other models. Interestingly, examples containing the word "Tom" produce both false positives and negatives, suggesting that this pattern is unlikely to serve as a reliable shortcut for classifying puns. In contrast, sentences beginning with "When the" are classified more reliably by all models.

C.4 RQ2: Pun Alterations

Table 11 contains the performance (recall) on the substitution categories for each prompt. For the "Pun" column, the recall is computed for class 1 (pun), while for the other columns, recall is relative to class 0 (non-pun). The "Rand sent" column represents the set of 100 generated non-puns used to control for unseen bias toward class 1 (pun). The highest values for each model are highlighted in bold.

Model confidence. We observe that, on most substitution categories in the PunBreak dataset, the standard deviation between measurements is significantly higher compared to the examples of positive (i.e., `pun`) and random sentences (`rand sent`). This indicates that models are less confident when classifying altered puns. To further investigate this, we retrieve the confidence values for the responses across the entire dataset. Specifically, we collect the log probabilities for the "yes" or "no" tokens.

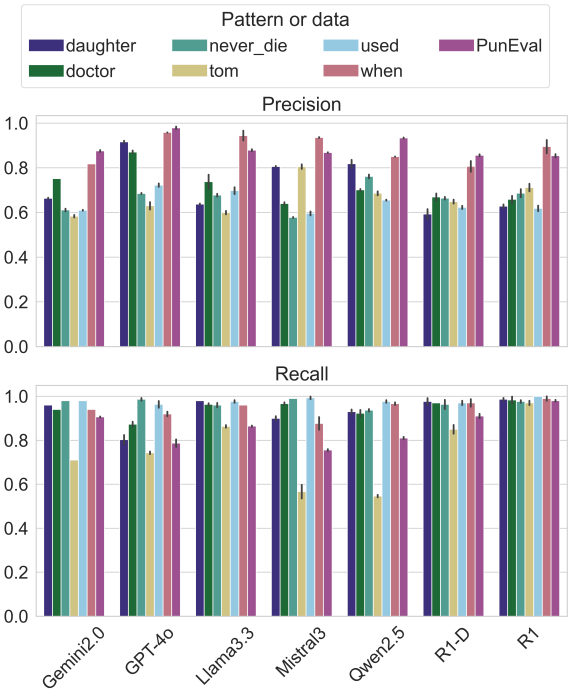

Figure 11: Precision and recall on the PunnyPattern dataset using the best prompt (all patterns).

Model	Prom.	F1		Precision		Recall		Accuracy	
		PP	PE	PP	PE	PP	PE	PP	PE
Gemini2	0S	71.9 ± 4.6	85.7 ± 0.2	57.6 ± 5.4	78.0 ± 0.2	96.2 ± 5.3	**95.3** ± 0.1	62.2 ± 7.6	81.7 ± 0.2
	FS	73.0 ± 6.4	**89.3** ± 0.1	59.1 ± 8.5	86.3 ± 0.1	**96.4** ± 4.8	92.5 ± 0.2	63.7 ± 10.1	**87.2** ± 0.1
	W	74.7 ± 6.6	88.3 ± 0.1	62.7 ± 8.8	85.5 ± 0.2	93.6 ± 7.5	91.3 ± 0.1	67.7 ± 10.0	86.0 ± 0.1
	W+S	**76.9** ± 7.5	89.1 ± 0.2	**66.7** ± 8.6	**87.6** ± 0.4	91.6 ± 9.5	90.7 ± 0.2	**72.3** ± 9.0	**87.2** ± 0.2
GPT-4o	0S	78.3 ± 8.1	89.7 ± 0.1	65.8 ± 12.2	84.9 ± 0.2	**98.3** ± 2.2	**95.0** ± 0.4	71.8 ± 11.6	87.4 ± 0.1
	FS	78.8 ± 9.0	**92.8** ± 0.2	67.0 ± 12.7	93.5 ± 0.3	97.1 ± 3.1	92.1 ± 0.2	72.6 ± 13.9	**91.8** ± 0.2
	W	**83.1** ± 7.8	87.3 ± 0.9	**79.7** ± 12.3	**97.9** ± 0.5	88.1 ± 8.6	78.7 ± 1.7	**81.7** ± 9.2	86.8 ± 0.7
	W+S	83.0 ± 8.7	88.9 ± 0.9	77.4 ± 13.5	97.3 ± 0.1	90.8 ± 5.9	81.8 ± 1.6	80.6 ± 11.6	88.2 ± 0.8
Llama3.3	0S	79.4 ± 7.3	84.0 ± 0.4	68.1 ± 9.0	77.8 ± 1.8	95.9 ± 6.1	91.4 ± 1.8	74.8 ± 9.4	79.9 ± 0.9
	FS	78.9 ± 7.5	84.4 ± 0.4	67.2 ± 10.5	77.8 ± 0.7	**96.8** ± 4.6	**92.2** ± 0.6	73.5 ± 10.4	80.3 ± 0.6
	W	**81.0** ± 7.5	**87.2** ± 0.2	**71.5** ± 11.1	**87.9** ± 0.5	94.2 ± 4.6	86.5 ± 0.3	**77.3** ± 9.5	**85.3** ± 0.3
	W+S	78.7 ± 9.1	86.4 ± 0.3	68.4 ± 12.6	84.2 ± 0.9	94.0 ± 7.4	88.7 ± 0.3	74.0 ± 11.5	83.9 ± 0.5
Mistral3	**0S**	**77.4** ± 8.0	**80.8** ± 0.2	**72.1** ± 13.2	**86.9** ± 0.1	87.5 ± 14.9	75.6 ± 0.4	**74.2** ± 9.8	**79.3** ± 0.2
	FS	66.3 ± 2.2	74.6 ± 1.9	53.7 ± 2.1	68.8 ± 1.9	87.2 ± 5.1	81.4 ± 2.3	55.8 ± 3.1	68.0 ± 2.4
	W	68.2 ± 3.5	78.4 ± 0.5	54.6 ± 3.8	74.0 ± 0.6	91.4 ± 7.3	83.4 ± 0.9	57.3 ± 5.6	73.6 ± 0.6
	W+S	68.7 ± 1.6	74.9 ± 0.9	53.1 ± 1.7	62.8 ± 1.3	**97.2** ± 2.9	**92.8** ± 1.0	55.6 ± 3.0	64.2 ± 1.7
Qwen2.5	0S	74.0 ± 8.3	83.6 ± 0.1	61.1 ± 10.9	75.9 ± 0.0	**95.6** ± 9.0	**93.1** ± 0.3	65.5 ± 12.4	79.0 ± 0.1
	FS	77.9 ± 7.1	87.2 ± 0.2	68.4 ± 6.7	90.7 ± 0.3	91.4 ± 11.1	83.9 ± 0.2	74.2 ± 7.6	85.8 ± 0.2
	W	**79.7** ± 9.6	86.8 ± 0.4	**74.1** ± 7.2	**93.4** ± 0.2	87.5 ± 15.1	81.0 ± 0.5	**78.2** ± 8.5	85.8 ± 0.3
	W+S	79.1 ± 9.2	**87.3** ± 0.1	72.6 ± 7.6	92.1 ± 0.6	88.3 ± 14.8	82.9 ± 0.3	77.1 ± 8.5	**86.0** ± 0.2
R1-D	0S	75.4 ± 4.1	86.7 ± 0.3	61.2 ± 5.2	78.5 ± 0.4	**98.5** ± 2.0	**96.7** ± 0.1	67.5 ± 6.5	82.9 ± 0.4
	FS	75.1 ± 4.4	87.0 ± 0.3	61.3 ± 5.6	80.0 ± 0.5	97.4 ± 3.1	95.3 ± 0.6	67.4 ± 7.0	83.5 ± 0.4
	W	77.2 ± 4.6	86.8 ± 0.4	65.6 ± 6.2	82.2 ± 0.7	94.1 ± 3.1	91.9 ± 1.0	71.9 ± 6.9	83.8 ± 0.5
	W+S	**78.3** ± 4.9	**88.3** ± 0.6	**66.9** ± 6.7	**85.6** ± 0.4	94.8 ± 4.5	91.1 ± 1.0	**73.4** ± 6.9	**86.1** ± 0.7
R1	0S	76.4 ± 6.6	88.5 ± 0.2	62.5 ± 9.4	79.8 ± 0.3	**99.3** ± 1.2	**99.2** ± 0.1	68.5 ± 10.2	85.1 ± 0.3
	FS	80.9 ± 6.5	90.8 ± 0.2	69.1 ± 10.3	84.3 ± 0.4	98.6 ± 1.1	98.5 ± 0.2	76.1 ± 8.9	88.5 ± 0.3
	W	79.9 ± 5.8	90.9 ± 0.1	67.3 ± 8.8	84.2 ± 0.2	99.0 ± 0.9	98.7 ± 0.1	74.6 ± 8.3	88.6 ± 0.1
	W+S	**81.1** ± 6.0	**91.3** ± 0.4	**69.5** ± 9.4	**85.4** ± 0.7	98.3 ± 1.0	98.0 ± 0.4	**76.6** ± 8.2	**89.2** ± 0.6

Table 10: Percentage F1-score, precision and recall (± standard deviation) on PunnyPattern (PP) and PunEval (PE) datasets. Best results and best overall prompt per model are in bold.

Model	Prompt	Substitution Category					Control groups	
		Homophone	Random	Pun syn	Alt syn	Pun	Rand sent	JOKER
Gemini2.0	0S	**23.7** ± 5.5	**37.0** ± 1.5	11.5 ± 0.5	18.0 ± 5.5	**99.0** ± 1.1	70.3 ± 0.5	26.3 ± 0.1
	FS	9.3 ± 0.8	16.2 ± 1.0	14.7 ± 0.5	20.3 ± 5.9	98.0 ± 1.1	85.7 ± 0.5	41.3 ± 0.2
	W	10.0 ± 1.1	14.5 ± 4.9	13.2 ± 1.0	20.3± 7.7	97.5 ± 1.4	87.0 ± 0.0	43.8 ± 0.3
	W+S	11.8 ± 1.0	21.5 ± 0.5	**18.2** ± 2.0	**25.3**± 7.3	97.0 ± 1.1	**89.3** ± 0.5	**47.9** ± 0.1
GPT-4o	0S	9.2 ± 1.3	23.8 ± 1.6	10.7 ± 0.5	23.2 ± 4.9	**99.2** ± 0.4	87.7 ± 1.4	47.4 ± 0.1
	FS	12.5 ± 3.1	26.2 ± 3.5	20.5 ± 0.5	32.5 ± 9.1	97.2 ± 1.2	91.3 ± 0.5	57.8 ± 0.6
	W	**32.5** ± 2.3	**57.5** ± 1.4	**37.3** ± 4.1	**58.7**± 4.8	92.8 ± 3.2	**99.0** ± 0.0	**77.4** ± 0.9
	W+S	24.0 ± 0.9	39.5 ± 5.5	30.0 ± 1.5	49.0 ± 5.9	93.5 ± 3.6	97.7 ± 0.5	71.8 ± 0.6
Llama3.3	0S	22.0 ± 1.4	45.5 ± 3.4	16.5 ± 2.7	35.2 ± 11.1	**96.8** ± 1.5	79.7 ± 2.1	50.6 ± 3.0
	FS	16.3 ± 6.0	37.0 ± 4.3	21.8 ± 1.5	40.3 ± 13.0	96.7 ± 1.6	88.7 ± 1.0	53.2 ± 0.1
	W	**23.3** ± 3.8	**49.2** ± 1.2	**35.2** ± 5.2	**51.5**± 7.2	90.5 ± 2.1	**94.0** ± 0.9	**63.9** ± 0.7
	W+S	13.2 ± 3.8	32.8 ± 5.9	26.5 ± 2.7	42.5 ± 7.6	92.3 ± 1.2	93.0 ± 0.0	56.0 ± 1.6
Mistral3	**0S**	**37.3** ± 6.7	**53.2** ± 3.3	**29.2** ± 1.2	**44.3**± 19.0	88.8 ± 2.0	**90.0** ± 0.0	**61.9** ± 0.2
	FS	14.5 ± 4.6	17.0 ± 5.8	15.8 ± 2.0	21.7 ± 5.0	86.7 ± 2.4	37.7 ± 1.4	26.4 ± 0.2
	W	5.2 ± 1.9	9.2 ± 1.5	12.3 ± 2.0	19.2 ± 5.1	93.3 ± 3.4	48.7 ± 3.1	24.1 ± 1.6
	W+S	2.7 ± 1.4	3.5 ± 2.5	6.5 ± 1.8	9.5 ± 3.4	**96.7** ± 1.5	25.7 ± 4.2	14.0 ± 1.7
Qwen2.5	0S	5.0 ± 0.6	11.2 ± 3.7	5.7 ± 0.8	11.3 ± 3.8	**98.2** ± 0.4	59.3 ± 2.1	36.6 ± 0.3
	FS	11.3 ± 3.4	14.7 ± 2.8	19.0 ± 4.6	33.7 ± 11.7	94.7 ± 1.0	90.0 ± 0.9	56.5 ± 0.9
	W	**19.2** ± 5.2	**32.7** ± 3.4	26.3 ± 5.6	**40.5**± 12.2	92.8 ± 1.6	**94.3** ± 1.4	**65.7** ± 0.5
	W+S	15.5 ± 4.4	29.8 ± 5.8	**27.0** ± 3.6	37.2 ± 11.9	95.0 ± 1.4	92.7 ± 1.4	62.7 ± 1.1
R1-D	0S	6.2 ± 1.2	24.9 ± 6.6	14.2 ± 1.2	21.2± 8.1	**99.3** ± 0.5	59.0 ± 2.4	37.5 ± 2.2
	FS	7.1 ± 2.3	27.3 ± 3.8	18.8 ± 4.5	26.9 ± 9.8	98.5 ± 0.5	69.7 ± 5.5	42.9 ± 1.3
	W	8.5 ± 3.7	**36.9** ± 3.1	26.7 ± 2.7	34.6± 8.3	96.5 ± 2.0	79.3 ± 5.2	52.1 ± 1.4
	W+S	**9.7** ± 2.2	34.3 ± 1.4	**28.8** ± 5.0	**38.8**± 11.6	96.3 ± 1.2	**82.7** ± 2.3	**55.5** ± 3.3
R1	0S	6.5 ± 2.7	27.3 ± 4.1	12.2 ± 2.0	18.8 ± 7.3	99.8 ± 0.4	71.3 ± 1.0	37.0 ± 1.7
	FS	8.5 ± 2.3	38.8 ± 3.7	22.3 ± 2.9	31.8 ± 4.1	99.3 ± 0.8	83.0 ± 0.9	48.2 ± 1.5
	W	9.5 ± 1.8	40.3 ± 2.7	22.3 ± 3.1	34.0 ± 6.0	99.8 ± 0.4	84.7 ± 2.3	44.9 ± 0.5
	W+S	**10.5** ± 2.6	**41.8**± 2.0	**25.5** ± 2.8	**35.3**± 8.3	**99.8** ± 0.4	**84.7** ± 1.9	**51.2** ± 0.1

Table 11: Percentage recall (on the true class) ± standard deviation for each substitution category on the PunBreak dataset. Note that for all columns except JOKER, *recall* equals *accuracy* because each evaluation set contains only puns or only non-puns. JOKER reports recall for the negative class on the full dataset. The best overall prompt per model is marked in bold.

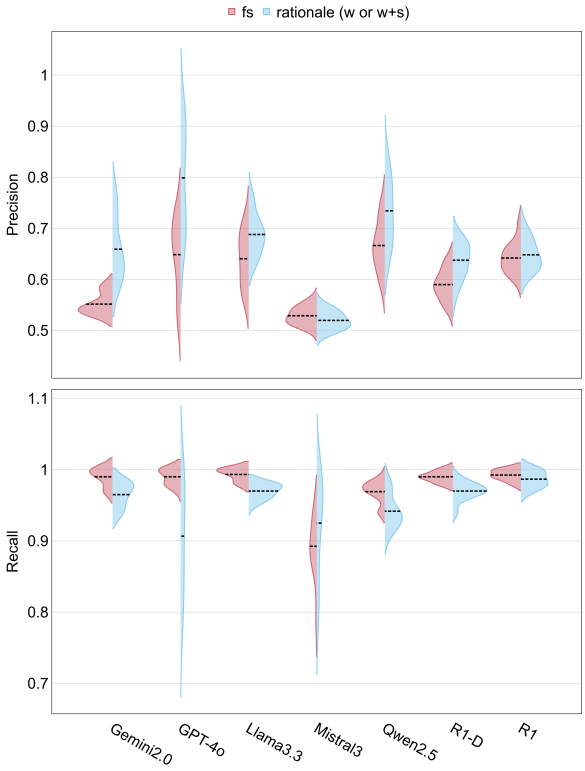

Figure 12: Precision and recall over the PunnyPattern dataset using the few-shot (`fs`) and the best prompt between `w` and `w+s`.

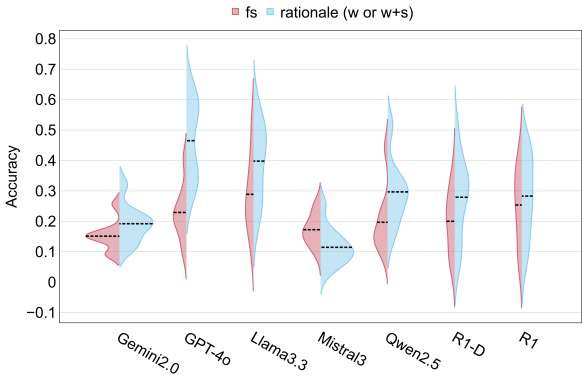

Figure 13: Accuracy over the PunBreak dataset (only negative examples) using the few-shot (`fs`) and the best prompt between `w` and `w+s`.

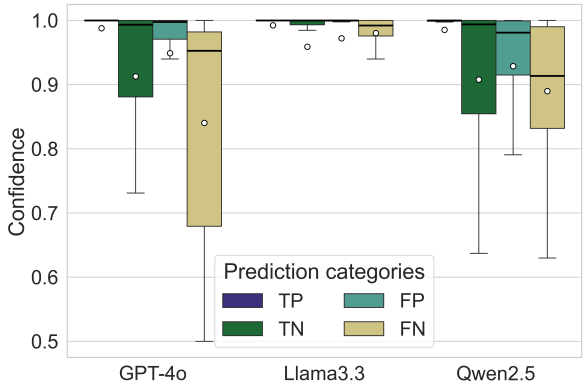

Figure 14: Confidence on each prediction class (`w+s` prompt).

Since not all LLMs provide the option to retrieve confidence scores at the time of writing (e.g., DeepSeek models and Gemini), we limit this step to GPT-4o, Qwen 2.5, and Llama3.3, which represent three top-performing models in the previous tasks.

Our analysis confirms that, on the PunBreak dataset, these models exhibit very high confidence when predicting puns but are much less confident when predicting the non-pun class. Fig. 14 clearly illustrates this trend, with the True Negative (TN) and False Negative (FN) classes showing significantly more variance than the other classes. It is important to note that the number of FNs is quite low, with only 38 instances across the three models. The lower confidence levels indicate that classifying altered examples as non-puns is significantly more challenging for the models compared to correctly identifying puns from the PunEval dataset and the negative sentences we generated as a control set.

C.5 RQ3

To isolate the quality of the rationales from raw detection performance, we introduce an alternative evaluation criterion. For this analysis, we calculate the PPA score exclusively on true positives (i.e., puns that the LLMs correctly identified). The results for this specific subset are presented in Table 12.

This targeted analysis reveals that most models can generate reasonably accurate rationales for the puns they successfully identify. Mistral and Qwen achieve lower scores in this setting, producing up to 40% incorrect rationales, while GPT-4o, Gemini, and R1 produce correct predictions for the wrong reasons at least 20% of the time.

However, this metric does not account for a model's overall detection rate; a model might excel at explaining the few puns it identifies, yet fail to detect many others. We therefore argue that the comprehensive PPA score in Table 3, which considers all predictions, provides a more robust and useful comparison. Consequently, we base our selection of models for manual analysis on that initial metric.

The barplot in Fig. 15 shows the distribution of

Model	NAP		JOKER		PunEval	
	W	W+S	W	W+S	W	W+S
Gemini2.0	**1.6**	**1.5**	1.4	1.4	1.6	1.6
GPT-4o	1.5	**1.5**	**1.6**	1.5	**1.8**	**1.8**
Llama3.3	1.4	1.4	1.4	1.4	1.6	1.6
Mistral3	1.2	1.2	1.3	1.2	1.4	1.3
Qwen2.5	1.3	1.3	1.3	1.4	1.5	1.6
R1-D	1.5	**1.5**	1.5	1.5	1.7	1.7
R1	1.5	**1.5**	**1.6**	**1.6**	**1.8**	**1.8**

Table 12: Average Pun Pair Agreement (PPA) measured on the true positive examples, in [0–2], for each prompt setting (W, W+S) across datasets. Standard deviation is 0.0 for all measurements, and the best results are marked in bold.

errors made by the LLMs in each category. Our analysis indicates that the most frequent error in the rationales is the identification of double meanings that lack support from the surrounding context (`Context`). For instance, the phrase "*Long fairy tales have a tendency to wyvern*" was incorrectly interpreted as a play on words between *wyvern* (a mythical dragon-like creature) and *wizen* (to become dry or shriveled). Although the meanings of the words are correctly matched (so it is not a `Word-sense pair` error), there are no contextual clues to support the use of *wizen*. Additionally, the annotators could not recognize any similarity in spelling or pronunciation between the two words, highlighting a mistake in the `Pun pair` category. A third typical mistake is the incorrect pairing of words to their meanings (`Word-sense pair`). For example, in the non-pun phrase "*Her decision to take up rock climbing was preset-pit-toes to say the least*" Llama incorrectly interpreted the expression *preset-pit-toes* as "ready to jump or spring into action" apparently referring back to the original pun word "precipitous" and ignoring the different spelling.

Examples of all the error categories are shown in Table 13, along with human annotators' questionnaire responses. For brevity, the questionnaire items shown in the table have been slightly shortened; the full questionnaire, guidelines, and annotation examples are attached at the end of this document.

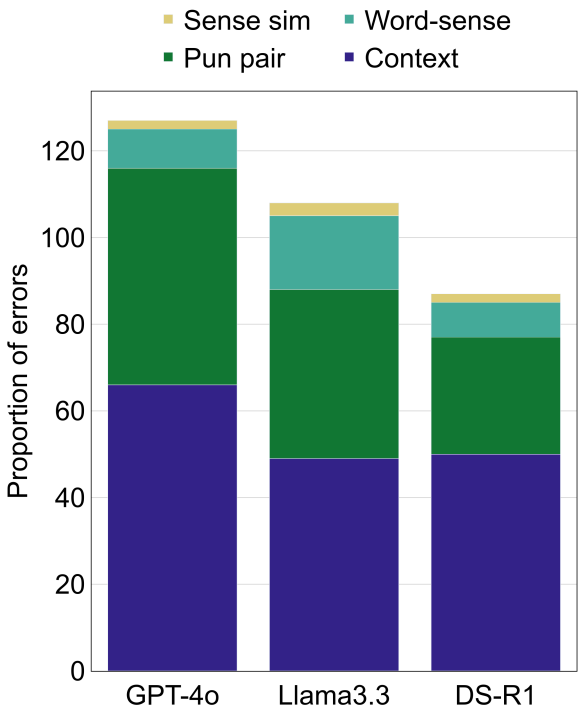

Figure 15: Frequency of LLMs' mistakes in the generated rationales.

Input	Output	Answers by annotators	Errors
		GPT-4o	
German sausage jokes are just the sausage	yes <the sausage> <the wurst> <the main part or essence of something> <a type of German sausage>	• A. Can "the sausage" be interpreted as "the main part or essence of something"? **No** • B. Can "the wurst" be interpreted as "a type of German sausage"? Yes • E. Does the context support the interpretation of "the wurst" as "a type of German sausage"? Yes • FG. Are "the sausage" and "the wurst" orthographically or phonetically close? **No** • H. Rate pun quality in $0-5$: 1	`word-sense` $(A\&B)$, `pun pair` $(F\|G)$
Those who make magnets can be quite charming	yes <charming> <charming> <attractive or pleasing> <using magic>	• A. Can "charming" be interpreted as "attractive or pleasing"? Yes • B. Can "charming" be interpreted as "using magic"? Yes • C. How diverse are the two previous senses in $0-5$? 2 • D. Does the context support the interpretation of "charming" as "using magic"? **No** • E. Does the context support the interpretation of "charming" as "attractive or pleasing"? Yes • H. Rate pun quality in $0-5$: 0	`context` $(D\&E)$
		Llama3.3	
The sailor called his girlfriend's belly ring an oven destroyer	yes <oven destroyer> <over destroyer> <a device that ruins an oven> <one who destroys from above>	• A. Can "oven destroyer" be interpreted as "a device that ruins an oven"? Yes • B. Can "over destroyer" be interpreted as "one who destroys from above"? **No** • D. Does the context support the interpretation of "oven destroyer" as "a device that ruins an oven"? **No** • FG. Are "oven destroyer" and "over destroyer" orthographically or phonetically close? Yes • H. Rate pun quality in $0-5$: 0	`word-sense` $(A\&B)$, `context` $(D\&E)$
A circus lion won't eat clowns because they taste humorous	yes <humorous> <humorous> <having or showing a sense of humor> <having a funny or unusual taste>	• A. Can "humorous" be interpreted as "having or showing a sense of humor"? Yes • B. Can "humorous" be interpreted as "having a funny or unusual taste"? Maybe • C. How diverse are the two previous senses in $0-5$? **1** • D. Does the context support the interpretation of "humorous" as "having or showing a sense of humor"? Yes • E. Does the context support the interpretation of "humorous" as "having a taste, especially a funny or unusual one"? Yes • H. Rate pun quality in $0-5$: 1	`sense sim` (C)
		DeepSeek-R1	
If a dog was computer literate, would his bark be worse than his octet?	yes <octet> <bite> <a group of eight bits in computing> <the act of using teeth to cut or pierce>	• A. Can "octet" be interpreted as "a group of eight bits in computing"? Yes • B. Can "bite" be interpreted as "the act of using teeth to cut or pierce"? Yes • C. How diverse are the two previous senses in $0-5$? 4 • D. Does the context support the interpretation of "octet" as "a group of eight bits in computing"? **No** • E. Does the context support the interpretation of "bite" as "the act of using teeth to cut or pierce"? **No** • FG. Are "octet" and "bite" orthographically or phonetically close? **No** • H. Rate pun quality in $0-5$: 0	`context` $(D\&E)$, `pun pair` $(F\|G)$
Why are frogs so happy? They eat whatever feather them	yes <feather> <fed them> <to cover with feathers> <to provide food for them>	• A. Can "feather" be interpreted as "to cover with feathers"? **No** • B. Can "fed them" be interpreted as "to provide food for them"? Yes • E. Does the context support the interpretation of "fed them" as "to provide food for them"? Yes • FG. Are "feather" and "fed them" orthographically or phonetically close? Yes • H. Rate pun quality in $0-5$: 2	`word-sense` $(A\&B)$

Table 13: Examples of error-analysis results for the three models analyzed in RQ3. The last column lists the errors derived from annotators' answers and used in the final statistics. The logical expression in brackets indicates how each error was computed; see §A.3.1 for details. Questions that were not relevant to the sample or were unanswered are omitted. Questions *F* and *G* are combined for brevity. Answers marked in red indicate the model's errors.

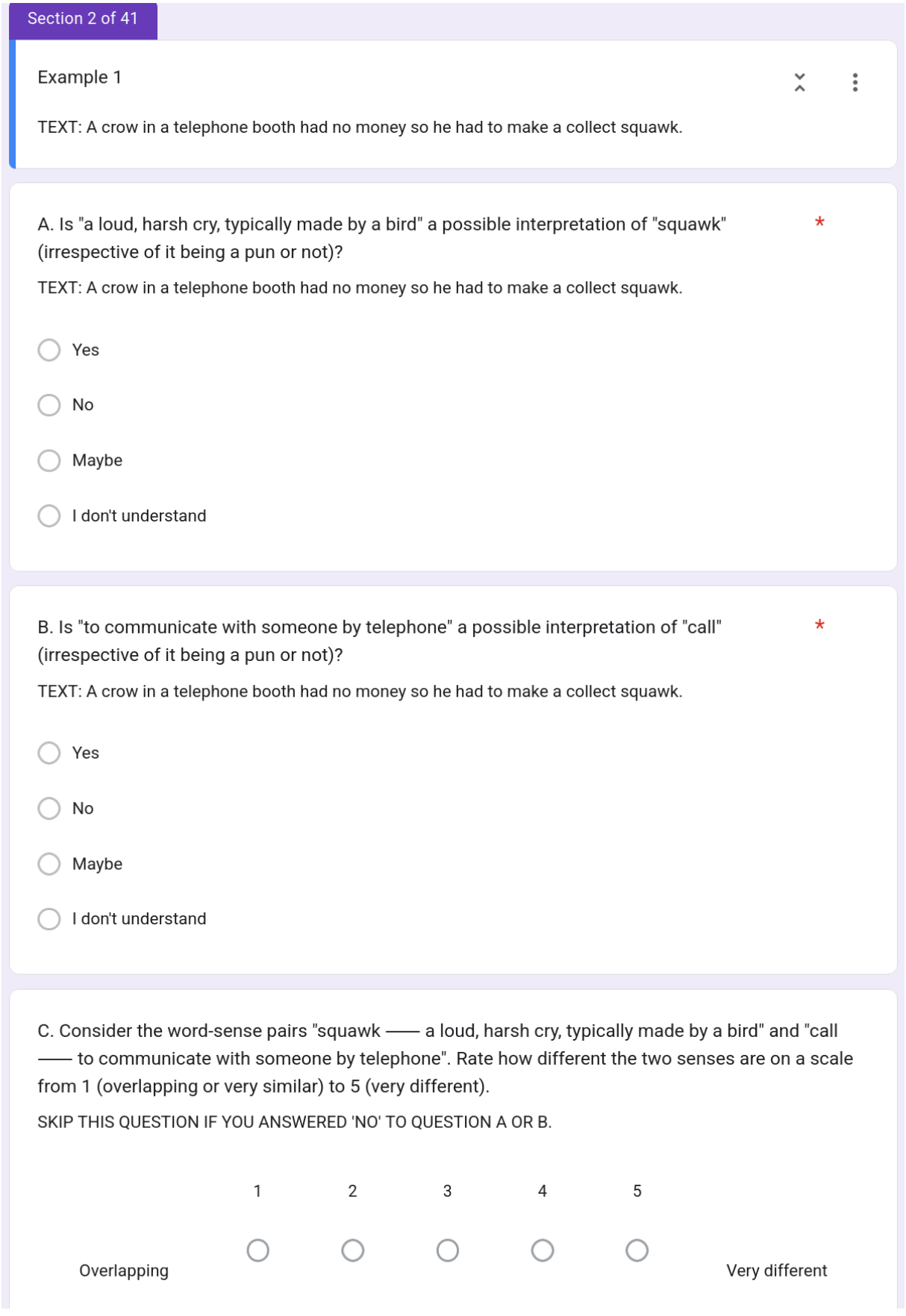

Figure 16: Questions A-B-C utilized in the Error Analysis.

D. Is there at least one contextual clue in the text (other than the expression itself) that supports the interpretation of "squawk" as "a loud, harsh cry, typically made by a bird" (irrespective of it being in a pun or not)?

SKIP THIS QUESTION IF YOU ANSWERED 'NO' TO QUESTION A.
TEXT: A crow in a telephone booth had no money so he had to make a collect squawk.

- ○ Yes
- ○ No
- ○ Maybe
- ○ I don't understand

E. Is there at least one contextual clue in the text (other than the expression itself) that supports the interpretation of "call" as "to communicate with someone by telephone" (irrespective of it being in a pun or not)?

SKIP THIS QUESTION IF YOU ANSWERED 'NO' TO QUESTION B.
TEXT: A crow in a telephone booth had no money so he had to make a collect squawk.

- ○ Yes
- ○ No
- ○ Maybe
- ○ I don't understand

F. Can you recognize a relationship or similarity between "squawk" and "call" in terms of phonetics (pronunciation)? *

TEXT: A crow in a telephone booth had no money so he had to make a collect squawk.

- ○ Yes
- ○ No
- ○ Maybe
- ○ I don't understand

Figure 17: Questions D-E-F utilized in the Error Analysis.

G. Can you recognize a relationship or similarity between "squawk" and "call" in terms of orthography (spelling)? *

TEXT: A crow in a telephone booth had no money so he had to make a collect squawk.

- ○ Yes
- ○ No
- ○ Maybe
- ○ I don't understand

H. Do you think the text contains a pun involving "squawk" and "call"? If you think no pun is present, answer 0. If you think there is a pun, rate the quality of the pun from 1 (very poor) to 5 (very effective). If you don't know, skip this question.

TEXT: A crow in a telephone booth had no money so he had to make a collect squawk.

	0	1	2	3	4	5	
Not a pun	○	○	○	○	○	○	Very effective pun

I. Do you think the text contains any puns involving words or expressions NOT previously mentioned? If so, write down the words or expressions that produce the pun, separated by commas. Otherwise, skip this question.

Example of answer: I am expression1, expression2
TEXT: A crow in a telephone booth had no money so he had to make a collect squawk.

Short-answer text

Figure 18: Questions G-H-I utilized in the Error Analysis.

Annotator's Guidelines

Pun Explanation - Error Analysis

Task Introduction

Welcome! Thank you for participating in this study.

Your responses will contribute to an **error analysis** that examines how Artificial Intelligence (AI) explains the use of puns in short English texts.

Please **read the following sections carefully** before proceeding to the questions.

Background

Puns are a type of wordplay that exploits words with multiple meanings or similar-sounding words for humorous or rhetorical effect. The *pun effect* is created by the greatly accelerated perception of two different senses of an expression.

For example, in the sentence "*You lamb! said Tom sheepishly*", there is a pun because "sheepishly" can be interpreted as "in an embarrassed manner" while also evoking "sheep" referring to the "woolly ruminant mammal".

Another example is "*The plot to his story of the pond was quite shallow*", where the word "shallow" can be interpreted in the sense of "lacking physical depth" and "lacking depth of intellect or knowledge".

The first sentence contains a **heterographic pun**, as "sheepishly" and "sheep" are different words. The second example features a **homographic pun**, where the same word "shallow" is used in two different senses.

In our study, we asked an AI agent to analyze short texts, some containing puns and some not, and to identify the words and senses that create the *pun effect*.

In the following quiz, you will see a selection of non-puns that the AI agent classified as puns. Your job is answering 9 questions to help us evaluate how wrong the AI agent's answers are.

Definitions

- ***Expression***. An expression can be a single word, such as "car" or "beach", or it can contain multiple words. Examples of these include phrasal verbs such as "put up with", or polywords such as "ice cream", and "run-through".
- ***Sense***. A meaning or interpretation of an expression. By *interpretation* we intend a legitimate reading of an expression that is not necessarily its definition. For example, two interpretations of "parrot" are (1) "*a species of tropical birds*", (2) "*a play on word on 'carrot'*" (this interpretation can be reasonable in certain contexts because the two words are similar). Conversely, the sense "*a colorful dress*" would not be considered a valid interpretation.
- ***Support***. In this evaluation, we discuss the concept of support in relation to expressions and their senses. Polysemous expressions can only be disambiguated if the context supports the intended meaning. We say that a pun is "supported" if some contextual clues in the sentence facilitate the perception of the two senses that create the pun effect. For instance, consider the previous pun "*The plot to his story of the pond was quite shallow*". What would happen if we replace "pond" with "mountain"?

  The plot to his story of the **mountain** *was quite shallow*.

  Now, the sense "*lacking physical depth*" cannot be perceived anymore, as the context does not support it. In the original text, "pond" evokes this specific meaning, but words related to "water", such as "lake" or "river," could also work.
- ***Relationship between words***. You will be asked to identify relationships in terms of *orthography* (spelling) or *phonetics* (sound).
  For example, in the previous heterographic pun, the words "sheepishly" and "sheep" share the same root, indicating an *orthographic and phonetic similarity*. Additionally, the words "sheep" and "ship" can also be considered similar in both regards.
  In homographic examples, you may be asked to rate the relationship/similarity

between the same word (*e.g.*, between “shallow” and “shallow”). In such cases, the similarity is trivially present, as the two words are identical.

Instructions

Each page of this survey will present a short text, which was not intended to contain a pun. You should answer the 7-9 questions related to the text, following these guidelines:

- Most questions require a ***Yes*** or ***No*** answer, but some will ask you to provide a rating on a **scale**;
- If you're unsure, you can choose ***Maybe*** if the answer depends on additional information;
- If you do not understand (some parts of) the text or the question, select ***I don't understand***;
- Some questions can be skipped depending on your answers to previous questions. These questions will be clearly marked, and we ask you to **ONLY SKIP** them **if the condition is met**.
  However, you can always choose to answer these questions if you wish; skipping them is only intended to help speed up the process;
- We expect you to spend around 2 minutes per page.

Keep in mind that:

- You will find the original text repeated under most questions, allowing you to quickly access it if needed;
- The 7-9 questions are always presented in the same order, which should help you speed up the answering process a little bit;
- You are required to sign in with your Google account in order to save your progress. This way, you can stop the annotation and continue it later, provided that you are **connected to the Internet** (check before closing your browser!). Google will save your progress for up to 30 days;
- You can reach out to [ANONYMOUS] for any question or clarification.

Thank you for your participation!

Questions with Examples

Here is the list of questions asked for each short text, along with some examples of possible answers.
To help clarify the task, some answers include justifications for the responses in brackets [], but keep in mind that some answers might be subjective.
Please **read these examples carefully**.

The first two questions (A, B) are meant to determine if any expression (w_p'/w_a') is associated with a legitimate interpretation, regardless of whether each expression is part of a pun.

A. Is **s_p'** a possible interpretation of **w_p'** (irrespective of it being a pun or not)?

B. Is **s_a'** a possible interpretation of **w_a'** (irrespective of it being a pun or not)?

Examples:

- Is "a species of tropical birds" a possible interpretation of "parrot" (irrespective of it being a pun or not)?
    - Answer: **Yes**
- Is "one who imitates the words or actions of another, especially without understanding them" a possible interpretation of "parrot" (irrespective of it being a pun or not)?
    - Answer: **Yes** [this is a possible figurative interpretation of the word]
- Is "a colorful dress" a possible interpretation of "parrot" (irrespective of it being a pun or not)?
    - Answer: **No**

Question C asks to rate how different two senses are. This should be answered regardless of the presence of the two senses in the original text and their correct usage in the pun.

C. **[SKIP THIS QUESTION IF YOU ANSWERED "NO" TO QUESTION A OR B]**
Consider the word-sense pairs "w_p' —— s_p' " and "w_a' —— s_a' ".
Rate how **different** the two **senses** are on a scale from 1 (overlapping or very similar) to 5 (very different).

Examples:

- Consider the word-sense pairs "parrot —— a species of tropical birds" and "parrot —— one who imitates the words or actions of another ". Rate how

different the two senses are on a scale from 1 (overlapping or very similar) to 5 (very different).
    - Answer: **5** [totally different senses, one literal, one figurative]
- Consider the word-sense pairs "parrot —— a species of tropical birds" and "parrot —— a colorful animal primarily found in subtropical forests". Rate how different the two senses are on a scale from 1 (overlapping or very similar) to 5 (very different).
    - Answer: **1** [this is a bit subjective, but birds are animals, so these senses refer to the same thing with different words]

The next two questions (D, E) ask to contextualize an expression in the original text and decide if its sense is supported or not, irrespective of the expression being part of a pun.

D. **[SKIP THIS QUESTION IF YOU ANSWERED "NO" TO QUESTION A]**
Is there at least one contextual clue in the text (other than the expression itself) that supports the interpretation of $\mathbf{w_p'}$ as $\mathbf{s_p'}$ (irrespective of it being in a pun or not)?

E. **[SKIP THIS QUESTION IF YOU ANSWERED "NO" TO QUESTION B]**
Is there at least one contextual clue in the text (other than the expression itself) that supports the interpretation of $\mathbf{w_a'}$ as $\mathbf{s_a'}$ (irrespective of it being in a pun or not)?

Examples:

TEXT: *A parrot flies, a coconut rolls.*

- Is there at least one contextual clue in the text (other than the expression itself) that supports the interpretation of "parrot" as "a species of tropical birds"?
    - Answer: **Yes** [the verb "flies" suggests we are talking about a bird; "coconut" also suggests a tropical setting, which fits the definition of the bird]
- Is there at least one contextual clue in the text (other than the expression itself) that supports the interpretation of "parrot" as "one who imitates the words or actions of another, especially without understanding them"?
    - Answer: **No** [there is nothing in the text that hints at this human behavior]

The next two questions (F, G) ask if you can recognize any similarity between two expressions, irrespective of how the expressions are used in the text. Additionally, these questions are not shown for some examples.

F. Can you recognize a relationship or similarity between $\mathbf{w_p'}$ and $\mathbf{w_a'}$ in terms of *phonetics* (pronunciation)?

G. Can you recognize a relationship or similarity between **w_p'** and **w_a'** in terms of *orthography* (spelling)?

Examples:

TEXT: *A parrot flies, a coconut rolls*.

- Can you recognize a relationship or similarity between "parrot" and "barrow" in terms of *phonetics* (pronunciation)?
    - Answer: **Yes** [however, this can be subjective]
- Can you recognize a relationship or similarity between "parrot" and "carrot" in terms of *orthography* (spelling)?
    - Answer: **Yes**

Finally, the last questions (H, I) ask you to decide if a pun is present in the text or not.

H. Do you think the text contains a pun involving **w_p'** and **w_a'** ?

If you think no pun is present, answer 0. If you think there is a pun, rate the quality of the pun from 1 (very poor) to 5 (very effective). If you don't know, skip this question.

I. Do you think the text contains any puns involving words or expressions NOT previously mentioned?

If so, write down the words or expressions that produce the pun, separated by commas. Otherwise, skip this question.

Examples:

TEXT: *Time flies like an arrow, fruit lies like a banana*.

- Do you think the text contains a pun involving "lies" and "flies" ?
  If you think no pun is present, answer 0. If you think there is a pun, rate the quality of the pun from 1 (very poor) to 5 (very effective). If you don't know, skip this question.
    - Answer: **0** [the word "lies" does not make any sense in the sentence, so the pun does not work]

TEXT: *What do you call a movie about a parrot mocking a topic? A parrot-y*.

- Do you think the text contains a pun involving "parrot-y" and "parody" ?
  If you think no pun is present, answer 0. If you think there is a pun, rate the quality of the pun from 1 (very poor) to 5 (very effective). If you don't know, skip this question.
    - Answer: **5** ["pardod-y" and "parroty" both make sense in the sentence, so there is a good pun]