“The Boating Store Had Its Best Sail Ever”: Pronunciation-attentive Contextualized Pun Recognition

Yichao Zhou, Jyun-yu Jiang, Jieyu Zhao, Kai-Wei Chang and Wei Wang
Computer Science Department
University of California, Los Angeles
{yz, jyunyu, jyzhao, kwchang, weiwang}@cs.ucla.edu

Abstract

Humor plays an important role in human languages and it is essential to model humor when building intelligence systems. Among different forms of humor, puns perform word-play for humorous effects by employing words with double entendre and high phonetic similarity. However, identifying and modeling puns are challenging as puns usually involved implicit semantic or phonological tricks. In this paper, we propose Pronunciation-attentive Contextualized Pun Recognition (PCPR) to perceive human humor, detect if a sentence contains puns and locate them in the sentence. PCPR derives contextualized representation for each word in a sentence by capturing the association between the surrounding context and its corresponding phonetic symbols. Extensive experiments are conducted on two benchmark datasets. Results demonstrate that the proposed approach significantly outperforms the state-of-the-art methods in pun detection and location tasks. In-depth analyses verify the effectiveness and robustness of PCPR.

1 Introduction

During the last decades, social media has promoted the creation of a vast amount of humorous web contents (Nijholt et al., 2017). Automatic recognition of humor has become an important task in the area of figurative language processing, which can benefit various downstream NLP applications such as dialogue systems, sentiment analysis, and machine translation (Melby and Warner, 1995; Augello et al., 2008; Ghosh et al., 2015; Bertero and Fung, 2016; Blinov et al., 2019). However, humor is one of the most complicated behaviors in natural language semantics and sometimes it is even difficult for humans to interpret. In most cases, understanding humor requires adequate background knowledge and a rich context.

	Homographic Puns
1.	Did you hear about the guy whose whole left side was cut off? He’s **all right** now.
2.	I’d tell you a chemistry joke but I know I wouldn’t get a **reaction**.
	Heterographic Puns
1.	The boating store had its best **sail (sale)** ever.
2.	I lift weights only on Saturday and Sunday because Monday to Friday are **weak (week)** days.

Table 1: Examples of homographic and heterographic puns.

Puns are a form of humorous approaches using the different meanings of identical words or words with similar pronunciations to explain texts or utterances. There are two main types of puns. Homographic puns rely on multiple interpretations of the same word. As shown in Table 1, the phrase *all right* means *good condition* or *opposite to left*; the word *reaction* means *chemical change* or *action*. The two meanings of the same expression are consistent with its context, which creates a humorous pun in both sentences when there is a clear contrast between two meanings. On the other hand, heterographic puns take advantage of phonologically same or similar words. For example, the word pairs *sale* and *sail*, *weak* and *week* in Table 1 have the same or similar pronunciations. The sentences are funny because both words fit the same context. Understanding puns is a big fish to fry for deep comprehension of complex semantics.

These two forms of puns have been studied in literature from different angles. To recognize puns in a sentence, word sense disambiguation techniques (WSD) (Navigli, 2009) have been employed to identify the equitable intention of words in utterances (Pedersen, 2017). External knowledge bases such as WordNet (Miller, 1998b) have been applied in determining word senses of pun words (Oele and Evang, 2017). However, these methods cannot tackle heterographic puns with distinct word spellings and knowledge bases that only contain a limited vocabulary. To resolve the issues of sparseness and heterographics, the word embedding techniques (Mikolov et al., 2013; Pennington et al., 2014) provide flexible representations to model puns (Hurtado et al., 2017; Indurthi and Oota, 2017; Cai et al., 2018). However, a word may have different meanings regarding its contexts. Especially, an infrequent meaning of the word might be utilized for creating a pun. Therefore, static word embeddings are insufficient to represent words. In addition, some puns are created by replacing a word with another word with the same or similar pronunciation as examples shown in Table 1. Therefore, to recognize puns, it is essential to model the association between words in the sentence and the pronunciation of words. Despite existing approaches attempt to leverage phonological structures to understand puns (Doogan et al., 2017; Jaech et al., 2016), there is a lack of a general framework to model these two types of signals in a whole.

In this paper, we propose Pronunciation-attentive Contextualized Pun Recognition (`PCPR`) to jointly model the contextualized word embeddings and phonological word representations for pun recognition. To capture the phonological structures of words, we break each word into a sequence of phonemes as its pronunciation so that homophones can have similar phoneme sets. For instance, the phonemes of the word *pun* are {`P`, `AH`, `N`}. In `PCPR`, we construct a pronunciation attentive module to identify important phonemes of each word, which can be applied in other tasks related to phonology. We jointly encode the contextual and phonological features into a self-attentive embedding to tackle both pun detection and location tasks. We summarize our contributions as following.

- To the best of our knowledge, `PCPR` is the first work to jointly model contextualized word embeddings and pronunciation embeddings to recognize puns. Both contexts and phonological properties are beneficial to pun recognition.
- Extensive experiments are conducted on two benchmark datasets. `PCPR` significantly outperforms existing methods in both pun detection and pun location. In-depth analyses also verify the effectiveness and robustness of `PCPR`.
- We release our implementations and pre-trained phoneme embeddings at `https://github.com/joey1993/pun-recognition` to facilitate future research.

2 Related Work

Pun Recognition and Generation To recognize puns, Miller et al. (2017) summarize several systems for the SemEval 2017 tasks. To detect the pun, Pedersen (2017) supposes that if there is one pun in the sentence, when adopting different Word Sense Disambiguation (WSD) methods, the sense assigned to the sentence will be different. To locate the pun, based on the WSD results for pun detection, they choose the last word which changes the senses between different WSD runs. Even though this method can tackle both homographic and heterographic pun detection, it does not use any pretrained embedding model. Xiu et al. (2017) detect the pun in the sentence using similarity features which are calculated on sense vectors or cluster center vectors. To locate the pun, they use an unsupervised system by scoring each word in the sentence and choosing the word with the smallest score. However, this model exclusively relies on semantics to detect the heterographic puns but ignores the rich information embedded in the pronunciations. Doogan et al. (2017) leverage word embeddings as well as the phonetic information by concatenating pronunciation strings, but the concatenation has limited expression ability. They also mention that their systems suffer for short sentences as word embeddings do not have much context information.

Besides, Zou and Lu (2019) jointly detect and locate the pun from a sequence labeling perspective by employing a new tagging schema. Diao et al. (2018) expand word embeddings using WordNet to settle the polysemy of homographic puns, following by a neural attention mechanism to extract the collocation to detect the homographic pun. However, all these methods only make use of limited context information. Other than the pun recognition, Yu et al. (2018) generate homographic puns without requiring any pun data for training. He et al. (2019) improve the homographic pun generation based on the "local-global surprisal principle" which posits that the pun word and the alternative word have a strong association with the distant and immediate context respectively.

Pronunciation Embeddings Word embeddings assign each word with a vector so that words with similar semantic meanings are close in the embedding space. Most word embedding models only make use of text information and omitting the rich information contained in the pronunciation. However, the pronunciation is also an important part of the language (Zhu et al., 2018). Prior studies have demonstrated that the phonetic information can be used in speech recognition (Bengio and Heigold, 2014), spell correction (Toutanova and Moore, 2002) and speech synthesis (Miller, 1998a). By projecting to the embedding space, words sound alike are nearby to each other (Bengio and Heigold, 2014). Furthermore, Kamper et al. (2016) make use of word pairs information to improve the acoustic word embedding. Zhu et al. (2018) show that combining the pronunciation with the writing texts can help to improve the performance of word embeddings. However, these pronunciation embeddings are word-level features, while in our approach, we make use of syllabic pronunciations which is phoneme-level and could help with the out-of-vocabulary (OOV) situation. Luo et al. (2019) also propose an adversarial generative network for pun generation, which does not require any pun corpus.

Contextualized Word Embeddings Traditional word embeddings assign a fixed vector to one word even if the word has multiple meanings under different contexts (e.g., "the river bank" v.s. "the commercial bank"). McCann et al. (2017) combine the pivot word embeddings as well as the contextual embeddings generated by an encoder from a supervised neural machine translation task. Peters et al. (2017) enrich the word embeddings by the contextual information extracted from a bidirectional language model. (Devlin et al., 2018) learn the language embedding by stacking multiple transformer layers with masked language model objective which advances the state-of-the-art for many NLP tasks. Yang et al. (2019) enable learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order and solve the problem of pretrain-finetune discrepancy.

3 Pronunciation-attentive Contextualized Pun Recognition

In this section, we first formally define the problem and then introduce the proposed method, PCPR.

3.1 Problem Statement

Suppose the input text consists of a sequence of N words $\{w_1, w_2, \cdots, w_N\}$. For each word w_i with M_i phonemes in its pronunciation, the phonemes are denoted as $R(w_i) = \{r_{i,1}, r_{i,2}, \cdots, r_{i,M_i}\}$, where $r_{i,j}$ is the j-th phoneme in the pronunciation of w_i. These phonemes are given by a dictionary. In this paper, we aim to recognize potential puns in the text with two tasks, including pun detection and pun location, as described in the following.

Task 1: Pun Detection. The pun detection task identifies whether a sentence contains a pun. Formally, the task is modeled as a classification problem with binary label y^D.

Task 2: Pun Location. Given a sentence containing at least a pun, the pun location task aims to unearth the pun word. More precisely, for each word w_i, we would like to predict a binary label y_i^L that indicates if w_i is a pun word.

In addition to independently solving the above two tasks, the ultimate goal of pun recognition is to build a pipeline from scratch to detect and then locate the puns in texts. Hence, we also evaluate the end-to-end performance by aggregating the solutions for two tasks.

3.2 Framework Overview

Figure 1 shows the overall framework of the proposed Pronunciation-attentive Contextualized Pun Recognition (PCPR). For each word in the input text, we first derive two continuous vectors, including contextualized word embedding and pronunciation embedding, as representations in different aspects. Contextualized word embeddings derive appropriate word representations with consideration of context words and capture the accurate semantics in the text. To learn the phonological characteristics, each word is divided into phonemes while each phoneme is projected to a phoneme embedding space, thereby obtaining pronunciation embeddings with the attention mechanism (Bahdanau et al., 2015). Finally, a self-attentive encoder blends contextualized word embeddings and pronunciation embeddings to capture the overall semantics for both pun detection and location.

3.3 Contextualized Word Embeddings

The context is essential for interpreting a word in the text. Hence, we propose to apply contextualized word embeddings to derive word representations. In the framework of PCPR, any contextualized word embedding method, such as BERT (Devlin et al., 2018), ELMo (Peters et al., 2018), and XLNet (Yang et al., 2019), can be utilized. Here, we choose BERT to derive contextualized word embeddings without loss of generality.

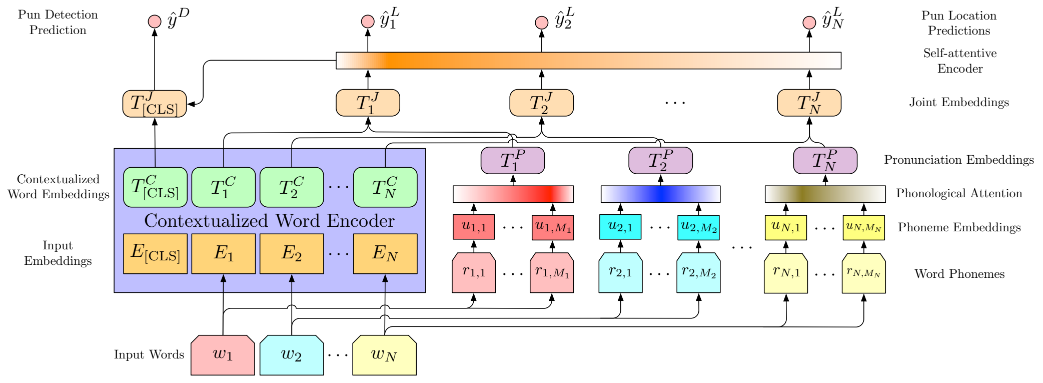


Figure 1: The overall framework of PCPR. We leverage the self-attention mechanism to jointly model contextualized embeddings and phonological representations. PCPR can tackle both pun detection and pun location tasks.

BERT deploys a multi-layer bidirectional encoder based on transformers with multi-head self-attention (Vaswani et al., 2017) to model words in the text after integrating both word and position embeddings (Sukhbaatar et al., 2015). As a result, for each word, a representative contextualized embedding is derived by considering both the specific word and all contexts in the document. Here we denote T_i^C as the d_C-dimensional contextualized word embedding for the word w_i. In addition, BERT contains a special token [CLS] with an embedding vector in BERT to represent the semantics of the whole input text.

3.4 Pronunciation Embeddings

To learn the phonological characteristics of words, PCPR models the word phonemes. For each phoneme $r_{i,j}$ of the word w_i, we project $r_{i,j}$ to a d_P-dimensional embedding space as a trainable vector $u_{i,j}$ to represent its phonological properties.

Based on the phoneme embeddings of a word, we apply the attention mechanism (Bahdanau et al., 2015) to simultaneously identify important phonemes and derive the pronunciation embedding T_i^P. Specifically, the phoneme embeddings are transformed by a fully-connected hidden layer to measure the importance scores α_i^P as follows:

$$v_{i,j} = \tanh(\mathcal{F}_P(u_{i,j})),$$

$$\alpha^P_{i,j} = \frac{v_{i,j}^\intercal v_s}{\sum_k v_{i,k}^\intercal v_s},$$

where $\mathcal{F}_P(\cdot)$ is a fully-connected layer with d_A outputs and d_A is the attention size; v_s is a d_A-dimensional context vector that estimates the importance score of each pronunciation embedding. Finally, the pronunciation embeddings T_i^P can be represented as the weighted combination of phoneme embeddings as follows:

$$T_i^P = \sum_j \alpha_{i,j} u_{i,j}.$$

Moreover, we can further derive the joint embedding T_i^J to indicate both word semantics and phonological knowledge for the word w_i by concatenating two different embeddings as follows:

$$T_i^J = \left[T_i^C; T_i^P\right].$$

Note that the joint embeddings are d_J-dimensional vectors, where $d_J = d_C + d_P$.

3.5 Pronunciation-attentive Contextualized Embedding with Self-attention

For the task of pun detection, understanding the meaning of input text is essential. Due to its advantages of interpretability over convolutional neural network (LeCun et al., 1995) and recurrent neural network (Schuster and Paliwal, 1997), we deploy the self-attention mechanism (Vaswani et al., 2017) to capture the overall semantics represented in the joint embeddings. For each word w_i, the self-attention mechanism estimates an importance vector α_i^S:

$$\mathcal{F}_S(T) = \text{Softmax}(\frac{TT^\intercal}{\sqrt{d}})T,$$

$$\alpha_i^S = \frac{\exp(\mathcal{F}_S(T_i^J))}{\sum_j \exp(\mathcal{F}_S(T_j^J))},$$

where $\mathcal{F}_S(\cdot)$ is the function to estimate the attention for queries, and d is a scaling factor to avoid extremely small gradients. Hence, the self-attentive embedding vector is computed by aggregating joint embeddings:

$$T^J_{[\text{ATT}]} = \sum_i \alpha_i^S \cdot T_i^J.$$

Note that the knowledge of pronunciations is considered by the self-attentive encoder but not the contextualized word encoder. Finally, the pronunciation-attentive contextualized representation for the whole input text can be derived by concatenating the overall contextualized embedding and the self-attentive embedding:

$$T^J_{\texttt{[CLS]}} = \left[T^C_{\texttt{[CLS]}}; T^J_{\texttt{[ATT]}}\right].$$

Moreover, each word w_i is benefited from the self-attentive encoder and is represented by a joint embedding:

$$T^J_{\texttt{i, [ATT]}} = \alpha^S_i \cdot T^J_i.$$

3.6 Inference and Optimization

Based on the joint embedding for each word and the pronunciation-attentive contextualized embedding for the whole input text, both tasks can be tackled with simple fully-connected layers.

Pun Detection. Pun detection is modeled as a binary classification task. Given the overall embedding for the input text $T^J_{\texttt{[CLS]}}$, the prediction $\hat{y}^D$ is generated by a fully-connected layer and the softmax function:

$$\hat{y}^D = \operatorname*{argmax}_{k \in \{0,1\}} \mathcal{F}_D(T^J_{\texttt{[CLS]}})_k,$$

where $\mathcal{F}_D(\cdot)$ derives the logits of two classes in binary classification.

Pun Location. For each word w_i, the corresponding self-attentive joint embedding $T^J_{\texttt{i, [ATT]}}$ is applied as features for pun location. Similar to pun detection, the prediction $\hat{y}^L_i$ is generated by:

$$\hat{y}^L_i = \operatorname*{argmax}_{k \in \{0,1\}} \mathcal{F}_L(T^J_{\texttt{i, [ATT]}})_k,$$

where $\mathcal{F}_L(\cdot)$ derives two logits for classifying if a word is a pun word.

Since both tasks focus on binary classification, we optimize the model with cross-entropy loss.

4 Experiments

In this section, we describe our experimental settings and explain the results and interpretations. We will verify some basic assumptions of this paper: (1) the contextualized word embeddings and pronunciation embeddings are both beneficial to the pun detection and location tasks; (2) the attention mechanism can improve the performance.

Dataset	SemEval		PTD
	Homo	Hetero	
Examples w/ Puns	1,607	1,271	2,423
Examples w/o Puns	643	509	2,403
Total Examples	2,250	1,780	4,826

Table 2: Data statistics. "Homo" and "Hetero" denote homographic and heterographic puns. Pun detection employs all of the examples in the two datasets while pun location only exploits the examples with puns in SemEval due to the limitation of annotations.

4.1 Experiment settings

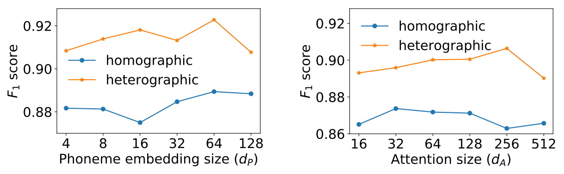
Experimental Datasets. We conducted experiments on the SemEval 2017 shared task 7 dataset[1] (SemEval) (Miller et al., 2017) and the Pun of The Day dataset (PTD) (Yang et al., 2015). For pun detection, the SemEval dataset consists of $4,030$ and $2,878$ examples for pun detection and location while each example with a pun can be a homographic or heterographic pun. In contrast, the PTD dataset contains $4,826$ examples without labels of pun types. Table 2 further shows the data statistics. The two experimental datasets are the largest publicly available benchmarks that are used in the existing studies. SemEval-2017 dataset contains punning and non-punning jokes, aphorisms, and other short texts composed by professional humorists and online collections. Hence, we assume the genres of positive and negative examples should be identical or extremely similar.

Evaluation Metrics. We adopt precision (P), recall (R), and F_1-score (Schütze et al., 2007; Powers, 2011) to compare the performance of PCPR with previous studies in both pun detection and location. More specifically, we apply 10-fold cross-validation to conduct evaluation. For each fold, we randomly select 10% of the instances from the training set for development. To conduct fair comparisons, we strictly follow the experimental settings in previous studies (Zou and Lu, 2019; Cai et al., 2018) and include their reported numbers in the comparisons.

Implementation Details. For data pre-processing, all of the numbers and punctuation marks are removed. The phonemes of each word are derived by the CMU Pronouncing Dictionary[2]. We initialize the phoneme embeddings by using the *fastText*

[1]http://alt.qcri.org/semeval2017/task7/

[2]http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/



(a) Phoneme emb. size d_P (b) Attention size d_A

Figure 2: Pun location performance over different phoneme embedding sizes d_P and attention sizes d_A on the SemEval dataset.

word embedding (Mikolov et al., 2018) trained on Wikipedia articles[3] crawled in December, 2017. The PCPR is implemented in PyTorch while the fused Adam optimizer (Kingma and Ba, 2014) optimizes the parameters with an initial learning rate of 5×10^{-5}. The dropout and batch size are set as 10^{-1} and 32. We follow BERT (BASE) (Devlin et al., 2018) to use 12 Transformer layers and self-attention heads. To clarify, in PCPR, tokens and phonemes are independently processed, so the tokens processed with WordPiece tokenizer (Wu et al., 2016) in BERT are not required to line up with phonemes for computations. To deal with the out-of-vocabulary words, we use the output embeddings of the first WordPiece tokens as the representatives, which is consistent with many state-of-the-art named entity recognition approaches (Devlin et al., 2018; Lee et al., 2019). We also create a variant of PCPR called CPR by exploiting only the contextualized word encoder without considering phonemes to demonstrate the effectiveness of pronunciation embeddings.

To tune the hyperparameters, we search the phoneme embedding size d_P and the attention size d_A from $\{8, 16, 32, 64, 128, 256, 512\}$ as shown in Figure 2. For the SemEval dataset, the best setting is $(d_P = 64, d_A = 256)$ for the homographic puns while heterographic puns favor $(d_P = 64, d_A = 32)$. For the PTD dataset, $(d_P = 64, d_A = 32)$ can reach the best performance.

Baseline Methods. We compare PCPR with several baseline methods.

For the SemEval dataset, nine baseline methods are compared in the experiments, including Duluth (Pedersen, 2017), JU_CES_NLP (Pramanick and Das, 2017), PunFields (Mikhalkova and Karyakin, 2017), UWAV (Vadehra, 2017), Fermi (Indurthi and Oota, 2017), and UWaterloo (Vechtomova, 2017). While most of them extract complicated linguistic features to train rule based and machine learning based classifiers. In addition to task participants, Sense (Cai et al., 2018) incorporates word sense representations into RNNs to tackle the homographic pun location task. The CRF (Zou and Lu, 2019) captures linguistic features such as POS tags, n-grams, and word suffix to model puns. Moreover, the Joint (Zou and Lu, 2019) jointly models two tasks with RNNs and a CRF tagger.

For the PTD dataset, four baseline methods with reported performance are selected for comparisons. MCL (Mihalcea and Strapparava, 2005) exploits word representations with multiple stylistic features while HAE (Yang et al., 2015) applies a random forest model with Word2Vec and human-centric features. PAL (Chen and Lee, 2017) trains a convolutional neural network (CNN) to learn essential feature automatically. Based on existing CNN models, HUR (Chen and Soo, 2018) improves the performance by adjusting the filter size and adding a highway layer.

4.2 Experimental Results

Pun Detection. Table 3 presents the pun detection performance of methods for both homographic and heterographic puns on the SemEval dataset while Table 4 shows the detection performance on the PTD dataset. For the SemEval dataset, compared to the nine baseline models, PCPR achieves the highest performance with 3.0% and 6.1% improvements of F_1 against the best among the baselines (i.e. Joint) for the homographic and heterographic datasets, respectively. For the PTD dataset, PCPR improves against HUR by 9.6%. Moreover, the variant CPR beats all of the baseline methods and shows the effectiveness of contextualized word embeddings. In addition, PCPR further improves the performances by 2.3% and 1.1% with the attentive pronunciation feature for detecting homographic and heterographic puns, respectively. An interesting observation is that pronunciation embeddings also facilitate homographic pun detection, implying the potential of pronunciation for enhancing general language modeling.

Pun Location. Table 3 shows that the proposed PCPR model achieves highest F_1-scores on both homographic and heterographic pun location tasks with 10.9% and 15.9% incredible increment against the best baseline method. The improvement is

[3]https://dumps.wikimedia.org/enwiki/latest/

Model	Homographic Puns						Heterographic Puns					
	Pun Detection			Pun Location			Pun Detection			Pun Location		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Duluth	78.32	87.24	82.54	44.00	44.00	44.00	73.99	86.62	68.71	-	-	-
JU_CSE_NLP	72.51	90.79	68.84	33.48	33.48	33.48	73.67	94.02	71.74	37.92	37.92	37.92
PunFields	79.93	73.37	67.82	32.79	32.79	32.79	75.80	59.40	57.47	35.01	35.01	35.01
UWAV	68.38	47.23	46.71	34.10	34.10	34.10	65.23	41.78	42.53	42.80	42.80	42.80
Fermi	90.24	89.70	85.33	52.15	52.15	52.15	-	-	-	-	-	-
UWaterloo	-	-	-	65.26	65.21	65.23	-	-	-	79.73	79.54	79.64
Sense	-	-	-	81.50	74.70	78.00	-	-	-	-	-	-
CRF	87.21	64.09	73.89	86.31	55.32	67.43	89.56	70.94	79.17	88.46	62.76	73.42
Joint	91.25	93.28	92.19	83.55	77.10	80.19	86.67	93.08	89.76	81.41	77.50	79.40
CPR	91.42	94.21	92.79	88.80	85.65	87.20	93.35	95.04	94.19	92.31	88.24	90.23
PCPR	**94.18**	**95.70**	**94.94**	**90.43**	**87.50**	**88.94**	**94.84**	**95.59**	**95.22**	**94.23**	**90.41**	**92.28**

Table 3: Performance of detecting and locating puns on the SemEval dataset. All improvements of PCPR and CPR over baseline methods are statistically significant at a 95% confidence level in paired t-tests. Comparing to PCPR, CPR does not model word pronunciations. Results show that both PCPR and CPR outperform baselines. With modeling pronunciations, PCPR performs the best.

Model	P	R	F_1
MCL	83.80	65.50	73.50
HAE	83.40	88.80	85.90
PAL	86.40	85.40	85.70
HUR	86.60	94.00	90.10
CPR	98.12	**99.34**	98.73
PCPR	**98.44**	99.13	**98.79**

Table 4: Performance of pun detection on the PTD dataset.

Model	Homographic Puns			Heterographic Puns		
	P	R	F_1	P	R	F_1
Joint	67.70	67.70	67.70	68.84	68.84	68.84
PCPR	**87.21**	**81.72**	**84.38**	**85.16**	**80.15**	**82.58**

Table 5: Performance of pipeline recognition in the SemEval dastaset.

Model	P	R	F_1
PCPR	**90.43**	**87.50**	**88.94**
w/o Pre-trained Phoneme Emb.	89.37	85.65	87.47
w/o Self-attention Encoder	89.17	86.42	87.70
w/o Phonological Attention	89.56	87.35	88.44

Table 6: Ablation study on different features of PCPR for homographic pun detection on the SemEval dataset.

much larger than that on pun detection task. We posit the reason is that predicting pun locations relies much more on the comparative relations among different tokens in one sentence. As a result, contextualized word embeddings acquire an enormous advantage. By applying the pronunciation-attentive representations, different words with similar pronunciations are linked, leading to a much better pinpoint of pun word for the heterographic dataset. We notice that some of the baseline models such as UWaterloo, UWAV and PunFields have poor performances. These methods consider the word position in a sentence or calculate the inverse document frequency of words. We suppose such rule-based recognition techniques can hardly capture the deep semantic and syntactic properties of words.

Pipeline Recognition. The ultimate goal of pun recognition is to establish a pipeline to detect and then locate puns. Table 5 shows the pipeline performances of PCPR and Joint, which is the only baseline with reported pipeline performance for recognizing the homographic and heterographic puns in the SemEval dataset. Joint achieves sub-optimal performance and the authors of Joint attribute the performance drop to error propagation. In contrast, PCPR improves the F_1-scores against Joint by 24.6% and 20.0% on two pun types.

4.3 Ablation Study and Analysis

Ablation Study. To better understand the effectiveness of each component in PCPR, we conduct an ablation study on the homographic puns of the SemEval dataset. Table 6 shows the results on taking out different features of PCPR, including pre-trained phoneme embeddings, the self-attentive encoder, and phonological attention. Note that we use the average pooling as an alternative when we remove the phonological attention module. As a result, we can see the drop after removing each of the three features. It shows that all these components are essential for PCPR to recognize puns.

Attentive Weights Interpretation. Figure 3 illustrates the self-attention weights α_i^S of three ex-

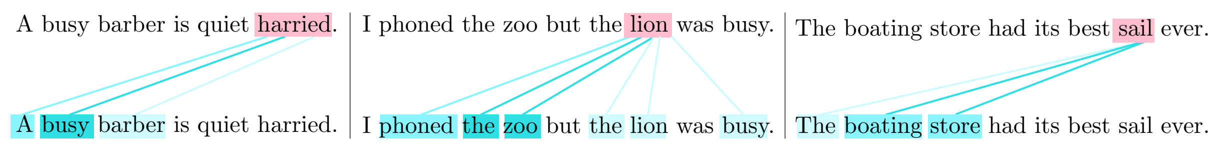
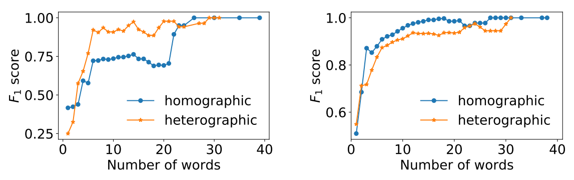

A busy barber is quiet harried.
A busy barber is quiet harried.
I phoned the zoo but the lion was busy.
I phoned the zoo but the lion was busy.
The boating store had its best sail ever.
The boating store had its best sail ever.

Figure 3: Visualization of attention weights of each pun word (marked in pink) in the sentences. A deeper color indicates a higher attention weight.

Sentence	Pun	`CPR`	`PCPR`
In the dark? Follow the son.	son	-	son
He stole an invention and then told patent lies.	patent	patent	lies
A thief who stole a calendar got twelve months.	got	-	-

Table 7: A case study of the model predictions for the pun location task of SemEval 2017.



(a) Pun Detection (b) Pun Location

Figure 4: Pun recognition performance over different text lengths for homographic and heterographic puns on the SemEval dataset.

amples from heterographic puns in the SemEval dataset. The word highlighted in the upper sentence (marked in pink) is a pun while we also color each word of the lower sentence in blue according to the magnitude of its attention weights. The deeper colors indicate higher attention weights. In the first example, *busy* has the largest weight because it has the most similar semantic meaning as *harried*. *The barber* also has relatively high weights. We suppose it is related to *hairy* which should be the other word of this double entendre. Similar, *the zoo* is corresponded to *lion* while *phone* and *busy* indicate *line* for the pun. Moreover, *boating* confirms *sail* while *store* supports *sale*. Interpreting the weights out of our self-attentive encoder explains the significance of each token when the model detects the pun in the context. The phonemes are essential in these cases because they strengthen the relationship among words with distant semantic meanings but similar phonological expressions.

Sensitivity to Text Lengths. Figure 4 shows the performance of pun detection and location over different text lengths for homographic and heterographic puns in the SemEval dataset. For both tasks, the performance gets higher when the text lengths are longer because the context information is richer. Especially in the pun detection task, we observe that our model requires longer contexts (more than 20 words) to detect the homographic puns. However, shorter contexts (less than 10 words) are adequate for heterographic pun detection, which indicates the contribution from phonological features. In short, the results verify the importance of contextualized embeddings and pronunciation representations for pun recognition.

Case Study and Error Analysis. Table 7 shows the results of a case study with the outputs of `CPR` and `PCPR`. In the first case, the heterographic pun comes from the words *son* and *sun*. `CPR` fails to recognize the pun word with limited context information while the phonological attention in `PCPR` helps to locate it. However, the pronunciation features in some cases can mislead the model to make wrong predictions. For example, *patent* in the second sentence is a homographic pun word and has several meanings, which can be found with the contextual features. Besides, the phonemes in *lies* are ubiquitous in many other words like *laws*, thereby confusing the model. In the last case, *got* is a widely used causative with dozens of meanings so that the word is hard to be recognized as a pun word with its contextual and phonological features.

5 Conclusions

In this paper, we propose a novel approach, `PCPR`, for pun detection and location by leveraging a contextualized word encoder and modeling phonemes as word pronunciations. Moreover, we would love to apply the proposed model to other problems, such as general humor recognition, irony discovery, and sarcasm detection, as the future work.

Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments.

References

Agnese Augello, Gaetano Saccone, Salvatore Gaglio, and Giovanni Pilato. 2008. Humorist bot: Bringing computational humour in a chat-bot system. In *CISIS 2008*, pages 703–708.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Samy Bengio and Georg Heigold. 2014. Word embeddings for speech recognition. In *ISCA 2014*.

Dario Bertero and Pascale Fung. 2016. Predicting humor response in dialogues from tv sitcoms. In *ICASSP 2016*, pages 5780–5784.

Vladislav Blinov, Valeria Bolotova-Baranova, and Pavel Braslavski. 2019. Large dataset and language model fun-tuning for humor recognition. In *ACL 2019*, pages 4027–4032.

Yitao Cai, Yin Li, and Xiaojun Wan. 2018. Sense-aware neural models for pun location in texts. In *ACL 2018*, pages 546–551.

Lei Chen and Chong MIn Lee. 2017. Predicting audience's laughter using convolutional neural network. *arXiv preprint arXiv:1702.02584*.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *NAACL 2018*, pages 113–117.

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*.

Yufeng Diao, Hongfei Lin, Di Wu, Liang Yang, Kan Xu, Zhihao Yang, Jian Wang, Shaowu Zhang, Bo Xu, and Dongyu Zhang. 2018. Weca: A wordnet-encoded collocation-attention network for homographic pun recognition. In *EMNLP 2018*, pages 2507–2516.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom savant at semeval-2017 task 7: Detection and interpretation of english puns. In *SemEval-2017*, pages 103–108.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. In *SemEval 2015*, pages 470–478.

He He, Nanyun Peng, and Percy Liang. 2019. Pun generation with surprise. In *NAACL 2019*.

Lluís-F Hurtado, Encarna Segarra, Ferran Pla, Pascual Carrasco, and José-Angel González. 2017. Elirf-upv at semeval-2017 task 7: Pun detection and interpretation. In *SemEval-2017*, pages 440–443.

Vijayasaradhi Indurthi and Subba Reddy Oota. 2017. Fermi at semeval-2017 task 7: Detection and interpretation of homographic puns in english language. In *SemEval-2017*, pages 457–460.

Aaron Jaech, Rik Koncel-Kedziorski, and Mari Ostendorf. 2016. Phonological pun-derstanding. In *ACL 2016*, pages 654–663.

Herman Kamper, Weiran Wang, and Karen Livescu. 2016. Deep convolutional acoustic word embeddings using word-pair side information. In *ICASSP 2016*, pages 4950–4954. IEEE.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

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)*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NeurIPS 2017*, pages 6294–6305.

Alan K Melby and Terry Warner. 1995. *The possibility of language: A discussion of the nature of language, with implications for human and machine translation*, volume 14. John Benjamins Publishing.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *EMNLP 2005*, pages 531–538.

Elena Mikhalkova and Yuri Karyakin. 2017. Punfields at semeval-2017 task 7: Employing roget's thesaurus in automatic pun recognition and interpretation. *arXiv preprint arXiv:1707.05479*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *LREC 2018*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Corey Andrew Miller. 1998a. Pronunciation modeling in speech synthesis.

George Miller. 1998b. *WordNet: An electronic lexical database*. MIT press.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. Semeval-2017 task 7: Detection and interpretation of english puns. In *SemEval-2017*, pages 58–68.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):10.

Anton Nijholt, Andreea I Niculescu, Valitutti Alessandro, and Rafael E Banchs. 2017. Humor in human-computer interaction: a short survey.

Dieke Oele and Kilian Evang. 2017. Buzzsaw at semeval-2017 task 7: Global vs. local context for interpreting and locating homographic english puns with sense embeddings. In *SemEval-2017*, pages 444–448.

Ted Pedersen. 2017. Duluth at semeval-2017 task 7: Puns upon a midnight dreary, lexical semantics for the weak and weary. *arXiv preprint arXiv:1704.08388*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP 2014*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL 2018*, pages 2227–2237.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

David Martin Powers. 2011. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Aniket Pramanick and Dipankar Das. 2017. Ju cse nlp @ semeval 2017 task 7: Employing rules to detect and interpret english puns. In *SemEval-2017*, pages 432–435.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2007. *An introduction to information retrieval*. Cambridge University Press,.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction.

Ankit Vadehra. 2017. Uwav at semeval-2017 task 7: Automated feature-based system for locating puns. In *SemEval-2017*, pages 449–452.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Olga Vechtomova. 2017. Uwaterloo at semeval-2017 task 7: Locating the pun using syntactic characteristics and corpus-based metrics. In *SemEval-2017*, pages 421–425.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yuhuan Xiu, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 7: Using supervised and unsupervised methods to detect and locate english puns. In *SemEval-2017*, pages 453–456.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *EMNLP 2015*, pages 2367–2376.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zhiwei Yu, Jiwei Tan, and Xiaojun Wan. 2018. A neural approach to pun generation. In *ACL 2018*.

Wenhao Zhu, Xin Jin, Jianyue Ni, Baogang Wei, and Zhiguo Lu. 2018. Improve word embedding using both writing and pronunciation. *PloS one*, 13(12):e0208785.

Yanyan Zou and Wei Lu. 2019. Joint detection and location of english puns. In *NAACL 2019*, pages 2117–2123.