A Unified Framework for Pun Generation with Humor Principles

Yufei Tian[1], Divyanshu Sheth[2]*, Nanyun Peng[1]
[1]Computer Science Department, University of California, Los Angeles,
[2]Department of Industrial and Systems Engineering, Indian Institute of Technology, Kharagpur
{yufeit, violetpeng}@cs.ucla.edu, divyanshusheth1@gmail.com

Abstract

We propose a unified framework to generate both homophonic and homographic puns to resolve the split-up in existing works. Our framework takes a theoretically motivated approach to incorporate three linguistic attributes of puns to language models: ambiguity, distinctiveness, and surprise. Our framework consists of three parts: 1) a context words/phrases selector to promote the aforementioned humor attributes, 2) a generation model trained on non-pun sentences to incorporate the context words/phrases into the generation output, and 3) a label predictor that learns the *structure* of puns which is used to steer the generation model at inference time. Evaluation results on both homophonic and homographic puns demonstrate the superiority of our model over strong baselines.[1]

1 Introduction

Recently, computational humor theories investigating why puns are funny have shown high correlations with human judgments. Kao et al. (2016) use a probabilistic model to decompose puns into two dimensions: **ambiguity** of meaning and **distinctiveness** of viewpoints, and show that these two aspects combined have the strongest alignment with human judgments (p<5‰). He et al. (2019) show that ambiguity/distincitveness alone cannot capture the whole picture, and develop an additional metric to measure how much **surprise** is aroused when the pun word and alternative word are flipped. For example in Figure 1, the pun word is *soled* and the alternative word is *sold*. Seeing *soled* in the phrase 'were *soled* at the store at half price' instead of *sold* arouses surprise in the local context but makes sense in the global context.

Despite the success in identifying important linguistic traits of successful pun sentences, how to incorporate these aforementioned theories into the pun generation process is still an open problem. Although He et al. (2019) propose a retrieve-and-edit approach to incorporate surprise, their error analysis shows that the proposed retrieval methods are often unsuccessful.

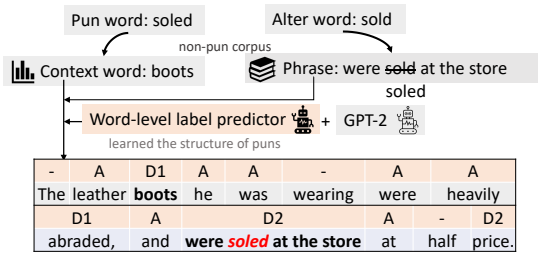

Figure 1: An illustration of our approach. The pun word pair (e.g. *'soled-sold'*) is the input. After retrieving a suitable context word and a phrase, we use a pun label predictor to steer the base GPT-2 model to produce puns. Labels D1/D2/A mean the next word should be distinct to (supporting) the pun word, distinct to (supporting) the alternative word, or ambiguous. A '-' mark means the label predictor is less confident and thus we do not intervene the generation process.

Moreover, existing works on pun generation are split up in terms of generating **homographic puns**, wherein the same written word has two or more meanings (Mittal et al., 2022; Yu et al., 2020, 2018), and **homophonic puns**, where two words that sound similar have different meanings (Luo et al., 2019; He et al., 2019; Hashimoto et al., 2018). There lacks a unified generation framework for both types of puns.

In this work, we incorporate all three principles: ambiguity, distinctiveness, and surprise into pun generation, and bridge the gap between the two pun types. We hypothesize that *there is a **learnable structure** for puns regardless of the pun type*, and propose a unified framework by converting homographic puns to homophonic ones. Specifically, we carefully extract from a non-pun corpus 1) a context word that supports the meaning of the pun word, and 2) a phrase that is both characteristic to the alternative word and compatible with the pun word. Next, we train a discriminator on existing homophonic pun data to learn the structure of a

*Work done when the author was interning at UCLA.
[1]Our code is available at https://github.com/PlusLabNLP/Unified_PunGen

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3253–3261
December 7-11, 2022

pun – the *type* of each word in the sentence, which could be one of 'A' – ambiguous, 'D1' – distinct to the pun word, or 'D2' – distinct to the alternative word. One challenge, however, is that there are no ground truth labels. To this end, we collect a small amount of human annotations and boost from weak, unsupervised models to stronger, supervised models. At inference time, a label predictor is used to guide a base GPT-2 model to generate puns. At each generation step, we re-score the tokens generated by the base language model according to the predicted type, except for the case when the label predictor's confidence is under a set threshold. Our model outperforms existing baselines for both pun types.

2 Related Works

Linguistic traits of puns. Kao et al. (2016) decompose puns into two dimensions — *ambiguity* of meaning and *distinctiveness* of viewpoints, and show that that ambiguity is useful in distinguishing non-puns from puns, while distinctiveness is useful when spotting good and funny puns from bad or boring non-puns. To the best of our knowledge, we are the first to formally incorporate the famous ambiguity-distinctiveness principle to guide pun generation. In addition, He et al. (2019) propose the *local-global surprisal principle* to measure the humorous effect aroused when a word appears unexpectedly in the local context but makes sense given the global context, based on which we improve the way surprise is introduced in generation.

Pun generation. Existing works on pun generation often rely on naive intuitions of semantic ambivalence. For example, Yu et al. (2018) and Luo et al. (2019) promote the ambivalence of the pun word via a constrained language model and reinforcement learning; others find related words to support semantic ambiguity (Yu et al., 2020; Mittal et al., 2022). However, these systems lack serious theoretical backbones and therefore none could evaluate their generated results with regards to the proposed intuitions. What's more, the nature of 'ambivalence' alone leads to generic/boring word choice and short outputs. By incorporating distinctiveness and surprise, we ensure that the generated puns are informative and interesting.

One reason that previous works leverage those simple intuitions to generate puns (He et al., 2019; Yu and Wan, 2019; Yu et al., 2020) is that the small corpora size (Miller et al., 2017; Sun et al., 2022a) makes it impractical to train generation models end-to-end using human written puns. We hence propose to learn the *structure* of puns instead of the *actual texts*, which requires far less data to train on. Finally, all previous works (except a concurrent one (Sun et al., 2022b)) can only generate either homographic puns or homophonic puns. Leveraging the shared structure of puns regardless of the pun type, our model can generate both pun types.

3 Methodology

The input to our system is a pun word-alternative word pair (pw-aw, e.g., *soled-sold*), and the target output is a high-quality pun sentence that contains pw, e.g., *'The leather boots he was wearing were heavily abraded, and were soled at the store at half price.'* In this section, we first describe the three components to generate homophonic puns as shown in Figure 1: a context word and phrase selector, a label predictor and the procedure of curating training signals, and the generation module in Section 3.1 to 3.3. Then, we migrate the whole system to homographic puns in Section 3.4.

3.1 Obtaining Context Words and Phrases

We retrieve and select two things: a context word that supports the meaning of the pun word, and a phrase that is both characteristic to the alternative word and compatible with the pun word.

Inspired by He et al. (2019), given a pun-alternative word pair ($pw - aw$), we look for an ideal phrase that contains aw and replace it with pw to arouse surprise. To this end, we first extract multiple (N_1=20) phrases that contain aw from a large non-pun corpus consisting of 20,000 sentences from Wikipedia and Gutenberg BookCorpus (Lebert, 2009), and rank the phrases by how well they exhibit the semantics of the pun pair. Specifically, we first replace aw with a '<mask>' token, and run RoBERTa-Large (Liu et al., 2019) to obtain the probability of aw in the masked position. We remove the less probable half, filtering out those that are less characteristic of aw, as shown in Table 1. Next, we conduct a similar mask infilling procedure for pw, and select the middle-ranking phrase to avoid it being either too general (e.g., 'a new taxi was created') or too incompatible (e.g., 'an export taxi on agricultural products'). These two rankings ensure the final selected phrase arouses surprisal when people see pw instead of aw, but also still find it reasonable.

For obtaining the context words, our idea is similar to that proposed by (Mittal et al., 2022). We retrieve sentences from the same non-pun corpus

Retrieved Phrase	Tax	Taxi
get in all that <mask> trouble	✓	✓
an export <mask> on agricultural products	✓	×
a new <mask> was created	✓	×
made <mask> deductible against income	✓	✓

Table 1: An example of the retrieved phrases which are characteristic of the alternative word, 'tax'. The pun-pair is 'tax-taxi'. A '✓' means the phrase is compatible with the corresponding word according to our mask in-filling model.

Category	A	D1	D2
$\mathbf{Bert_n}$	0.81	0.68	0.60
- human labeled train data	- 0.02	- 0.06	- 0.05
+ high confidence (T=0.9)	+0.03	+0.02	+0.05

Table 2: The F1 scores of $\mathbf{Bert_n}$ and its ablations on human annotated testset.

containing the target word (pw), and then extract keywords from the sentences using RAKE (Rose et al., 2010). Based on the TF-IDF values of those keywords, we take the top N_2 (N_2=20) words that uniquely co-occur with the target word, and then randomly sample one to encourage creativity.

3.2 Label Predictor

Our label predictor aims at learning and predicting the word-level structures of 1,500 human-written pun sentences collected by Miller et al. (2017). Each word in a pun sentence falls into one of three types: 'A' for ambiguous, 'D1' for distinct to the pun word, 'D2' for distinct to the alternative word. We finetune a BERT (Devlin et al., 2018) sequence classification model to *predict* the *next* token type. In this section, we will first talk about a data-efficient label collection procedure that boosts from a weak, unsupervised method to a stronger, weakly supervised method .

Ground Truth Label Curation Before we could train the model, how can we automatically categorize each word in a pun sentence? We start with an unsupervised approach: word semantic similarity. Specifically, we compute the cosine similarity between the glove embeddings of pw and aw and each word in the sentence (tw), and label the word as D1/D2 if the difference is larger than a threshold T (i.e., $|cos(tw, pw) - cos(tw, aw)| > T$). Otherwise, we label the word as A. We also compute their correlation with human judgements. In total, we collected human annotations on 500 data points. Since the label predictor should predict the type/category of the *next* word without knowing the future, we mimic this setting and show human annotators 1) an incomplete pun sentence, i.e. containing the part of the sentence *before* the current word tw being evaluated, 2) tw, and 3) pw and aw, and ask them to decide whether tw is distinct to pw, distinct to aw, or ambiguous. With grid search, we find out that with optimal T set to 0.15, the aforementioned purely unsupervised method gets 72.9% labeling accuracy.[2]

To further improve the reliability of the 'ground truth' labels, we finetune a BERT-base model as a sequence classifier to classify each word into the three categories. The intuition is to provide this BERT classifier with *less noisy data* so that it can learn the task better than the unsupervised approach. The training data of this BERT classifier includes 8,000 automatic labels obtained using the word unsupervised method. A word is considered distinct (D1/D2) if the difference is > 1.5T, and ambiguous if the difference is < T. In order to compose a dataset with cleaner labels, we simply disregard those training samples whose semantic difference is [T, 1.5T]. We include the incomplete pun sentence, the current word, and the pun pairs as input. Using this, we are able to improve the 'gold' label accuracy to 84.6% on a human annotated held-out dataset. We call this model $\mathbf{Bert_c}$.

Training The label predictor used in our framework predicts the type of the *next* word that is yet to be generated. We call this model $\mathbf{Bert_n}$.[3] The training data of $\mathbf{Bert_n}$ comes from two parts: an additional 430 human labeled data that the unsupervised method and the $\mathbf{Bert_c}$ disagree on, and 8,000 automatic labels where both models agree. A breakdown of its performance by category is reported in Table 2. In addition, we could further improve the predictor's F1 score by considering its confidence. We gain an average of 14.9% increase by discarding only 9.8% cases that the label predictor is less confident on. We argue that there can be multiple choices for the next word, and hence the best performance of this task is bounded. Considering that the task of predicting the type of the next word is much harder because there can be multiple choices as next words, we take into account the confidence level when we use it in the next step.

[2]We have also tried using BERT embedding for semantic similarity. Yet the correlation with human judgment is much lower than the glove embeddings.

[3]Note that there are two models: a classifier $\mathbf{Bert_c}$ and a predictor $\mathbf{Bert_n}$. The former knows the current token while the later does not. We need $\mathbf{Bert_n}$ in inference, and $\mathbf{Bert_c}$ is introduced to help curate $\mathbf{Bert_n}$'s training signals.

Algorithm 1 Discriminative Generation (Single Step)

function DISCRIMINATIVEGEN
Parameters: pun word (pw), alter word (aw), predicted next word label L, confidence c and its threshold T
$cands$ = gpt2.gen_next_word(num = N)
  if $c > T$ **then**
    if L==A **then** sort($cands$, pw+aw)
    if L==D1 **then** sort($cands$, pw)
    if L==D2 **then** sort($cands$, aw)
      ▷ Sort according to semantic relevance in Section 3.2
return $cands$

3.3 Generation Module

Data Preparation and Fine-tuning We fine-tune the GPT-2 model on a combination of Gutenberg BookCorpus and jokes (Annamoradnejad and Zoghi, 2020) to learn the *task format*: given a keyword and a phrase as input, generate a sentence containing them both. For each sentence in the corpus, we use RAKE to extract two salient words. We then include the surrounding context around one of the salient words as the phrase. We make sure that positions of the two extracted keywords are far away enough that the phrase does not contain the other extracted word.

Inference At inference time, we feed the phrase and context word obtained in Section 3.1 as input, and steer the finetuned language model using the label predictor. At each step, we get the predicted type of next token, L, with the corresponding confidence c. If c is larger than a threshold T, we score and rerank the candidate next words by the corresponding label, which can be found in Algorithm 1. Otherwise, if our label predictor is less confident, we do not intervene in the language model's generation. We also enforce the appearance of the complete phrase during decoding when the first two words in the phrase have been generated.

3.4 Migrate to Homographic Puns

We convert the task of generating homographic puns to homophonic puns by leveraging a word-sense disambiguation (WSD) model (Bevilacqua and Navigli, 2020). For example, if the target pun word is "sentence" and the two sense definitions are *"a set of words..."* and *"the punishment..."*, we run the WSD model to identify which extracted phrases exhibit the second sense. Next, we obtain two new words using a reverse dictionary (Qi et al., 2020): *'clause'* for the first sense and *'conviction'* for the second. Then the task can be viewed as that for homophonic puns, where the substitute pw is 'clause' and the substitute aw is 'conviction'.

Homophonic Pun	A	D1	D2	Avg D	S
LCR	**24.93**	4.57	0.90	2.73	0.15
SurGen	23.18	1.65	3.94	2.80	0.54
Base GPT-2	14.85	0.77	0.72	0.74	-0.54
+ label predictor*	18.90	3.83	2.69	3.26	0.53
+ select*	17.56	3.72	4.22	3.97	0.68
+ both*	20.27	6.71	5.46	6.09	**0.77**
Human	22.50	**11.72**	**6.17**	**8.95**	0.71
Homographic Pun	A	D1	D2	Avg D	S
Pun-GAN	20.12	1.22	1.08	1.15	0.22
AmbiPun	17.14	2.39	2.01	2.20	0.34
Base GPT-2	18.45	0.84	0.82	0.83	-0.21
+ label predictor*	**20.64**	4.54	3.18	3.86	0.45
+ select*	19.80	2.89	3.20	3.05	0.59
+ both*	18.27	5.87	6.62	6.25	0.80
Human	18.00	**7.01**	**8.76**	**7.88**	**0.83**

Table 3: Results of automatic evaluation on ambiguity (A), distinctiveness to pun word (D1) and alternative word (D2), and surprisal ratio (S). The * indicates ablations of our method where paired t-test shows that the difference between our best performing model and the best baseline is statistically significant ($p<0.05$). Boldface denotes the best score and underline denotes the second best.

The rest of the generation process is the same as in Section 3.2 and 3.3.

4 Experiments

4.1 Compared Models

We compare with the best two existing models for each pun type. **Homophonic**: SurGen (He et al., 2019), a retreive-and-edit model using the local-global surprisal principle; and LCR (Yu et al., 2020), the SOTA model that first finds appropriate lexical constraints and then rewrites the sentence. **Homographic**: Pun-GAN (Luo et al., 2019), a model that adopts GANs to encourage ambiguity; AmbiPun (Mittal et al., 2022), the SOTA model that generates puns by including contexts words from both senses. We also compare the ablations of our own models: the base GPT-2 model where a random word and phrase is given, the base model with the label predictor added or the selected context word and phrase (which we call '+ select') added, and best model that includes both the label predictor and selection.

4.2 Automatic Evaluation

For each system, we compute the ambiguity, distinctiveness, and the surprisal ratio (Kao et al., 2016; He et al., 2019), and report the results in Table 3. For both pun types, our model surpasses the best baseline by a large margin in terms of distinctiveness, meaning that our model supports distinct viewpoints in the sentence. Notably, our surprisal ratio surpasses that in human-written puns.

Homophonic Pun	Success	Informative	Funny
LCR	39%	2.11	2.14
SurGen	42%	2.74	2.35
Base GPT-2	16%	2.52	1.56
+ label predictor*	40%	2.96	2.04
+ select*	49%	3.35	2.57
+ both*	56%	3.60	2.96
Human	**89%**	**4.56**	**4.04**
Homographic Pun	Success	Informative	Funny
Pun-GAN	20%	1.72	1.54
AmbiPun	44%	2.76	2.40
Base GPT-2	14%	2.17	1.55
+ label predictor*	29%	2.84	2.03
+ select*	43%	3.13	2.51
+ both*	47%	3.32	2.83
Human	**85%**	**4.23**	**3.87**

Table 4: Results of human evaluation on pun success rate, informativeness and funniness. The * indicates ablations of our method. Boldface denotes the best score and underline denotes the second best.

Moreover, He et al. (2019) have shown that while higher D and S scores usually indicate higher quality, that is not the case for ambiguity. Intuitively, since many ambiguous sentences are not informative (e.g. "I went to the bank"), ambiguity alone is insufficient. Our results correlate with the findings that A is useful in distinguishing non-puns from puns, while D and S are useful when spotting good and funny puns from bad or boring puns. Besides, our statistics show that human tend to context the pun word more when writing homophonic puns: 24% of the words are distinct to the pun word, versus 14% for the alternative word. This partially explains the imbalance between D1 and D2 for human-written puns. Our label predictor also learns such distribution and steers base GPT-2 more towards the pun word.

4.3 Human Evaluation

We ask qualified workers to judge if the given sentence is a successful pun, and rate the informativeness (or specificness) and funniness on a scale from 1 to 5. The evaluation details can be found in Appendix B. Results in Table 4 show that our model achieves the highest success rate and is the most informative and funny among all machines. We also observe that the improvements over homographic puns are smaller than that of homophonic puns. Upon error analysis, we find that half of the failure cases of homographic puns are due to inappropriate substitute pun/alternative words. Instead of using the sense keys provided in WordNet (Miller, 1995), if the user can manually provide the sense definitions to ensure the substitute pun pair is reasonable, such a bottleneck shall be resolved.

Pun pair	mane-main
LCR	The mane object of the hair was accomplished.
SurGen	A trot later, he was sitting away from the mane dining area.
Ours	In some places, hair also makes up the mane entrance to fashion salons.
Human	Lions don't have to worry about every little detail in life, just the mane thing.

Pun pair	sentence $\Longrightarrow$clause-punishment
Pun-GAN	Due to the sentence it is in the United States.
AmbiPun	The sentence is ungrammatical. The jury didn't hear it.
Ours	The language on a two-page sentence for fraud is full of guilt.
Human	The judge has got a stutter. Looks like I am not getting a sentence.

Table 5: Example outputs of different models. The pun pairs are randomly selected.

4.4 Ablation and Case Study

To validate the effectiveness of each proposed module, we report their performance in Table 3 and 4 and a bar chart in Appendix B for more straightforward illustration. Both the label predictor and the word/phrase selection process positively contribute to the outputs, and it works best when combined.

A comparison between our model and the baselines is in Table 5. Although existing approaches also include related words for semantic relevance, they tend to be too vague (e.g. LCR and Pun-GAN) or abrupt (e.g. 'a trot later' by SurGen). We also showcase the outputs for two more pun pairs along with the predicted token types in Appendix C. Both results demonstrate that our model is best at generating informative puns with humorous effects.

5 Conclusion

We propose a novel pun generation approach that incorporates three humor principles. To this end, we learn the sentence structures from human-written puns, and convert the task of homographic pun generation to homophonic pun generation. Our model achieves strong performance for both types.

6 Limitations

We discuss several limitations of this work to inspire future research directions. First, our method rely on a small amount of human written puns as a training corpora, and thus might not work well for low resource languages. Second, as can be seen in Table 2, the overall performance of the label predictor is not perfect. While we argue that the task of predicting the type of the next word is naturally difficult as there can be multiple good candidates,

the errors of the label predictor may propagate and lead to unnatural outputs.

Third, our system could not generate homographic puns as successfully as homophonic ones. Human evaluation and further error analysis show that the main reason of failure is that the generated substitute pun-alternative word pair is bad. Given a homographic pun word, we are currently retrieving its two sense definitions from WordNet (Miller, 1995) using the sense keys provided in the SemEval 2017 dataset (Miller et al., 2017), where the retrieved sense definitions are sometimes imprecise. Future directions include refining the procedure of finding the substitute pun-alternative word pairs, and curating a more accurate definition dataset for homograpic pun words.

Our proposed method is independent of the specific language model being used. The selection process is purely unsupervised and our label predictor can be theoretically combined with any language model as long as we can obtain the top k tokens it produces. Another future direction includes applying our technique to steer the GPT-3 (Brown et al., 2020) or GPT-J model[4] to generate humorous puns.

Acknowledgments

The authors would like to thank the members of PLUSLab and the anonymous reviewers for helpful comments. The research is suported in part by a Meta SRA. Yufei Tian is supported by an Amazon Fellowship.

Ethics

Our proposed methods are based on the pretrained language model. It is known that pretrained language models could capture the bias reflected in the training data (Sheng et al., 2019; Wallace et al., 2019). Considering the nature of humorous puns, our models may potentially generate offensive content for certain groups or individuals. We suggest to carefully examine the potential biases before deploying the models to real-world applications.

References

Issa Annamoradnejad and Gohar Zoghi. 2020. Colbert: Using bert sentence embedding for humor detection. *arXiv preprint arXiv:2004.12765*.

Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, Online. Association for Computational Linguistics.

Pavel Braslavski, Vladislav Blinov, Valeria Bolotova, and Katya Pertsova. 2018. How to evaluate humorous response generation, seriously? In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval*, pages 225–228.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2019*, pages 1734–1744. Association for Computational Linguistics (ACL).

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

Marie Lebert. 2009. A short history of ebooks.

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*.

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.

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

Tristan Miller, Christian F 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.

Anirudh Mittal, Yufei Tian, and Nanyun Peng. 2022. Ambipun: Generating humorous puns with ambiguous context. *NAACL*.

[4] https://huggingface.co/EleutherAI/gpt-j-6B

Fanchao Qi, Lei Zhang, Yanhui Yang, Zhiyuan Liu, and Maosong Sun. 2020. Wantwords: An open-source online reverse dictionary system. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–181.

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

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*.

Jiao Sun, Anjali Narayan-Chen, Shereen Oraby, Alessandra Cervone, Tagyoung Chung, Jing Huang, Yang Liu, and Nanyun Peng. 2022a. Expunations: Augmenting puns with keywords and explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

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.

Zhiwei Yu and Xiaojun Wan. 2019. How to avoid sentences spelling boring? towards a neural approach to unsupervised metaphor generation. 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 861–871, Minneapolis, Minnesota. 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.

Appendix

A Implementation Details

A.1 Experimental Settings

For the label predictor described in section 3.2, we finetune the bert-base model for 6 epochs. The training time is only 20 minutes on a single NVIDIA A100 GPU. As for the finetuning the language model to learn the task of generating a sentence that includes the given keyword and phrase described in Section 3.3, we finetune GPT-2 large for 6 epochs and the training time is about 3 hours on a single NVIDIA A100 GPU. The max sequence length for target and source is set to 50 and batch size is set to 6.

A.2 Pretrained Model Selection

For sequence classification tasks – improving the reliability of the 'ground truth' labels that contribute towards training data for the label predictor and the label prediction task itself, wee compare the base and large variants of the BERT and RoBERTa models and find no significant improvement using RoBERTa or the large variants, which is why we stick with the smallest BERT-base model. For the mask-infilling task towards obtaining an ideal phrase, we do find an improvement using RoBERTa-Large over other variants.

A.3 Human Annotation to Train/Test the Label Predictor

Recall that we collected human annotation on 430 word pairs to boost the performance of $\mathbf{Bert_c}$ and another 500 word pairs to test the model. Each data point is annotated by three people. Namely, given tw, pw, and aw, the annotators are asked to decide whether tw is distinct to pw, aw, or ambiguous. The inter annotator agreement is 0.62 for Krippendorff's alpha, indicating a strong agreement.

A.4 Decoding Details

Since GPT-2 uses BPE tokenizer and generate a subword at each decoding step, we ask the finetuned GPT-2 model to continue generating subwords until a complete word is generated. In addition, we keep a beam size of 20 at all time, so that we could obtain a list of complete words as candidate next words.

B Evaluation

Implementation of the Ambiguity-Distinctiveness Model The initial probabilistic model of ambiguity and distinctiveness is proposed by Kao et al. (2016), yet He et al. (2019) improve the implementation by training a skip-gram model to avoid human labor. For more straightforward illustration, the bar chart in Figure 2 shows the improvement after adding each component on homophonic puns. Note that the numbers in this bar chart is the same as that in Table 3 in the main paper.

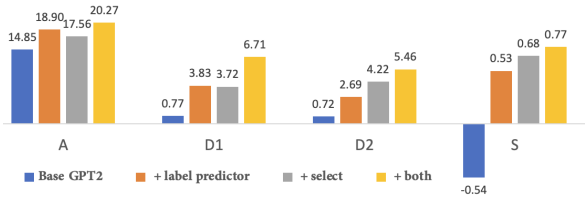

Figure 2: Bar chart showing the improvement after adding each component on homophonic puns.

Pun pair: Principal-principle

-	D2	D1	-	A	A
The	head	**teacher**	exercises	him	occasionally,
D2				A	D1
as the guiding *principal* of				their	school.

Pun pair: Taxi-tax

-	D2	D1	D1	A	
An	unlucky	cab	**driver**	might	
D2		A	-	A	A
get in all that *taxi* trouble		by	forgetting	to	tell
A	D1	A	-	A	D1
his	passenger	how	much	he	wanted.

Figure 3: Two randomly sampled examples generated by our model and the predicted labels. The context words and the extracted phrases are in boldface.

Human Evaluation The workers are paid $20 per hour. For pun success judgement (yes/no question), we take the majority vote among three workers, while for specificness and funniness (1 to 5), we take the average ratings. We then use the pairwise kappa coefficient to measure the inter-annotator agreement (IAA). The average inter-annotator agreement of all raters for pun success, specificness and funniness are 0.59, 0.44 and 0.47, meaning that annotators moderately agree with each other. Considering the subjectivity of this task (Braslavski et al., 2018), we argue that our collected results are reasonably reliable for the purpose of pun generation. Besides, we conducted paired t-test and show that the success rate and funniness ratings of our systems differentiate from the best baseline model with statistical significance (p-value < 0.05).

C More generated examples by our model

Figure 3 shows two randomly sampled examples generated by our model and the predicted labels. We can see that the extracted phrases (i.e., 'as the guiding principal of' and 'get in all that taxi trouble') arouses surprise when people read it the local context, while making sense in the global context. In addition, the label predictor successfully steer the base GPT-2 model to generate more ambigu-

ously or distinctively at each step, resulting in humorous puns.