Phonetic Cues Improve LLM-Based Pun Detection in Short Text

Adith Santosh Thaniserikaran
Purdue University
athanise@purdue.edu

Govind Harikrishnan
Purdue University
gharikr@purdue.edu

Abstract

This paper studies joke detection in short text, focusing only on jokes triggered by lexical ambiguity. Following Attardo and Raskin, we treat these jokes as cases where humor arises from a script opposition activated through a logical mechanism such as homography or homophony. Our framework combines contextual semantic analysis for homographs with phoneme-level similarity for homophones and near-homophones, using CMUdict, weighted Levenshtein distance, and prompt-based reasoning to recover ambiguities that are not visible in spelling alone. Results show that explicit phonetic modeling improves detection of sound-based puns.

1 Introduction

Automatic humor detection has attracted increasing attention in computational linguistics, but recent work suggests that humor understanding remains highly sensitive to the specific linguistic mechanism involved. Studies of caption ranking, pun-focused reasoning, stand-up transcript analysis, and LLM-based humor evaluation show that current systems still struggle when humor depends on structured ambiguity rather than broad surface cues or general fluency (Zhou et al., 2025; Cocchieri et al., 2025; Romanowski et al., 2025; Goes et al., 2023). In particular, pun-centered work highlights that recognizing verbal humor requires models to recover the specific linguistic relation that licenses the joke, rather than merely detect that a text appears playful or surprising (Cocchieri et al., 2025). The core problem addressed in this paper is that existing humor detection systems do not reliably identify puns whose humor depends on different forms of ambiguity, especially when the relevant cue is phonetic rather than purely orthographic or semantic. A model may detect that a sentence is unusual or joke-like without correctly recovering the ambiguity that actually produces the humorous effect. This makes pun detection difficult because successful analysis requires the system to determine not just whether a text is humorous, but what specific linguistic mechanism supports that humor. This paper therefore focuses narrowly on ambiguity-driven pun detection in short text, especially cases where humor arises from either lexical-semantic ambiguity or phonetic similarity. This focus is also motivated by humor theory: following Attardo and Raskin's General Theory of Verbal Humor, verbal jokes can be analyzed in terms of script opposition and logical mechanism, making puns a particularly suitable subtype for computational study because their humor often depends on a clearly identifiable ambiguity trigger (Attardo and Raskin, 1991). Later work on logical mechanisms further reinforces this view by treating ambiguity, analogy, and partial incongruity resolution as central to how verbal jokes are structured and interpreted (Hempelmann and Attardo, 2011; Attardo et al., 2002). To address this problem, we present a combined system for detecting pun-based humor from both written and sound-based cues. The framework integrates lexical preprocessing and part-of-speech filtering, phoneme extraction with the CMU Pronouncing Dictionary (Carnegie Mellon University, 1998), a weighted Levenshtein metric (Levenshtein, 1966) for phonetic similarity, GPT-5.4-based ambiguity detection and phonetic cue augmentation for ambiguity reasoning. Rather than treating humor detection as only a binary joke/non-joke task, the system aims to identify the linguistic mechanism underlying the humor and determine whether the ambiguity is meaningful in context. In this way, the proposed framework is aligned with recent calls for mechanism-aware humor analysis in both computational and theoretical work (Cocchieri et al., 2025; Romanowski et al., 2025; Attardo and Raskin, 1991).

Proceedings of the 2nd Workshop on Computational Humor (CHum 2026), pages 72–80

2 Related Work

Related work on computational humor in our paper is most relevant in three areas: humor understanding with large language models, pun-specific evaluation, and humor theory. Recent LLM studies show that humor remains difficult when success depends on recovering the mechanism that makes a text funny, rather than relying on broad fluency or surface plausibility (Zhou et al., 2025; Romanowski et al., 2025; Goes et al., 2023). In particular, work on pun-focused benchmarking shows that current models often recognize apparent wordplay without reliably verifying whether the ambiguity is actually valid, especially in misleading or structurally constrained cases (Cocchieri et al., 2025). This challenge is closely tied to humor theory. In the General Theory of Verbal Humor, verbal jokes are analyzed in terms of script opposition and logical mechanism, and later work argues that ambiguity and partial incongruity resolution are central to how many jokes are structured (Attardo and Raskin, 1991; Attardo et al., 2002; Hempelmann and Attardo, 2011). These theoretical accounts also motivate separating pun-based humor into homographic and homophonic types, since they rely on different forms of ambiguity and therefore may require different computational treatments. Our work builds on these threads by focusing narrowly on short-text puns whose humor depends on lexical-semantic ambiguity, phonetic similarity, or both; we treat weighted Levenshtein distance as a lightweight cueing heuristic informed by prior phonetic-similarity and phonological-alignment work (Kondrak, 2000; Fontan et al., 2016), rather than as a novel phonological similarity metric. Unlike prior LLM-based humor studies that evaluate broader humor understanding or judgment (Zhou et al., 2025; Goes et al., 2023; Romanowski et al., 2025), we target mechanism-level detection of pun-based humor and explicitly model the ambiguity that licenses the joke.

3 System Overview

The proposed system is a hybrid humor-detection pipeline that combines rule-based phonetic analysis with GPT-5.4-based semantic reasoning and classification (OpenAI, 2026). It is designed to detect pun-based humor by modeling two major sources of ambiguity: *phonetic ambiguity*, which underlies homophone-based jokes, and *semantic ambiguity*, which underlies homograph-based jokes. Rather than relying only on an end-to-end language model prediction, the system first extracts linguistically motivated cues and then uses GPT-5.4 to interpret them in context. At a high level, each input sentence passes through a sequence of processing stages. The text is first preprocessed through tokenization, content-word filtering, part-of-speech tagging, and lemmatization. These steps reduce noise and prepare the sentence for both symbolic phonetic analysis and semantic ambiguity detection. The phonetic module then consults the CMU Pronouncing Dictionary to obtain candidate pronunciations for relevant tokens. Using these phoneme representations, the system computes pairwise similarity with a custom weighted Levenshtein distance that accounts for vowel–vowel, consonant–consonant, and vowel–consonant substitutions differently. Word pairs whose normalized similarity exceeds a predefined threshold are treated as candidate homophone or near-homophone cues. When a strong phonetic match is found, the system augments this information directly into the classification prompt. This phonetic cue augmentation step makes hidden sound-based ambiguity explicit for GPT-5.4, enabling the model to reason about puns that may not be obvious from spelling alone. In this way, the phonetic module serves as a symbolic front end that supplies interpretable evidence to the language model. If no strong phonetic ambiguity is detected, the system invokes GPT-5.4 in a separate semantic-analysis stage to identify possible homographs or words with multiple contextual meanings. The model is prompted to return structured output describing any such ambiguity, allowing semantic cues to be incorporated into downstream classification. Finally, GPT-5.4 performs the joke classification step using the sentence together with any detected phonetic or semantic cues. The system outputs a binary prediction (*Joke* or *Non-joke*) along with a short natural-language explanation of the reasoning behind the decision. If the initial explanation is too brief or underspecified, a secondary prompt is used to elicit a clearer justification. This makes the overall pipeline not only predictive but also auditable, since each decision can be traced back to explicit phonetic or semantic evidence.

3.1 Humor-Theoretic Motivation

The system is also grounded in humor theory, particularly incongruity-based accounts of verbal humor (Attardo and Raskin, 1991; Hempelmann and

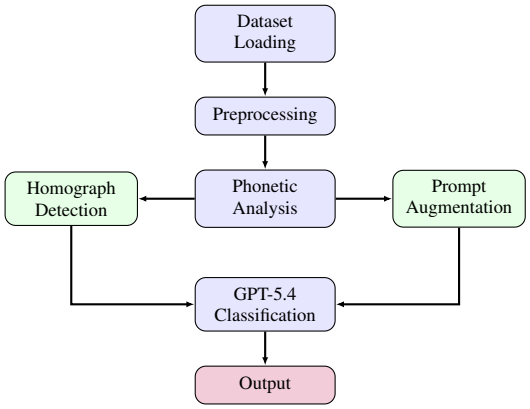

Figure 1: Compact block diagram of the humor detection system.

Attardo, 2011). In both homophone-based and homograph-based jokes, humor arises when a sentence supports more than one plausible interpretation, creating an incongruity between an initial reading and a later, competing one. The joke becomes successful when the listener or reader recognizes this hidden ambiguity and reinterprets the utterance accordingly. In this sense, pun-based humor is not merely a surface phenomenon of unusual words, but a structured interaction between ambiguity, incongruity, and partial resolution in context.

4 Methodology

Our humor-detection pipeline integrates rule-based phonetic modeling, semantic ambiguity resolution, and large language model reasoning. The system is modular, with each component responsible for capturing a distinct linguistic signal associated with humor. Figure 1 provides an overview of the full architecture.

4.1 Text Pre-processing

We begin by normalizing the input text through standard NLP operations, including tokenization, lowercasing, and punctuation removal. Using NLTK (Loper and Bird, 2002), we filter out stopwords and retain only *content-bearing* tokens (nouns, verbs, adjectives, adverbs). Each surviving token is lemmatized and POS-tagged to support downstream phonetic and semantic analysis. This step reduces noise and ensures that ambiguity detection concentrates on linguistically meaningful words.

4.2 Phoneme Mapping and Phonetic Modeling

A central component of the extended humor-detection system involves modeling sound similarity between words. Because written text does not directly encode pronunciation, the system must derive phonetic structure before it can detect homophones or near-homophones. To accomplish this, we rely on the CMU Pronouncing Dictionary (CMUdict), a well-established phonetic lexicon widely used in speech and computational linguistics research. The CMUdict provides pronunciations for more than 134,000 English words and maps each entry to its canonical ARPABET phoneme sequence, a standardized set of phonetic symbols. In our implementation, the system attempts to lookup using both the surface form of a token and its lemmatized form, ensuring coverage for inflected variants (e.g., cats $\rightarrow$ cat). If either the original token or its lemma appears in the CMUdict, its phoneme sequence is stored for further similarity analysis. All pronunciations are retrieved in ARPABET, a compact symbol set representing the phonemes of American English (e.g., CAT $\rightarrow$ K AE T). ARPABET is particularly useful because it disambiguates vowels and consonants clearly. Also, it provides a discrete sequence suitable for edit-distance metrics. This representation enables systematic comparisons between phoneme sequences, which is essential for identifying humor based on subtle sound similarities.

Why a Phoneme-Based Approach? Humor involving homophones and near-homophones depends on similar pronunciation, not spelling. A purely text-based model—such as one trained on embeddings—cannot reliably detect jokes like: "When the band broke up, it wasn't drama — it was just drum!" In orthographic form, drum and drama appear unrelated, yet acoustically they share strong phonetic overlap. A phonetic ambiguity often drives the humor mechanism, and without phoneme-level modeling, these jokes appear semantically incoherent. Therefore, phoneme-based modeling is crucial. Thus, the phonetic layer provides the foundation for identifying sound-driven ambiguity before handing contextual interpretation over to GPT-5.4.

4.3 Weighted Levenshtein Distance for Sound Similarity

After extracting ARPABET phoneme sequences from the CMU Pronouncing Dictionary, the system computes sound similarity using a custom *weighted Levenshtein distance* algorithm. Unlike the standard Levenshtein formulation which assigns a uniform cost to insertions, deletions and substitutions, our method incorporates phonetic knowledge to better model near-homophone similarity relevant to pun-based humor. To account for phonetic closeness, the substitution cost is determined by the phoneme classes of the segments involved:
1) 0 cost for identical phonemes.
2) .5 cost for vowel–vowel or consonant–consonant substitutions.
3) 1 cost for vowel–consonant substitutions.
Insertions and deletions follow the standard unit cost. This scheme captures coarse phonetic similarity by treating substitutions within the same broad phoneme class as less costly than substitutions across classes. For example, vowel–vowel substitutions such as AE $\rightarrow$ EH receive lower cost than vowel–consonant substitutions.

However, this weighting is intentionally coarse: it does not distinguish finer phonological features such as voicing, place of articulation, or manner of articulation. Therefore, the resulting similarity score should be interpreted as a lightweight heuristic for identifying candidate near-homophones, rather than as a full phonological similarity model. From a humor-theoretic perspective, this step is important because homophone-based puns rely on phonetic ambiguity between expressions that sound alike but differ in meaning. Such ambiguity creates the incongruity that underlies many forms of verbal humor: the listener initially interprets one form, then recognizes a competing sound-linked interpretation that shifts the meaning of the utterance. By identifying word pairs with strong phonetic overlap, the weighted Levenshtein module serves not only as a similarity measure but also as a computational mechanism for detecting one of the central triggers of pun-based humor.

Score Normalization Let P_1 and P_2 denote the phoneme sequences of two words. If $D_{weighted}(P_1, P_2)$ is the minimal weighted edit distance between them, the similarity score S is computed as:

$$S = 1 - \frac{D_{weighted}(P_1, P_2)}{\max(|P_1|, |P_2|)}. \quad (1)$$

This normalization bounds similarity in the interval $[0, 1]$, allowing consistent comparison across words of different lengths.

Homophone Threshold

Following empirical evaluation and linguistic validation from the project, a threshold of:

$$S > 0.8 \quad (2)$$

is used to identify homophones or near-homophones. This value was chosen because it reflects a high degree of phonetic similarity while still allowing for small pronunciation differences that commonly occur in near-homophonic word pairs. In practice, true homophones and plausible near-homophones tend to achieve scores close to 1, whereas unrelated pairs with only partial sound overlap usually fall below this range. As a result, the 0.8 cutoff provides a practical balance between retaining meaningful phonetic ambiguities and reducing false positives. Pairs exceeding this threshold are passed to the language model as candidate phonetic ambiguities, improving precision in the detection of sound-based puns. The weighted Levenshtein framework thus provides a linguistically motivated and computationally efficient mechanism for detecting phonetic similarity, serving as the core of the system's homophone-based humor recognition.

4.4 Homograph Detection via GPT-5.4

If no strong phonetic ambiguity is found, the system switches to detecting semantic ambiguity. Many jokes rely on homographs—words with multiple possible senses—whose interpretation depends on context. While lexical resources such as WordNet (Miller, 1995) contain sense inventories, they lack contextual reasoning, coverage for creative language, and the ability to decide which sense-shift is relevant to humor. To address this, we use GPT-5.4 as a context-aware semantic ambiguity detector. This stage also reflects humor theory, since homograph-based jokes depend on lexical ambiguity that activates multiple semantic scripts at once. Humor emerges when the surface reading of a word is suddenly replaced or enriched by another valid sense, producing an incongruous reinterpretation. GPT-5.4 is prompted to identify words in the sentence that could function as homographs and to list their alternative interpretations in a structured JSON format. This design ensures interpretability and easy downstream parsing. The exact prompt used in our system is shown below:

```
 You are a linguistic analyzer.
Find any word in the following sentence
that could have multiple meanings
(homograph pun).
List its possible interpretations
clearly.

Sentence: "SENTENCE_HERE"

Answer in JSON format like:
{"word": "flies", "meanings": ["move
quickly", "insects"]}
If no such word, return {"word": null}.
```

GPT-5.4 returns either a homograph candidate with its sense set or a null indicator. This structured output allows us to attach explicit semantic rationale to the joke classification stage. Because GPT-5.4 can leverage context, pragmatic cues, and world knowledge, we use it as a flexible context-aware alternative to dictionary-based sense lookup for humor-related ambiguity.

4.5 Phonetic Cue Augmentation

Phonetic ambiguity is not directly observable from written text, and large language models often fail to recognize homophone-based wordplay unless the relevant sound information is explicitly supplied. To address this, we use a phonetic cue augmentation mechanism: whenever the phonetic module detects a high-scoring homophone pair (similarity ≥ 0.80), the pair is programmatically inserted into the GPT-5.4 classification prompt.

In theoretical terms, this step helps the model recover an incongruity that may remain hidden in written form. Human listeners can often notice such ambiguity through pronunciation or contextual expectation, but text-only systems may miss it unless the latent phonetic relation is made explicit.

This augmentation provides GPT-5.4 with side information about the possible sound-based ambiguity, allowing the model to reason about humor mechanisms that would otherwise remain hidden. The augmented prompt explicitly highlights the homophone pair and requests a joke judgment using ambiguity-aware reasoning. The core prompt template is shown below:

```
 You are a humor detector.
This sentence likely relies on a
*homophone pun* (sound-based).
Analyze whether the text expresses
deliberate wordplay or simple absurdity.

Detected homophone pair: "W1" "W2"
Phonetic similarity = SCORE

Examples:
- "The chef couldn't bear it when the
bare bear stole his pie." → Joke
- "The knight trained every night before
the tournament." → Non-joke

Now classify the following:
"SENTENCE_HERE"
Answer:
```

The augmented pair narrows the model's search space by making the candidate sound-based relation explicit. This does not guarantee a correct humor judgment, but it provides the classifier with an interpretable cue that may otherwise be implicit in written text. Prompt augmentation therefore acts as a bridging layer between the rule-based phonetic module and the LLM classifier, allowing the system to combine deterministic similarity scores with contextual language-model reasoning.

4.6 Classification and Rationale Generation

The final stage integrates the outputs of the phonetic module, homograph analysis, and conditionally augmented prompts to perform humor classification. GPT-5.4 receives a structured prompt that includes (1) the input sentence, (2) any detected homophone pair or homograph information, and (3) examples of how ambiguity influences joke interpretation. GPT-5.4 then produces a label ("Joke" or "Non-joke") followed by a short natural-language explanation.

To maintain deterministic behavior, the system post-processes the model's response by extracting the text after the "Answer:" tag and reading the first line as the predicted label. Heuristic matching (e.g., checking for "joke" vs. "non-joke") ensures robustness to minor variations in formatting. The explanation is taken from subsequent lines containing keywords such as "reason" or "because". If the model provides no explanation or only a very short one (fewer than three words), the system automatically issues a secondary clarification prompt:

```
  You classified the text below as
"LABEL".
Now, explain your reasoning in 2–3
sentences with clear semantic logic.
Focus on ambiguity, wordplay, or
literalness.

Text: "SENTENCE_HERE"
```

This re-prompting mechanism ensures that each prediction is accompanied by a human-readable post-hoc rationale grounded in identifiable linguistic cues. We treat these explanations as useful decision summaries rather than faithful accounts of the model's internal reasoning. The final output includes the predicted label, the refined rationale, and any detected homophone or homograph cues,

creating an auditable record of the cues used by the pipeline.

5 Experimental Setup

5.1 Dataset

The evaluation was conducted on a subset of 500 short text instances drawn from the SemEval 2017 Task 7 pun detection dataset (Miller et al., 2017), consisting of 250 jokes and 250 non-jokes. The joke examples primarily contain pun-based humor, including both phonetic wordplay and semantic ambiguity, while the non-joke examples consist of proverbs, aphorisms, literal statements, and witty but non-humorous expressions. This balanced setup allows the system to be evaluated not only on its ability to detect humor, but also on its ability to distinguish genuine jokes from linguistically ambiguous yet non-humorous text.

Category	Count	Percentage
Joke	250	50.0%
Non-joke	250	50.0%
Total	500	100.0%

Table 1: Label distribution of the 500-example evaluation subset drawn from the SemEval–2017 Task 7 pun detection dataset.

5.2 LLM Configuration

All GPT-based components in the pipeline, including homograph detection, ambiguity reasoning, and final joke classification, were implemented using **gpt-5.4** (OpenAI, 2026). Unless otherwise specified, all model calls were executed with **temperature = 0.0** and a maximum of **160 completion tokens**. The zero-temperature setting was chosen to reduce output variability and encourage consistent classification behavior across runs, while the token limit was sufficient to allow both label prediction and brief natural-language justification without unnecessarily increasing generation length.

5.3 Task

The task is formulated as a binary classification problem: given an input sentence, the system predicts whether it should be labeled as *Joke* or *Non-joke*. In addition to the final label, the system also generates a short explanation describing the reasoning behind the prediction, allowing us to analyze whether the model relies on phonetic similarity, semantic ambiguity, or other contextual cues.

5.4 System Variants

We evaluate a hybrid humor-detection system together with several baselines and ablations in order to measure the contribution of each component.

Full system. The full system combines three sources of information: (1) direct LLM-based classification, (2) phonetic similarity cues derived from pronunciation-based matching, and (3) homograph or semantic ambiguity cues identified through language-model reasoning. These signals are integrated through prompt-guided reasoning to produce the final prediction.

Baselines. Two main baselines are considered:

- **plain_gpt**: a direct LLM classifier with no additional ambiguity-focused guidance.
- **ambiguity_prompt**: an LLM classifier explicitly prompted to focus on ambiguity and wordplay when making its prediction.

Ablation settings. To understand the contribution of each component, we evaluate the following ablations:

- **phonetic_only**: prediction based only on phonetic similarity cues.
- **no_prompt**: the hybrid system without the phonetic cue augmentation component.
- **no_homograph**: the hybrid system without homograph or semantic ambiguity reasoning.
- **no_phonetic**: the hybrid system without phonetic similarity cues.

These comparisons allow us to determine whether the full model truly benefits from the integration of symbolic and LLM-based signals, and which components are most responsible for performance gains.

5.5 Evaluation Metrics

We report *accuracy*, *macro F1*, and per-class F1 scores for the *Joke* and *Non-joke* classes. Accuracy provides an overall measure of correctness, while macro F1 gives equal weight to both classes and is therefore more informative in assessing balanced classification quality. Per-class F1 scores are included to show whether the system performs equally well on humorous and non-humorous instances, or whether it is biased toward one class.

5.6 Evaluation Goal

The primary goal of the evaluation is not only to measure overall classification performance, but also to determine whether phonetic and semantic ambiguity cues provide meaningful improvements over a plain LLM baseline. The ablation study is therefore central to the analysis, as it reveals which components contribute useful complementary information and which have limited practical impact in the current system design.

6 Results

Mode	Acc.	MF1	J-F1	NJ-F1
plain_gpt	0.710	0.692	0.767	0.617
ambiguity	0.684	0.655	0.755	0.556
phonetic_only	0.504	0.368	0.075	0.661
no_prompt	0.724	0.709	0.775	0.644
no_homograph	0.732	0.718	0.780	0.656
no_phonetic	0.610	0.551	0.714	0.389
full_system	**0.732**	**0.720**	**0.780**	**0.660**

Table 2: Performance of baseline and ablation variants on the 500-example evaluation set. MF1 = Macro F1, J-F1 = Joke F1, NJ-F1 = Non-joke F1.

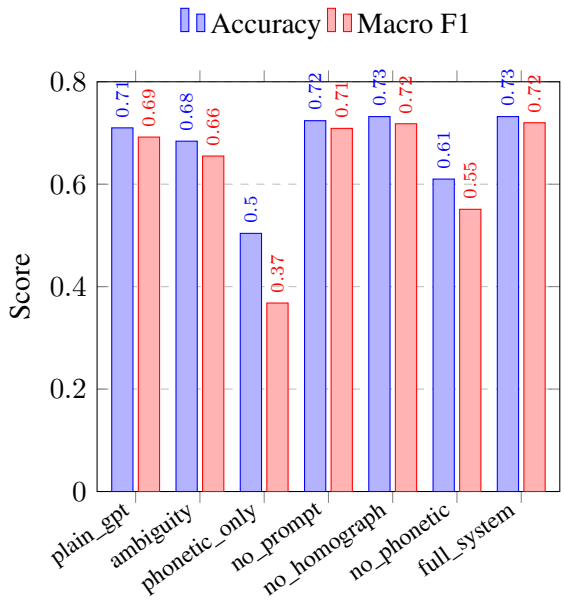

Figure 2: Accuracy and Macro F1 across baseline and ablation variants. The plot visually summarizes the same results reported in Table 2, with the largest drop occurring when phonetic cues are removed.

7 Discussion

The results show that the proposed hybrid system provides a *modest* but consistent improvement over the plain GPT baseline on the 500-example evaluation set. The full system achieves an accuracy of 0.732 and a macro F1 score of 0.720, compared with 0.710 accuracy and 0.692 macro F1 for the plain GPT setting. Although this gain is not large, it suggests that explicit linguistic cues can provide useful complementary information beyond direct LLM classification alone.

The ablation study provides a clearer picture of which components are responsible for this improvement. The strongest finding is the effect of removing phonetic information: the `no_phonetic` variant drops substantially to 0.610 accuracy and 0.551 macro F1. This indicates that phonetic similarity is the most important auxiliary signal in the current architecture. At the same time, the `phonetic_only` condition performs poorly overall, with an accuracy of 0.504 and a macro F1 score of 0.368. Taken together, these two results suggest that phonetic cues are not sufficient on their own, but they become highly valuable when combined with contextual reasoning from GPT-5.4. In other words, the phonetic module functions best as a complementary signal rather than as a standalone humor detector.

By contrast, removing the homograph module has almost no effect on overall performance. The `no_homograph` setting achieves 0.732 accuracy and 0.718 macro F1, which is nearly identical to the full system. This suggests that, in the current implementation, semantic ambiguity reasoning contributes far less than phonetic modeling. One possible explanation is that GPT-5.4 already captures some homograph-based ambiguity implicitly during classification, reducing the added value of a separate homograph-detection stage. Another possibility is that the semantic ambiguity module is currently too weak or too loosely integrated to provide a measurable benefit.

The role of prompt design is also informative. The `ambiguity_prompt` baseline performs worse than the plain GPT baseline, indicating that simply directing the model to attend to ambiguity does not reliably improve humor classification. In fact, making ambiguity too salient may encourage the model to over-predict humor in sentences that are merely proverb-like, ironic, or stylistically unusual. The `no_prompt` ablation performs slightly worse than the full system, suggesting that phonetic cue augmentation contributes a small but positive effect. This indicates that prompt steering is more useful when tied to explicit phonetic evidence than when applied as a general instruction.

An additional pattern emerges from the class-wise results. The full system achieves a much

higher recall for *Joke* instances than for *Non-joke* instances, indicating that the model remains biased toward predicting humor. This means that many non-joke examples are still misclassified as jokes, especially when they contain lexical ambiguity, stylistic oddity, or witty phrasing. This is an important limitation of the current system and reflects a deeper challenge in computational humor: not all ambiguous language is humorous, and not all surprising language is intended as a joke. Distinguishing deliberate pun-based humor from aphorisms, ironic statements, and creative but non-humorous expressions remains difficult even for LLM-based systems.

Overall, these findings support a nuanced conclusion. The hybrid system does not dramatically outperform the plain LLM baseline, but it does show that phonetic modeling provides meaningful complementary value when combined with contextual reasoning. The experiments also reveal that the current gains are driven primarily by phonetic cues rather than by homograph-based semantic ambiguity modeling. Future work should therefore focus on improving non-joke discrimination, strengthening the semantic ambiguity component, and evaluating whether more targeted integration of homograph reasoning can yield larger and more balanced performance gains.

8 Limitations

A notable limitation of the current system is its inability to recognize *multi-word or cross-token homophones*. While the model performs reliably for single-word homophones and homographs, the phonetic analysis pipeline assumes that each candidate homophone corresponds to a single lexical token. During preprocessing, the system queries the CMU Pronouncing Dictionary only for individual words, and phonetic similarity is computed exclusively between isolated tokens.

As a result, the system does not evaluate whether a *sequence of words* may form a phonological unit whose pronunciation matches another lexical item. This limitation becomes apparent in jokes such as:

> *My therapist told me to express my feelings, so now I scream for ice cream.*

Here, the humor relies not on the single word "scream," but on the multi-word sequence "I scream," which phonetically overlaps with "ice cream." Because the system analyzes "I" and "scream" independently, it never constructs the combined phoneme sequence necessary to compare against the pronunciation of "icecream." Consequently, the weighted Levenshtein algorithm cannot identify the intended sound-based ambiguity.

Although GPT may still classify the sentence as a joke based on contextual reasoning, the phonetic ambiguity is not captured at the algorithmic level.

Addressing this limitation would require extending the phoneme extraction module to operate over character n-grams or token n-grams, enabling the system to capture a broader range of sound-based humor that arises beyond single word boundaries. Unlike APUN-Bench, which evaluates pun understanding from audio, our system only approximates sound-based ambiguity from written text using CMUdict, leaving prosody and pronunciation variation for future work (Su et al., 2026).

9 Conclusion and Future Work

This work presented a unified humor-detection framework for identifying both homograph-based and homophone-based wordplay in written text. The system integrates phoneme extraction using the CMU Pronouncing Dictionary, a weighted Levenshtein phonetic similarity metric, and contextual semantic analysis through GPT-5.4. By combining symbolic linguistic methods with large language models, it provides both reliable classification and interpretable explanations of the linguistic mechanisms underlying humor.

The evaluation results show that the pipeline can recognize semantic ambiguity, phonetic similarity, and context-dependent humor across jokes and non-jokes. Overall, the findings suggest that computational humor detection benefits from hybrid architectures that combine structured linguistic processing with LLM-based inference.

Future work could incorporate n-gram phoneme modeling to capture multi-word homophonic sequences, requiring phrase-level phoneme synthesis and similarity computation over variable-length units. The framework could also be extended to detect humor arising from other mechanisms, such as morphological ambiguity, metaphorical reinterpretation, and pragmatic incongruity.

References

Salvatore Attardo, Christian F. Hempelmann, and Sara Di Maio. 2002. Script oppositions and logical mech-

anisms: Modeling incongruities and their resolutions. *Humor*, 15(1):3–46.

Salvatore Attardo and Victor Raskin. 1991. Script theory revis(it)ed: joke similarity and joke representation model. *Humor*, 4(3–4):293–347.

Carnegie Mellon University. 1998. The CMU Pronouncing Dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Accessed 2026-05-12.

Alessio Cocchieri, Luca Ragazzi, Paolo Italiani, Giuseppe Tagliavini, and Gianluca Moro. 2025. "What do you call a dog that is incontrovertibly true? Dogma": Testing LLM Generalization through Humor. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22922–22937, July 27–August 1, 2025. Association for Computational Linguistics.

Lionel Fontan, Isabelle Ferrane, Jérôme Farinas, Julien Pinquier, and Xavier Aumont. 2016. Using phonologically weighted levenshtein distances for the prediction of microscopic intelligibility. In *Proceedings of Interspeech 2016*, pages 650–654, San Francisco, USA.

Fabricio Goes, Piotr Sawicki, Marek Grzes, Dan Brown, and Marco Volpe. 2023. Is gpt-4 good enough to evaluate jokes? Unpublished manuscript. Manuscript.

Christian F. Hempelmann and Salvatore Attardo. 2011. Resolutions and their incongruities: Further thoughts on logical mechanisms. *Humor*, 24(2):125–149.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. *arXiv preprint cs/0205028*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

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.

OpenAI. 2026. Gpt-5.4. https://developers.openai.com/api/docs/models/gpt-5.4. Accessed 2026-05-12.

Adrianna Romanowski, Pedro H. V. Valois, and Kazuhiro Fukui. 2025. From punchlines to predictions: A metric to assess llm performance in identifying humor in stand-up comedy. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 36–46. Association for Computational Linguistics.

Yuchen Su, Shaoxin Zhong, Yonghua Zhu, Ruofan Wang, Zijian Huang, Qiqi Wang, Na Zhao, Diana Benavides-Prado, and Michael Witbrock. 2026. Words at play: Benchmarking audio pun understanding in large audio-language models. *arXiv preprint arXiv:2603.18678*.

Kuan Lok Zhou, Jiayi Chen, Siddharth Suresh, Reuben Narad, Timothy T. Rogers, Lalit K. Jain, Robert D. Nowak, Bob Mankoff, and Jifan Zhang. 2025. Bridging the creativity understanding gap: Small-scale human alignment enables expert-level humor ranking in llms. *arXiv preprint arXiv:2502.20356*.