ExPUNations: Augmenting Puns with Keywords and Explanations

Jiao Sun[1*†] **Anjali Narayan-Chen**[2†] **Shereen Oraby**[2] **Alessandra Cervone**[2]
Tagyoung Chung[2] **Jing Huang**[2] **Yang Liu**[2] **Nanyun Peng**[2,3]
[1]University of Southern California
[2]Amazon Alexa AI
[3]University of California, Los Angeles
jiaosun@usc.edu
{naraanja,orabys,cervon,tagyoung,jhuangz,yangliud}@amazon.com
violetpeng@cs.ucla.edu

Abstract

The tasks of humor understanding and generation are challenging and subjective even for humans, requiring commonsense and real-world knowledge to master. Puns, in particular, add the challenge of fusing that knowledge with the ability to interpret lexical-semantic ambiguity. In this paper, we present the ExPUNations (ExPUN) dataset, in which we augment an existing dataset of puns with detailed crowdsourced annotations of keywords denoting the most distinctive words that make the text funny, pun explanations describing why the text is funny, and fine-grained funniness ratings. This is the first humor dataset with such extensive and fine-grained annotations specifically for puns. Based on these annotations, we propose two tasks: explanation generation to aid with pun classification and keyword-conditioned pun generation, to challenge the current state-of-the-art natural language understanding and generation models' ability to understand and generate humor. We showcase that the annotated keywords we collect are helpful for generating better novel humorous texts in human evaluation, and that our natural language explanations can be leveraged to improve both the accuracy and robustness of humor classifiers.

1 Introduction

Humor serves multiple purposes and provides numerous benefits, such as relieving anxiety, avoiding painful feelings and facilitating learning (Buxman, 2008). As a specific example of humor, the creative uses of puns, wordplay and ambiguity are important ways to come up with jokes (Chiaro, 2006). Pun understanding and generation are particularly challenging tasks because they require extensive commonsense and world knowledge to compose and understand, even for humans. Despite growing interest in the area, there are limited amounts of data available in the domain of humor understanding and generation.

Text	When artists dream in color it's a pigment of their imagination.
KWD	artists, dream, color, pigment, imagination.
NLEx	Pigments are non-soluble materials often used in painting, and pigment sounds like figment, which is something that is not real but someone believes it is.
Text	The man found something to catch fish, which was a net gain.
KWD	catch fish, net gain.
NLEx	This is a play on words. A "net gain" means an increase in revenue but here "net" refers to how a net is used to catch fish.

Table 1: Two examples of annotated Keywords (KWD) and Natural Language Explanations (NLEx) for puns in our dataset. The highlighted texts are annotated keywords that contribute to making the text funny.

Existing humor datasets are usually only annotated with binary labels indicating whether each sentence is a joke, pun, or punchline (Hasan et al., 2019; Weller and Seppi, 2019; Castro et al., 2018; Mittal et al., 2021). This is insufficient to benchmark models' ability to understand and generate novel humorous text, since hardly anything meaningful can be learned from such a sparse supervision signal and coarse-grained annotation.

To facilitate research on humor understanding and generation, we present the ExPUNations (ExPUN) dataset, in which we augment an existing dataset of puns from SemEval 2017 Task 7 (Miller et al., 2017) with detailed crowdsourced annotations of fine-grained funniness ratings on a Likert scale of one to five, along with keywords denoting the most distinctive words that make the text funny and natural language explanations describing why the text is funny (Table 1). In addition, we collect annotations indicating whether a person understands the sentence, thinks it is a pun, and finds

*Work done during Jiao's internship at Amazon.
†Both authors equally contributed to the paper.

Text	Be True to your teeth, or they will be false to you.	Drinking too much of a certain potent potable may require a leave of absinthe.
Understandable	[1, 1, 1, 1, 0]	[1, 1, 1, 1, 1]
Offensive/Inappropriate	[0, 1, 0, 0, 0]	[0, 0, 0, 0, 0]
Is a joke?	[1, 0, 1, 0, 0]	[1, 1, 1, 1, 1]
Funniness (1-5)	[2, 0, 1, 0, 0]	[3, 4, 2, 1, 2]
Natural Language Explanation (NLEx)	NLEx1: Talking about being true as in being real or they will be fake/false teeth. NLEx2: False teeth are something people who lose their teeth may have, and being true to your teeth may be a way of saying take care of them otherwise you'll lose them.	NLEx1: It's a pun that replaces the word absence with absinthe, which is notoriously strong alcohol. NLEx2: This is a play on words. Absinthe here represents the liquor by the same name but is meant to replace the similar-sounding "absence". Too much absinthe will make you ill.
Joke keywords (KWD)	KWD1: ["true", "teeth", "false"] KWD2: ["be true", "teeth", "false to you"]	KWD1: ["drinking", "leave of absinthe"] KWD2: ["drinking too much", "leave of absinthe"]

Table 2: Two examples with annotation fields that we collect. We use underline to mark the commonsense knowledge that people need in order to understand the joke.

the joke offensive or inappropriate. Since these tasks are all highly subjective, we collect multiple annotations per sample, and present a detailed agreement analysis. We believe our annotations can be used in many other applications beyond pun understanding and generation, such as toxicity detection.

The contributions of our work are threefold:

- We contribute extensive high-quality annotations for an existing humor dataset along multiple dimensions.[1]
- Based on the annotations, we propose two tasks, explanation generation for pun classification and keyword-conditioned pun generation, to advance research on humor understanding and generation.
- We benchmark state-of-the-art NLP models on explanation generation for pun classification and keyword-conditioned pun generation. Our experiments demonstrate the benefits of utilizing natural language keywords and explanations for humor understanding and generation while highlighting several potential areas of improvement for the existing models.

2 ExPUN Dataset

In this section, we describe our data annotation procedure, including details of the annotation fields and our assessment of the annotation quality.

2.1 Data Preparation

The original SemEval 2017 Task 7 dataset (Miller et al., 2017)[2] contains puns that are either homographic (exploiting polysemy) or heterographic (exploiting phonological similarity to another word). The dataset also contains examples of non-pun text. We sample 1,999 text samples from SemEval 2017 Task 7 as the basis for our humor annotation. [3]

2.2 Dataset Annotation

The annotated fields (AF) come in the order of:

AF_1 [*understandability*]: whether the annotator understands the text or not, regardless of whether they perceive it as funny.

AF_2 [*offensiveness*]: whether the annotator finds the text offensive or inappropriate.

AF_3 [*joke*]: whether the annotator thinks the text is intended to be a joke.

AF_4 [*funniness*]: rate the funniness on a Likert scale of 1-5, where 1 means very not funny and 5 means very funny.

AF_5 [*explanation*]: explain in concise natural language about why this joke is funny. More specifically, if external or commonsense knowledge is required to understand the joke and/or its humor, the annotator should include the relevant knowledge in the explanation. If the joke is a pun or play on words, they must provide an explanation of how the play on words works.

[1]Resources will be available at: `https://github.com/amazon-research/expunations`

[2]`https://alt.qcri.org/semeval2017/task7/`. The data is released under CC BY-NC 4.0 license (`https://creativecommons.org/licenses/by-nc/4.0/legalcode`).

[3]We sample 834 heterographic puns, 1,074 homographic puns and 91 non-puns.

AF_6 [*joke keywords*]: pick out (as few as possible) keyword phrases from the joke that are related to the punchline/the reason the joke is funny. We emphasize that phrases should be sparse and mainly limited to content words, can be multiple words long, and the keywords should be copied verbatim from the joke.

If an annotator rates the instance as not understandable, they will skip the rest of the annotation for that instance (AF_2-AF_6). In addition, if an annotator rates an example as not a joke, they can skip the rest of the annotation (AF_4-AF_6). Table 2 shows two examples in our dataset. The first example has two annotators who think the text is a joke, and therefore it has two explanations. In the second instance, all annotators unanimously agree it is a joke. Here, we sample two explanations from the original five. For both instances, we use underline to highlight the external commonsense knowledge in the explanation. If the joke is a play on words, the explanation also shows how the play on words works (e.g., the second joke). We show the full annotation guidelines, including calibrating examples, in Appendix A.

We crowdsourced 5 annotations per sample using a professional team of 10 dedicated full-time annotators within our organization. Before starting the task, we held a kick-off meeting with the team to explain the annotation guidelines in detail. We then conducted 3 pilot rounds for calibration and iteratively met with annotators, including more details and examples to address annotator questions.[4] Finally, we conducted 7 rounds of annotation, each with between 100-300 puns per round grouped into minibatches of 50 examples. Each sample in a minibatch was annotated by consistent subteams of 5 annotators. After receiving a completed batch of annotations, we manually examined their quality and provided feedback on any quality issues, redoing batches as necessary.

2.3 Dataset Statistics and Quality Control

We report overall dataset statistics in Table 3. For $AF_1 - AF_3$, we count the number of samples labeled positive by majority vote. For AF_4, we compute the average of all funniness scores, excluding blank annotations, and find that while annotators recognized most samples as jokes, they did not find them to be particularly funny. For AF_5 and AF_6, we compute lexical statistics of our explanations and keyword annotations and provide deeper analysis of these key annotation fields in Section 2.4.

	total	AF_1	AF_2	AF_3
# samples	1,999	1,795	65	1,449
AF_4: Avg. funniness				1.68
AF_5: Explanations				
total # explanations				6,650
avg. # explanations/sample				3.33
avg. # tokens/expl.				31.67
avg. # sentences/expl.				2.01
AF_6: Keyword phrases				
avg. # tokens/keyword phrase				1.33
avg. # keyword phrases/sample				2.09

Table 3: Overall stats for annotation fields in ExPUN.

We report inter-annotator agreement for all annotation fields in Table 4.[5] For fields AF_1-AF_4, we compute agreement using (1) the average of Cohen's kappa scores of each annotator against the majority vote, and (2) the average Spearman correlation between each pair of annotators. We find that annotators show moderate agreement when deciding if the given text is a joke (AF_3), but lower agreement on the task of understanding the text (AF_1) as well as the much more subjective task of rating how funny a joke is (AF_4). We also find weak average Spearman correlation between each pair of annotations for the subjective categories of offensiveness (AF_2),[6] whether the text is a joke (AF_3) and joke funniness (AF_4).

For the free text fields in AF_5 and AF_6, we compute averaged BLEU-4 (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) scores in a pairwise fashion. We treat each annotator's explanation (for AF_5) or list of keyword phrases joined into a string (for AF_6) as candidate text, with the remaining annotators' annotations as a set of references. We find high similarity between joke keyword annotations, suggesting that annotators identify similar spans of keyword phrases, and a lower degree of similarity between pun explanations.

2.4 Dataset Analysis

Explanations. As seen in Figures 1a and 1b, on average, samples are annotated with multiple explanations, and the explanations are lengthy, spanning multiple sentences, and lexically diverse (14,748

[4]See Appendix A.2 for more details on pilot round feedback.

[5]When computing agreement, we exclude the first 100 annotated samples, as these were used as a calibrating pilot.

[6]See Appendix A.3 for more details.

Annotation Field	κ	ρ	BLEU	MET.
AF_1: Understand (0/1)	0.40	0.16	-	-
AF_2: Offensive (0/1)	0.16	0.34	-	-
AF_3: Joke (0/1)	0.58	0.32	-	-
AF_4: Funny (1-5)	0.41	0.30	-	-
AF_5: Explain (Text)	-	-	0.18	0.30
AF_6: Keywords (Text)	-	-	0.58	0.74

Table 4: Agreement stats for annotated fields in the ExPUN dataset. We report averaged Cohen's κ and Spearman's ρ for numeric ratings ($AF_1 - AF_4$), and averaged BLEU-4 and METEOR for text fields ($AF_5 - AF_6$).

token vocabulary size, with 210,580 tokens overall). Figure 3 in Appendix B shows the distribution of the top 50 most frequent content-words in our explanations. The frequent use of *usually* and *often* indicate the explanation of commonsense knowledge, e.g., *thunder and lightning are usually present in a weather storm* or *"pain" means physical discomfort often felt by a hospital patient.* The most frequent words, *means* and *word*, indicate that annotators frequently provide word sense information as part of their explanations, while *sounds* frequently appears in explanations of heterographic puns. Each of these most frequent words comprise less than 2.8% of all tokens in the explanations, illustrating the rich diversity of our corpus. [7]

Keywords. As seen in Figures 1c and 1d, on average, keyword phrases in ExPUN, which are derived from the original puns, are short and sparse (5,497 token vocabulary size, with 27,820 tokens overall). This follows from our guidelines to annotate keywords concisely, focusing mainly on content words that are essential to understanding the joke. Table 5 shows two examples of pun keyword annotations in our dataset that showcase different annotation styles among annotators. For instance, one annotator may tend to select wordy keyword phrases that introduce unnecessary tokens, while another may omit salient keywords that other annotators mention. Aggregating these annotations among annotators to construct a single ground truth set of keyword phrases is therefore challenging because of differing annotation styles. The problem of merging keywords is further complicated because the keywords from different annotators are often not aligned well, as different annotators may annotate varying numbers of keyword phrases and

[7]We show an analysis of highly-frequent explanation templates, as well as unique and highly-informative templates, in Appendix B.

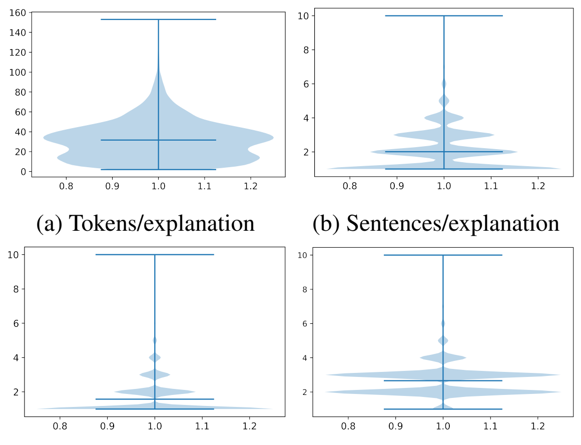

(a) Tokens/explanation (b) Sentences/explanation
(c) Tokens/keyword phrase (d) Keyword phrases/sample

Figure 1: Distributions of (a) number of tokens and (b) number of sentences in explanations (AF_5), (c) tokens in keyword phrases (AF_6), and (d) keyword phrases per sample. Horizontal lines are used to show the min, mean, and max values for each distribution.

different spans. Taking these considerations into account, we propose a keyword aggregation algorithm to address these issues and construct a single set of aggregated keywords per sample.

Keywords Aggregation. Algorithm 1 in Appendix C describes our keyword aggregation method. The algorithm aims to generate a comprehensive list of concise keywords for each sample. First, we compute a reliability score for each annotation, defined as the average of (# keyword phrases−# average tokens in each keyword phrase). The higher the score, the more comprehensive and concise the keywords from an annotator should be. We choose the annotator with the highest score to be the *anchor*. We note, however, that keyword annotations are not always error-free; e.g., in the first example of Table 5, w_4 has an incorrect word (*fancy chairs* instead of *royal chairs*). Therefore, for each keyword phrase, we compute the fuzzy matching score between the anchor's annotation with the rest of annotators' annotations. For each annotator, we keep the keyword phrase that has the highest fuzzy matching score with the anchor annotator's, with a minimum threshold score of 60. [8] This process produces a filtered keyword list where each of the remaining keyword phrases look similar to the anchor's. Then, we compute the average fuzzy matching score between the anchor's keyword phrase and each element in the filtered keyword list. We then choose the annotator with

[8]This is empirically determined.

	Royal chairs are rarely throne out.	**She didn't marry the gardener. Too rough around the hedges.**
w_1	[Royal chairs, throne out]	[didn't marry the gardener, too rough around the hedges]
w_2	[Royal chairs, throne out]	[didn't marry the gardener, rough around the hedges]
w_3	[Royal chairs, rarely throne out]	[didn't marry the gardener, rough around the hedges]
w_4	[fancy chairs, throne]	[gardener, rough, hedges]
w_5	[Royal chairs, throne]	[gardener, rough around the hedges]
w_A	[royal chairs, throne out]	[gardener, rough, hedges]

Table 5: Keyword annotations from different workers. w_A shows aggregated keywords from our algorithm.

the second-highest reliability score to be the anchor, and repeat the above process. Finally, by choosing the resulting keyword phrases that attain the maximum average fuzzy matching score between the first and second anchors, we get the final aggregated keywords for this instance.

3 Experiments

With the collected annotations, we propose two new tasks, pun explanation and keyword conditioned pun generation, to showcase novel tasks that our dataset uniquely enables and push the frontiers of NLU and NLG for humor. Note that the rich annotations in ExPUN can also enable many other interesting tasks such us pun keywords extraction, fine-grained funniness prediction, and others. However, we prioritize NLG tasks as they are relatively under-explored compared to NLU tasks. In this section, we benchmark current state-of-the-art models' performance on the proposed tasks.

3.1 Pun Explanation

The task of pun explanation takes a pun sentence as input and outputs a natural language explanation of why the pun is funny. This requires extensive understanding of background and commonsense knowledge. We hypothesize that existing NLP models would struggle to generate high-quality explanations for puns. On the other hand, high-quality explanations can improve humor understanding, and thus help tasks such as humor classification.

Formally, given text T, our target is to generate an explanation E_T of why T is funny. Additionally, we use the explanations to support the task of pun classification, where, given T (and optionally an explanation E_T), we output whether T is a joke.

Data Preparation. For each data sample, we use the longest human-written explanation from ExPUN (AF_5), substituting in the pun text if no explanations exist. [9] For pun classification, we assign output labels using the majority vote of AF_3 (is a joke). For both tasks, we split our dataset into 1,699/100/200 for train/dev/test. Dev and test contain an equal distribution jokes to non-jokes, while training contains 1,299 jokes and 400 non-jokes.

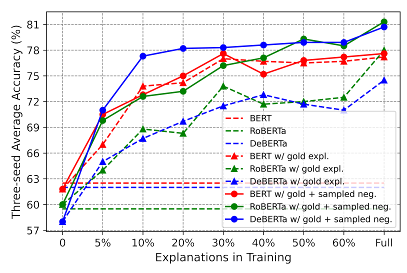
Evaluation Metrics. We do not report lexical overlap metrics as our primary evaluation metric for generated explanations because these are not suited for measuring plausibility (Camburu et al., 2018; Kayser et al., 2021; Clinciu et al., 2021) or faithfulness of explanations (Jacovi and Goldberg, 2020). Rather, we follow prior work and use the "*simulatability score*" metric from Wiegreffe et al. (2021) to measure explanation quality from the lens of usability of the explanation. It reflects the utility of explanations by measuring the improvement in task performance when explanations are provided as additional input vs. when they are not: acc(IE $\rightarrow$ O) $-$ acc(I $\rightarrow$ O), where I denotes the input text, E is the explanation and O is the classification of whether I is a joke. We evaluate how useful explanations can be by measuring the performance increase of acc(IE $\rightarrow$ O) as we increase the ratio of samples with explanations in the training data, and report acc(I $\rightarrow$ O) as a constant baseline that uses no explanations.

Models. We use the following model variations: [10]

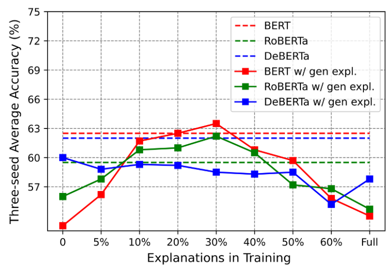
No explanations. As a baseline, we finetune BERT-base (Devlin et al., 2019), RoBERTa-base (Liu et al., 2019) and DeBERTa-base (He et al., 2021) to classify whether the given text is a joke without any explanations in the input.

Gold explanations. To find the upper bound of how useful explanations can be, we augment the input to the above baseline models with gold human-annotated explanations in both training and testing. The majority of non-punny examples (identified as unfunny by majority vote and thus labeled as unfunny) contain at least one explanation from an annotator who marked it as funny. In these cases,

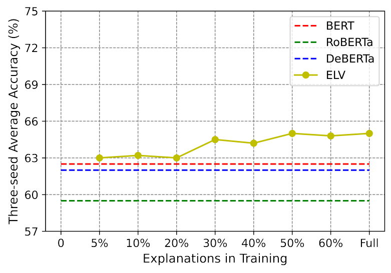
[9] Only 168 samples have no annotated explanations.
[10] Further experimental details in Appendix D.



(a) Gold explanations during test.



(b) Generated explanations during test.



(c) ELV model.

Figure 2: The impact of using human-written (2a) and model-generated explanations (2b and 2c) vs. no explanations (constant dotted lines) on pun classification accuracy. All reported numbers are computed with three-seed average. For each data point, we train a model on the full dataset, but only provide explanations for a given percentage, as shown on the x-axis.

we use any provided explanations as E, both in training and in testing with gold explanations. Otherwise, to construct training examples that have no annotated explanations, or where explanations are held out, we try two variants: (1) representing the missing explanation as an empty string (*"w/ gold expl."*), or (2) randomly sampling a negative explanation from another annotated example to use as input (*"w/ gold + sampled neg."*).

Generated explanations. Following previous work on explanation generation (Wiegreffe et al., 2021), we first finetune a T5 (Raffel et al., 2020) model to generate pun explanations given pun sentences as input. For text that contains no annotated explanations, we use the pun sentence itself as the output explanation. We then use gold human-annotated explanations to train and T5-generated explanations to test the explanation-augmented classification models.

ELV (Zhou et al., 2020a). ELV is a probabilistic framework for text classification where natural language **E**xplanations are treated as **L**atent **V**ariables. Two modules, an explanation generation module and an explanation-augmented prediction module are jointly trained using a variational EM framework. As another baseline, we train an ELV model for pun classification using the ExPUN dataset.

Results. We show our results on the pun classification task in Figure 2. Baseline performance of the *no explanations* models are shown using constant dotted lines. Figure 2a shows the upper bound of performance improvement when models are provided with *gold explanations*, indicating that human-written explanations are useful for this task, and that including more gold explanations in training data generally helps. In particular, adding randomly-sampled negative explanations ("*w/ gold + sampled neg.*") further improves the classification accuracy, showing the utility of our collected explanations in improving model performance. However, Figure 2b shows that using *generated explanations* at test time does not help to improve classification accuracy. Using the more carefully-designed ELV framework to jointly train the generation and classification modules shows improvement in classification accuracy (Figure 2c); however, qualitative analysis of the ELV explanations showed that many generated outputs are not fluent natural language, suggesting that performance improvements may stem more from modeling improvements as opposed to explanations. Given the huge improvements we see when incorporating gold explanations during test, we note explanations are clearly highly valuable if the quality of generated explanations can be improved.

Table 6 shows examples of T5-generated explanations for given puns. Qualitative analysis shows that generated explanations often identify the relevant pun word, and can include somewhat accurate word sense information for one sense of the pun. However, the model usually fails to explain the alternate word sense and its relation, which is crucial to understanding the wordplay. The model especially fails to explain phonological similarity in heterographic puns; e.g., in the first three examples, explanations fail to mention alternate words *carry*, *whet* and *humor*. For both pun types, our model can devolve into repetitively copying words from the input. Our results exhibit the challenge of generating good pun explanations and that high-quality explanations are useful for understanding humor.

3.2 Keyword-Conditioned Pun Generation

The task of keyword-conditioned pun generation takes human-annotated pun keywords as input and

Input Pun	**Generated Explanation**
My name is <u>Cary</u>. I'm a porter.	The joke is a pun on the word "cary". A porter is someone who transports goods.
Fishers often <u>wet</u> their appetite.	This is a play on words. The word "wet" means to wet your appetite, which is a characteristic of fish.
A gossip is someone with a great sense of <u>rumor</u>.	This is a play on words. The word "rumor" sounds like "rumor". A gossip is someone who has a great sense of rumor.
Oil executives are always using <u>crude</u> language.	The joke is a pun on the word "crude". Crude language is used to describe crude oil, which is a type of petroleum product.
Please mix me a martini, said Tom, <u>dryly</u>.	This is a play on words. The word "dryly" means dryly, but "dryly" sounds like "dryly".

Table 6: Pun explanations generated by the T5 model. We use <u>underline</u> to indicate the pun word in the input.

produces novel puns as output. This benchmarks models' capability to draw connections among words to generate novel fluent, sensible, and humorous texts. This is a challenging task with many downstream applications, such as context-situated humor generation, a task that involves generating humorous text in a given situation or context. In this case, input keywords can come from conversational context (e.g., chatbot dialogues) or narrative context (e.g., creative short stories).

More formally, we take as input keywords K, the pun word p_w and alternate pun word a_w, [11] and produce novel and fluent puns that incorporate the keywords. [12] Optionally, we also include pun word sense annotations S_{p_w} and S_{a_w} from the original SemEval 2017 Task 7 annotations.

Data Preparation. For this task, we limit our data to samples that contain both (1) annotated human keywords K from ExPUN (AF_6), and (2) pun word sense annotations S_{p_w} and S_{a_w} from SemEval 2017 Task 7. There are 1,482 such samples that have both annotations, from which we reserve 100 as test data and use the rest for model training. To construct input human-annotated keywords for this task, we aggregate keywords for each sample using the method described in Section 2.4. Additionally, we evaluate the effect of finetuning on automatically-extracted keywords instead of human-annotated keywords by automatically extracting keywords for each sample by running the RAKE (Rose et al., 2010) algorithm on the pun text.

Evaluation Metrics. We use both automatic metrics and human evaluation to evaluate the quality of generated puns. For automatic evaluation, we calculate word incorporation rate for both pun words and keywords, which measure the model's ability to incorporate all input keywords. Additionally, we run human evaluation using Amazon Mechanical Turk, in which we asked Turkers to label whether or not a given generated pun was successful. [13]

Models. We use the following models:

<u>*AmbiPun*</u> (Mittal et al., 2022). We use the current state-of-the-art homographic pun generation model, AmbiPun, with no further finetuning. We follow the AmbiPun prompt format: "generate sentence: K, p_w, a_w".

<u>*Finetuned T5*</u> (T5$_{\text{FT}}$). We finetune T5-base on ExPUN using input prompt "generate a pun that situated in K, using the word p_w, p_w means S_{p_w}, a_w means S_{a_w}." The output is the pun itself. [14]

<u>*Finetuned T5 with pretraining*</u> (T5$_{\text{PT+FT}}$). To increase the model's ability to incorporate keywords, we pretrain T5 on non-pun text. For a given pun word, we first extract 200 sentences that contain the pun word from BookCorpus (Zhu et al., 2015), then use RAKE to automatically extract keywords for each sentence. We construct examples where inputs are automatically extracted keywords, and outputs are sentences from BookCorpus including pun words. We pretrain a T5 model with this data before finetuning it on ExPUN.

Results. Table 7 shows results of our pun generation models. While the AmbiPun baseline achieves superior word incorporation performance, our T5$_{\text{PT+FT}}$ model finetuned using ExPUN keywords generates successful puns at a higher rate, showing the value of training on our dataset. Furthermore, while pun word incorporation is im-

[11] $p_w = a_w$ for homographic puns.

[12] We refer to "fluent puns" primarily in the context of the pun generation task, since generating fluent natural language realizations is often non-trivial, particularly in the case of controllable language generation tasks such as ours.

[13] Turkers had to pass a qualifier by correctly labeling $>=$ 80% of 20 samples that we manually annotated. Success is defined as whether the text supports both senses of the pun word. We measure inter-annotator agreement among 3 annotators using Fleiss' kappa ($\kappa = 0.49$), showing moderate agreement.

[14] Further experimental details in Appendix E.

Key-words	Model	Word Incorp. % p_w	K	both	Success Rate %
RAKE	T5$_{\text{FT}}$	90.0	76.4	80.2	35.0
	T5$_{\text{PT+FT}}$	**99.0**	72.9	81.2	54.0
ExPUN	AmbiPun	**99.0**	**92.1**	**94.4**	51.0
	T5$_{\text{FT}}$	58.0	80.3	72.3	40.0
	T5$_{\text{PT+FT}}$	93.0	80.2	83.5	**77.0**

Table 7: Automatic (Word Incorporation Rate) and human evaluation (Success %) of puns generated by models finetuned using automatically-extracted (RAKE) and human-annotated (ExPUN) keywords (with AmbiPun baseline (Mittal et al., 2022)). `PT` stands for Pre-Training and `FT` stands for Fine-Tuning. Both T5$_{\text{PT+FT}}$ models finetuned with RAKE-based keywords or ExPUN-based keywords use RAKE-based keywords during pretraining.

proved by pretraining on outside sources using RAKE keywords, using automatically-extracted keywords when training on in-domain pun text does not translate to more successful puns. Instead, models finetuned with the more carefully-selected, human-annotated ExPUN keywords generate puns relatively more successfully than their RAKE-trained counterparts.

Table 8 shows examples of generated puns from our ExPUN-T5$_{\text{PT+FT}}$ model. The model is able to generate both homographic and heterographic puns somewhat coherently using one of the pun word senses. However, while some puns are successful, Rows 3 and 6 show some ways our model can struggle to generate the respective pun types: it does not always incorporate the alternate word sense in a clever or meaningful way, and can stitch copied input keywords together into incoherent sentences. Our results show pun generation is a very challenging task, and that careful selection of pun keywords and a deeper understanding of humor in wordplay is essential for generating puns successfully.

4 Related Work

In this work, we contribute annotations for a humor dataset as well as two humor-related generation tasks. The work is broadly related to pun generation, pun detection, explanation generation, and humor generation. We briefly summarize works in these directions.

Pun generation. Many of the previous works on pun generation have focused on phonological or syntactic patterns rather than semantic patterns (Miller and Gurevych, 2015; Hong and Ong, 2009; Petrović and Matthews, 2013; Valitutti et al., 2013),

#	p_w, a_w	K	Generated Pun
1	solution/ solution	scientist, problem, liquid chemicals	A liquid chemicals scientist has a problem with a solution.
2	makeup/ makeup	class, beauty school	A beauty school class was cancelled because of a lack of makeup.
3	charges/ charges	farmer, bull	The farmer, the bull, had to pay the charges.
4	fission/ fishing	nuclear physicist, vacation, trip	The nuclear physicist took a trip to the Bahamas for his fission vacation.
5	fare/ fair	carnival, county	The carnival in the county was a fare event.
6	vault/ fault	bankers, generous	OLD BACHERS never die they just become very generous. They have a vault fault.

Table 8: Examples of input pun words and keywords and the resulting generated puns. We show examples of both homographic and heterographic generated puns.

thus lacking flexibility. He et al. (2019) make use of local-global surprisal principle to generate homophonic puns and Yu et al. (2020) uses constrained lexical rewriting for the same task. Hashimoto et al. (2018) use a retrieve and edit approach to generate homographic puns and Yu et al. (2018); Luo et al. (2019) propose complex neural model architectures such as constrained language model and GAN. Mittal et al. (2022) generate homographic puns given a polyseme and try to incorporate the multiple senses of the polyseme. Tian et al. (2022) proposed a unified framework to generate both homographic and homophonic puns leveraging humor principles. Our keyword-conditioned pun generation task encourages models to focus more on the linguistic structures via pun keywords as we observe that human-extracted keywords usually reflect the ambiguity and distinctiveness principles as discussed in Kao et al. (2016). The keyword-conditioned pun generation setup can also facilitate more engaging pun generation scenarios such as context-situated pun generation (Sun et al., 2022).

Humor generation. With the recent advent of diverse datasets (Hasan et al., 2019; Mittal et al., 2021; Yang et al., 2021), it has become easier to detect and generate humor. While large pre-trained models have become fairly successful at detection, humor generation still remains an unsolved problem. Therefore, humor generation is usually studied in a specific settings. Petrović and Matthews

(2013) generates jokes of the type 'I like my X like I like my Y, Z'. Garimella et al. (2020) develops a model to fill blanks in a Mad Libs format to generate humorous sentences and Yang et al. (2020) edit headlines to make them funny. More research is required to generate humorous sentences that are not constrained by their semantic structure.

Natural language explanation generation. Collecting and utilizing natural language explanations to help various NLP tasks is an emerging topic. The earliest work by Ling et al. (2017) collected natural language justifications, called rationales, to help solve math problems. However, their setup is limited to solving math problems given how their rationales and models were structured. Jansen et al. (2018) composed a dataset of explanation graphs for elementary science questions to support multi-hop inference. Like Ling et al. (2017), they emphasized the explanations structures. Several works have introduced large-scale datasets of natural language explanations for the natural language inference (NLI) (Camburu et al., 2018; Kumar and Talukdar, 2020), commonsense reasoning (Rajani et al., 2019), and hate speech detection (Mathew et al., 2021) tasks. However, there are no existing datasets or models that focus on explaining humor, which is a challenging task that involves commonsense and world knowledge.

Pun detection. Being able to detect puns can be an essential step to generating them. SemEval 2017 Task 7 (Miller et al., 2017) introduced the challenge of pun detection, location detection and sense interpretation for homographic and heterographic puns. They also released a dataset which has become the backbone of our and several other related works. Diao et al. (2019) make use of gated attention networks to detection heterographic puns. Zou and Lu (2019) introduce a tagging scheme to jointly detect and locate puns, and apply this approach to both heterographic and homographic puns. Zhou et al. (2020b) jointly model contextual and phonological features into a self-attentive embedding in their approach for pun detection and location tasks.

5 Conclusion

In this paper, we contribute a dataset of extensive, high-quality annotations of humor explanation, keywords, and fine-grained funniness ratings. This is the first humor dataset with such extensive and fine-grained annotations. Based on the annotations, we propose two tasks: pun explanation and keyword-conditioned pun generation, to challenge state-of-the-art natural language understanding and generation models' ability to understand and generate humorous text. We benchmark several strong models' performances on the two proposed tasks to validate the practical usage of the proposed annotations, and show that our human-annotated explanations and keywords are beneficial in understanding and generating humor. Future directions include a deeper analysis of how to characterize pun explanation more objectively within our annotation scheme, as well as further exploration of better models for both the pun explanation and pun generation tasks.

Acknowledgements

The authors would like to thank Scott Benson and the rest of the Alexa Data Services Rapid Machine Learning Prototyping (RAMP) team for all of their help with preparing and performing the annotation task. We also thank anonymous reviewers for their constructive feedback and suggestions that helped improve the paper.

Limitations

This work focuses on understanding and generation of puns, a single and very specific form of humorous language. We hope that our annotation schema and methods can be used in the future to extend to other forms of humor, e.g., joke generation. Additionally, we acknowledge that humor is a highly subjective area, i.e., what might be perceived as humorous may differ greatly from one person to another depending on their unique backgrounds and experiences. We hope this work can be used as an initial framework to begin characterizing humor through human-written explanations, such that it can be used more broadly to give insight into what contributes to humorous content for different individuals and groups.

Finally, since we use pretrained language models for our generation tasks, we note that this makes our models susceptible to generating biased or sensitive content. While we do not explicitly address concerns around bias/sensitive content within our framework to date, we aim to incorporate these considerations into pun generation as we develop new models, including methods to filter our inputs and generated data for toxicity and biased references that may be deemed offensive.

Ethics

We hereby acknowledge that all of the co-authors of this work are aware of the provided *ACL Code of Ethics* and honor the code of conduct.

The text in the dataset (puns and non-pun text) is from the SemEval 2017 Task 7 dataset (Miller et al., 2017) including jokes, aphorisms, and other short texts sourced from professional humorists and online collections. No user data from commercial voice assistant systems is used. We collect the human annotation of pun keywords, explanations, and other meta-fields via full-time employees (with all employee-entitled fringe benefits) who are hired by the co-authors' organization for the purposes of data annotation and are not co-authors of the paper. We ensure that all the personal information of the workers involved (e.g., usernames, emails, urls, demographic information, etc.) is discarded in our dataset. Overall, we ensure our pay per task is above the the annotator's local minimum wage (approximately $15 USD / Hour).

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Karyn Buxman. 2008. Humor in the OR: A stitch in time? *AORN journal*, 88(1):67–77.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-SNLI: Natural language inference with natural language explanations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9539–9549. Curran Associates, Inc.

Santiago Castro, Luis Chiruzzo, Aiala Rosá, Diego Garat, and Guillermo Moncecchi. 2018. A crowd-annotated Spanish corpus for humor analysis. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 7–11, Melbourne, Australia. Association for Computational Linguistics.

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

Miruna-Adriana Clinciu, Arash Eshghi, and Helen Hastie. 2021. A study of automatic metrics for the evaluation of natural language explanations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2376–2387, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yufeng Diao, Hongfei Lin, Liang Yang, Xiaochao Fan, Di Wu, Dongyu Zhang, and Kan Xu. 2019. Heterographic pun recognition via pronunciation and spelling understanding gated attention network. In *The World Wide Web Conference*, page 363–371.

Aparna Garimella, Carmen Banea, Nabil Hossain, and Rada Mihalcea. 2020. "judge me by my size (noun), do you?" YodaLib: A demographic-aware humor generation framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2814–2825, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Md Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency, and Mohammed (Ehsan) Hoque. 2019. UR-FUNNY: A multimodal language dataset for understanding humor. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2046–2056, Hong Kong, China. Association for Computational Linguistics.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S. Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *NeurIPS*, pages 10073–10083.

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

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 24–31, Boulder, Colorado. Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. 2018. WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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

Maxime Kayser, Oana-Maria Camburu, Leonard Salewski, Cornelius Emde, Virginie Do, Zeynep Akata, and Thomas Lukasiewicz. 2021. E-ViL: A dataset and benchmark for natural language explanations in vision-language tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1244–1254.

Sawan Kumar and Partha Talukdar. 2020. NILE: Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742, Online. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. 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. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Fuli Luo, Shunyao Li, Pengcheng Yang, Lei Li, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Pun-GAN: Generative adversarial network for pun generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3388–3393, Hong Kong, China. Association for Computational Linguistics.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. HateXplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14867–14875.

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.

Anirudh Mittal, Pranav Jeevan P, Prerak Gandhi, Diptesh Kanojia, and Pushpak Bhattacharyya. 2021. “so you think you’re funny?”: Rating the humour quotient in standup comedy. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10073–10079, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria. Association for Computational Linguistics.

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.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1(1-20):10–1002.

Jiao Sun, Anjali Narayan-Chen, Shereen Oraby, Shuyang Gao, Tagyoung Chung, Jing Huang, Yang Liu, and Nanyun Peng. 2022. Context-situated pun generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yufei Tian, Divyanshu Arun Sheth, and Nanyun Peng. 2022. A unified framework for pun generation with humor principles. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alessandro Valitutti, Hannu Toivonen, Antoine Doucet, and Jukka Toivanen. 2013. "let everything turn well in your wife": Generation of adult humor using lexical constraints. volume 2.

Orion Weller and Kevin Seppi. 2019. Humor detection: A transformer gets the last laugh. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3621–3625, Hong Kong, China. Association for Computational Linguistics.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ziqing Yang, Yiming Cui, Zhipeng Chen, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. 2020. TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 9–16, Online. Association for Computational Linguistics.

Zixiaofan Yang, Shayan Hooshmand, and Julia Hirschberg. 2021. CHoRaL: Collecting humor reaction labels from millions of social media users. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4429–4435, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Zhiwei Yu, Hongyu Zang, and Xiaojun Wan. 2020. Homophonic pun generation with lexically constrained rewriting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2870–2876, Online. Association for Computational Linguistics.

Wangchunshu Zhou, Jinyi Hu, Hanlin Zhang, Xiaodan Liang, Maosong Sun, Chenyan Xiong, and Jian Tang. 2020a. Towards interpretable natural language understanding with explanations as latent variables. In *Advances in Neural Information Processing Systems*, volume 33, pages 6803–6814. Curran Associates, Inc.

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

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

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

A ExPUN Dataset Annotation Details

A.1 Annotation Guidelines

Below, we include the annotation guidelines we used to collect the ExPUN dataset. All pun texts in the provided examples are from the original SemEval 2017 Task 7 dataset (Miller et al., 2017).[15]

Guidelines You will be provided a CSV file of short texts, one short text to be annotated per row. Each row contains the text content as well as columns for each of the requested annotations. For each row, read the text carefully, and provide the following annotations:

1. Mark whether you understood the text with 0/1 (0 didn't understand, 1 understood the text).
   - If you don't understand the meaning of the text (regardless of whether or not it should be perceived as funny), rate the sample as 0 (didn't understand).
   - For this assessment, you can use a quick Google search to look up any vocabulary/terms you don't immediately understand. However, if the amount of research it would take to understand the text goes beyond a quick (<1 min) search, rate the sample as 0 (didn't understand).
   - *Example text that was marked "don't understand" (0):* A doctor's mistakes go six feet under; a dentist's cover an acre.
   - If you rate this sample as 0 (didn't understand), skip the rest of the annotation for this sample.
2. Mark whether you find the text offensive or inappropriate with 0/1 (0 not offensive, 1 offensive), meaning the text is racist or is biased against marginalized groups, or is generally offensive. If you rate this sample as 1 (is offensive), you may optionally skip the rest of the annotation for this sample.
3. Mark whether you think the text is intended to be a joke with 0/1 (0 not a joke, 1 is a joke).
   - Text should be labeled as 1 (is a joke) even if it intends to be humorous, but falls flat or is a lame/bad joke.
   - *Example text labeled 0 (not a joke):* All that glistens is not gold.
   - *Example text labeled 1 (is a joke):* These are my parents, said Einstein relatively.
     Why is this a joke? Though subtle, the text is a pun on the word "relatively" that associates Einstein with his relatives (parents) and his theory of relativity.
   - If you rate this sample as 0 (not a joke), skip the rest of the annotation for this sample.
4. Rate funniness on a Likert scale of 1-5 (1 very not funny, 5 very funny).
   - *Score of 1:* A very not funny joke consists of a joke that is not funny at all, or tries to be funny but does not achieve the intended effect.
     Example of Funniness 1 (not funny): These are my parents, said Einstein relatively.
   - *Score of 3:* An average joke consists of a joke that that is average and may elicit some chuckles (or groans) from you or others.
     Example of Funniness 3 (average funniness): When they told him that his drum couldn't be fixed, it didn't resonate very well.
   - *Score of 5:* A very funny joke consists of a good joke that you find humorous and potentially would want to share/tell to others.
     Example of Funniness 5 (very funny): Yesterday I accidentally swallowed some food coloring. The doctor says I'm OK, but I feel like I've dyed a little inside.
5. Explain in concise natural language about why this joke is funny. If external or commonsense knowledge is required to understand the joke and/or its humor, please include the relevant knowledge in your explanation. If the joke is a pun or play on words, you must provide an explanation of how the play on words works.
   - *Example joke:* What do you use to cut a Roman Emperor's hair? Caesars.
     Bad explanation: The joke is a play on words about Caesar and scissors.
     Good explanation: The joke is a play

[15] https://alt.qcri.org/semeval2017/task7/. The data is released under CC BY-NC 4.0 license (https://creativecommons.org/licenses/by-nc/4.0/legalcode).

on words: Caesar was a Roman Emperor, and "Caesars" sounds like "scissors", which is something you use to cut hair.

- *Example joke:* There was a kidnapping at school yesterday. Don't worry, though – he woke up!
  Bad explanation: The joke is a play on words about kidnapping → kid napping.
  Good explanation: The joke is a play on words. The word "kidnapping" implies that a kid was taken hostage at school, but "he woke up" suggests that it was actually just a kid taking a nap instead.

6. Pick out (as few as possible) keyword phrases from the joke that are related to the punchline/the reason of the joke being funny (written as a pipe-separated (|) list of phrases with spaces).
   - Phrases can be multiple words long.
   - The keyword phrases should be copied verbatim from the joke (no need to reword them).
   - Keep keyword phrases sparse and mainly limited to content words. The keyword phrases should not span the entire joke. As a general guideline, the words in keyword phrases should make up <50% of the words in the full joke (though this may be difficult to achieve for shorter jokes).
   - *Example joke:* I used to hate maths but then I realised decimals have a point.
     Bad keywords (too dense): I used to hate maths but | decimals have a point
     Good keywords: maths | decimals | point

We note that this is a highly subjective task, since different people perceive humor differently! We encourage you to do your best to determine how to annotate each item as consistently as possible.

Example annotations (funniness ratings are subjective, and may differ from yours!):

- *Text:* Yesterday I accidentally swallowed some food coloring. The doctor says I'm OK, but I feel like I've dyed a little inside.
  Understand: 1
  Offensive: 0
  Is a Joke: 1
  Funniness: 5
  Explanation: The joke is a pun. The main character feels they've "died a little inside" meaning they've been changed for the worse by swallowing food coloring. At the same time, food coloring contains dye, so the main character has been "dyed" on the inside by swallowing some.
  Keywords: swallowed | food coloring | dyed a little inside

- *Text:* Waiter, there's a fly in my soup! "I know. It gives you a nice buzz doesn't it?"
  Understand: 1
  Offensive: 0
  Is a Joke: 1
  Funniness: 2
  Explanation: This is both a pun and a reference to a common joke format. "Waiter, there's a fly in my soup!" is an old joke setup with varying punchlines. Flies make a noise commonly described as a "buzz". "Buzz" can be used as a noun referring to a pleasant heightened sensation, commonly from drinking alcohol.
  Keywords: fly | soup | buzz

- *Text:* The evil onion had many lairs.
  Understand: 1
  Offensive: 0
  Is a Joke: 1
  Funniness: 3
  Explanation: This is a pun. An evil lair is a hideout for a villain in a comic book or show. Onions are layered vegetables. The joke is that the onion had many lairs because it was evil.
  Keywords: evil onion | many lairs

- *Text:* Hope for the best, but prepare for the worst.
  Understand: 1
  Offensive: 0
  Is a Joke: 0
  (No need to fill in any more information in subsequent columns, as this text is not a joke.)

Additional calibrating examples The following examples were rated with an average Funniness rating >= 2 in previous pilot rounds and can be used to calibrate your rubric for assigning Funniness scores.

- *Text:* Drinking too much of a certain potent potable may require a leave of absinthe.
  Funniness ratings: [3, 4, 2, 1, 2]
  Average rating: 2.4

- *Text:* Animals that tunnel in the soil have to have an escape root.
  Funniness ratings: [3, 3, 1, 2, 2]
  Average rating: 2.2

- *Text:* My friend's bakery burned down last night. Now his business is toast.
  Funniness ratings: [4, 3, 2, 1, 2]
  Average rating: 2.4

- *Text:* What is the best store to be in during an earthquake? A stationery store.
  Funniness ratings: [1, 3, 2, 2, 3]
  Average rating: 2.2

A.2 Feedback from Pilot Rounds

We did a few pilot rounds to help annotators calibrate on funniness, since not only is funniness highly subjective, but also since many puns aren't "traditionally funny", but instead more humorous due to being "clever" or "creative". Feedback we received from annotators was mostly around including more detailed definitions and examples for highly-subjective criteria such as "funniness" and "offensiveness". We added questions on whether annotators "understood the text" to help distinguish between puns that were not understood vs. puns that were understood but then marked as "not funny", and added clarifying examples of "joke keywords" to discourage excessive copying of the input text in the annotations.

A.3 Inter-Annotator Agreement for Offensiveness (AF_2)

We note relative low inter-annotator agreement for AF_2, as identifying offensive/inappropriate content is a highly subjective and complex task, as it generally covers social stereotypes, biases, aggressive expressions, micro-aggressions, etc. Kappa looks at agreement of raw scores, while Spearman computes correlation of ranks. Combining these differences with the subjectivity of the task could explain the disparity between Kappa and Spearman scores for AF_2.

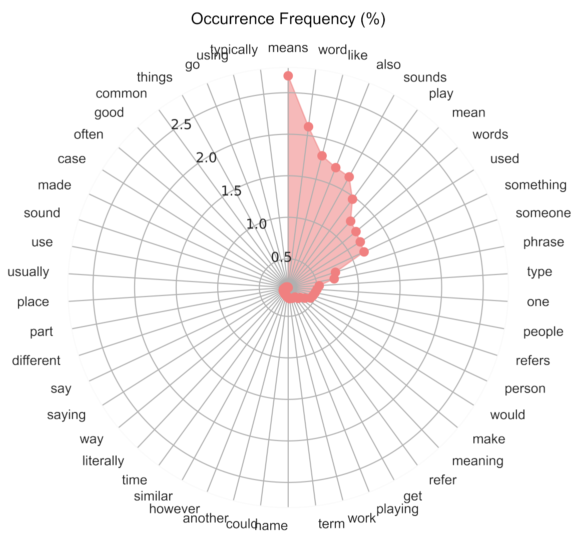


Figure 3: Top 50 most frequent words in explanations.

B Analysis of Annotated Explanations

B.1 Frequent Explanation Keywords

Figure 3 shows the distribution of the top 50 most frequent words in our annotated explanations (after removing stop words and punctuation, as well as the task-specific words "pun" and "joke").

B.2 Explanation Sentence Templates

To further explore what kinds of explanations annotators have provided within ExPUN, we use a simple templatization scheme to uncover common patterns. Given an input pun and explanation pair from the dataset, we templatize an explanation by replacing any content words (non-stop words) from the pun that show up in the explanation. For example, for the pun "I wrote a novel about a fellow who had a small garden. It didn't have much of a plot.", the explanation sentence "This is a play on the word plot." would become the template "this is a play on the word [X]." We then count the number of unique templates across the dataset.

Table 9 shows counts frequency counts for different selected templates found in ExPUN. The top half of the table shows instances of highly-frequent explanations, such as "[X] sounds like [X]" indicating a heterographic pun. The bottom half of the table shows examples of unique templates that show up only once but exemplify rich explanations that include context-specific words that are useful for pun understanding. We note that the more frequent templates help to characterize common ways to explain the humor within puns, while the unique templates serve as highly-informative descriptions

Freq	Explanation Sentence Template
466	[X] sounds like [X]
116	this is a pun on the word “[X]”
16	[X] is a type of [X]
11	this joke is playing on the word “[X]” and its different meanings
8	[X] is another word for [X]
3	[X] has two meanings
Pun	**A gambling gardener usually hedges his bets.**
1	to “hedge your [X]” means to take a chance on something and a hedge is a type of shape that a [X] can cut plants into
Pun	**I must attend my flock, said Tom, sheepishly.**
1	this is an attempt at a joke since a group of sheep are known as a [X] and ‘sheepishly’ is another term for ‘embarrassed’
Pun	**People who make necklaces may get** beady eyes.
1	“[X]” here refers to the beads in [X] and the way [X] have to hyper-focus their [X] when crafting hand-made jewelry

Table 9: Sample explanation sentence templates collected in ExPUN, along with their frequencies.

that can aid in the pun classification task (e.g., a detailed, contextualized definition of a word/phrase).

C Keyword Aggregation Algorithm

We propose the keyword aggregation algorithm in Algorithm 1 to merge keywords annotation among different workers.

D Classifier Implementation Details

We finetune pretrained language models for classifying whether given text samples are jokes, and we use HuggingFace (Wolf et al., 2020) throughout our implementation for accessing model checkpoints and modeling. For hyper-parameter search, we tried the combinations of learning rate $\{1e^{-4}, 3e^{-4}, 1e^{-5}, 3e^{-5}\}$ * training epoch {3, 10, 20}. The final hyperparameters for bert-base, roberta-base and deberta-base are: learning rate $1e^{-5}$, training epoch 20 and training batch size 32. For roberta-large-mnli and bart-large-mnli models, we reduce the training epochs to 3 and training batch size to 8. We choose the checkpoint with the best accuracy on the dev set for inference.

For ELV model, we use the released code and inherited most of their default hyperparameters for *ELV-sa*.[16] We change the training batch size per GPU to 4 to accelerate the training.

[16] https://github.com/JamesHujy/ELV/blob/main/EST_sa/train_restaurant.sh

Algorithm 1 Keyword Aggregation Algorithm

Input: For each instance X_i, $i \in \{1, ..., N\}$, annotations from every worker w_j, $j \in \{1, ..., 5\}$ denoted as X_{ij}.
Output: keywords for X_i

```
1:  for $j \in \{1, ..., 5\}$ do
2:      // calculate the reliability score
        $S_j = \frac{1}{N}\sum_{i=0}^{N}$ (#keywords−#average tokens in each keyword)
3:  end for
4:  sort all workers with $S$ and get preferred worker list $L$
5:  // set worker with the highest $S$ as anchor worker $w_a$
6:  aggregated_keywords = []
7:  for $K_z \in X_{ia}$ do
8:      filtered_keywords $K_{\text{filter}}$ = []
9:      for $j \in \{1, ..., 5\}$ do
10:         for $K_p \in X_{ij}$ do
11:             calculate $F(K_z, K_p)$
12:         end for
13:         choose the keyword $K_P$ in $X_{ij}$ with highest $F$
14:         if $F(K_z, K_P) > 60$ then
15:             append keyword $K_P$ to $K_{\text{filter}}$
16:         end if
17:     end for
18:     $AVG_a = \frac{1}{\text{len}(F_{filter})}\sum F(K_z, K) K \in K_{\text{filter}}$
19:     set the worker with the second highest $S$ as new anchor
        worker $w_b$. Repeat $L_6$-$L_{18}$ and get $AVG_b$
20:     if $AVG_a \geq AVG_b$ then
21:         append $X_{ia}$ to aggregated_keywords
22:     else
23:         append $X_{ib}$ to aggregated_keywords
24:     end if
25:     /* if only one worker has keyword annotation, append
        this worker's annotation to aggregated_keywords */
26:     Remove duplication from aggregated_keywords
27: end for
```

E T5 Implementation Details

We finetune multiple T5 models (Raffel et al., 2020) in our work, and we use T5-base from SimpleT5 [17] throughout our implementation. We use 512 and 256 for the maximum source length and the maximum target length respectively. As the optimizer, we use AdamW (Loshchilov and Hutter, 2019) with a learning rate of 0.0001. For the pretraining stage, we finetune T5 for 3 epochs on retrieved BookCorpus data. During the finetuning stage, we train each model on a Tesla V100 with a batch size of 8 for 30 epochs. During inference, we use beam search as the decoding method with a beam size of 2. We terminate decoding when the EOS token is generated or the maximum target length is reached.

[17] https://github.com/Shivanandroy/simpleT5