# Heterographic Pun Recognition via Pronunciation and Spelling Understanding Gated Attention Network

Yufeng Diao
School of Computer Science and Technology
Dalian University of Technology, Inner Mongolia University for Nationalities, Dalian, Liaoning, China, diaoyufeng@mail.dlut.edu.cn

Hongfei Lin[†]
School of Computer Science and Technology
Dalian University of Technology, Dalian, Liaoning, China, hflin@dlut.edu.cn

Liang Yang
School of Computer Science and Technology
Dalian University of Technology, Dalian, Liaoning, China, liang@dlut.edu.cn

Xiaochao Fan
School of Computer Science and Technology
Dalian University of Technology, Xinjiang Normal University, Dalian, Liaoning, China, fxc1982@mail.dlut.edu.cn

Di Wu
School of Computer Science and Technology
Dalian University of Technology, Dalian, Liaoning, China, wudi@dlut.edu.cn

Dongyu Zhang
School of Software
Dalian University of Technology, Dalian, Liaoning, China, zhangdongyu@dlut.edu.cn

Kan Xu
School of Computer Science and Technology
Dalian University of Technology, Dalian, Liaoning, China, xukan@dlut.edu.cn

## ABSTRACT

Heterographic pun plays a critical role in human writing and literature, which usually has a similar sounding or spelling structure. It is important and difficult research to recognize the heterographic pun because of the ambiguity. However, most existing methods for this task only focus on designing features with rule-based or machine learning methods. In this paper, we propose an end-to-end computational approach – Pronunciation Spelling Understanding Gated Attention (PSUGA) network. For pronunciation, we exploit the hierarchical attention model with phoneme embedding. While for spelling, we consider the character-level, word-level, tag-level, position-level and contextual-level embedding with attention model. To deal with the two parts, we present a gated attention mechanism to control the information integration. We have conducted extensive experiments on SemEval2017 task7 and Pun of the Day datasets. Experimental results show that our approach significantly outperforms state-of-the-art methods.

[†] Corresponding author

## CCS CONCEPTS

• Applied computing • Sociology

## KEYWORDS

Heterographic Pun Recognition, Pronunciation, Spelling, Gated Attention

**ACM Reference format:**

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 *Proceedings of WWW '19: The Web Conference (WWW '19), May 13, 2019, San Francisco, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3308558.3313505

## 1 Introduction

A pun, also called paronomasia, is a form of word play that suggests two or more meanings by exploiting multiple meanings of a word or by replacing a similar sounding word for an intended humorous or rhetorical effect (Tristan and Mladen, 2016). They are widely used in written and spoken literature, intended as jokes. From Shakespeare's works to modern advertisement catchphrases (Tanaka, 1992), puns have been widely applied as a humorous device. In literature, speeches and slogans, puns are also standard rhetorical ploys where can also be used non-humorously. Both humorous and non-humorous pun, the special form of ambiguity, have been the theme of extensive and attractive works for creating statements with ambiguous and distinct meanings to arise the attractive effect.

The study of puns recognition can be seen as a respectable research topic in traditional linguistics and the cognitive sciences in Nature Language Processing (NLP). Many scholars study on classifying puns by the semantic relationship between the different meaning (polysemy), or similarly pronounced sound (phonology). For example, Redfern (1987) categorizes pun into homographic pun and heterographic pun, which exploits distinct meanings of the same written word and distinct meanings of the similar spoken word, respectively. There are many relevant research rely on this classification system. Our work also think that a pun is consisted of homographic and heterographic pun.

Homographic pun and heterographic pun both have different meanings to produce a humorous or rhetorical effect. Whereas these two types of puns have own special characteristics.

Exp.1 "Getting rid of your boat for another could cause a whole raft of problems."

A homographic pun is characterized by two or more meanings of a word, each of which leads to a different but valid interpretation. In an example 1 shown above, the word "raft" is the pun word which denoting "raft" as a batch or a type of boat.

Exp.2 "A dentist hates having a bad day at the orifice."

A heterographic pun relies on a different kind of senses between two words which have the similar sound and rhyme, or nearly spelling. In an example 2 shown that the pun word "orifice" has a similarly sounding and spelling word "office", which is associated with the previous word "dentist".

In this study, we focus on heterographic pun, defined as puns containing words that sound identical or similar spell to other words, because these puns widely used everywhere and easily obtain in the current works (Tristan and Mladen, 2016). However, the study of heterographic pun recognition in the exiting works has a challenge because of the confused and ambiguous meanings.

To settle the problem, we propose an automatic computational end-to-end Pronunciation Spelling Understanding Gated Attention network (PSUGA) to recognize heterographic pun. Our model takes pronunciation and spelling into account for heterographic recognition. Based on the experiments, the results show that our work achieves the state-of-the-art performance compared to existing methods, which use manual rules or other machine learning models. Here, our contributions are listed as follows.

(i) We apply the hierarchical attention mechanism to understand the latent semantic phoneme representation by CMU pronunciation dictionary for solving the pronunciation of heterographic pun.

(ii) We propose a neural attention network to learn the spelling relationship by character-level, word-level, tag-level, position-level and contextual embeddings for heterographic pun classification.

(iii) We propose the gated attention strategy to exploit the combination of the pronunciation and spelling which is useful for this task. Experimental results on the datasets of SemEval2017 task 7 and Pun of the Day demonstrate that our method outperforms several strong baselines.

## 2 Related Work

Puns have been discussed in rhetorical and literary criticism since ancient times and have increasingly become a respectable research topic in recent years. However, it is surprising that puns have attracted little attention for computational linguistics and natural language processing (Tristan and Mladen, 2016). In this section, we will introduce the prior work about puns and recognition methods.

Puns contain semantic ambiguous, sentence pattern, similar sound these own characteristics. Hempelmann (2008) studied the automatic recognition of pun target words and humor text generation. Recently, Miller and Gurevych (2015) proposed methods for homographic pun to detect the ambiguous multiple meanings for the word sense disambiguation. Tristan and Mladen (2016) applied a computational method for the detecting of puns in running text and for the isolation of the intended meanings, and evaluated WSD-inspired systems in a dedicated pun identification task.

Some researchers studying puns tend to have a phonological or syntactic puns rather than semantic puns. Zwicky and Zwicky (1986) provided an analysis of the properties of paronomasia puns, especially with regard to the phonological segments. Binsted et al. (1996) applied the pronunciation dictionary (Robinson 1996) and designed the similar sound rules to generate the word pairs of similar pronounce for puns generation. Kao et al. (2015) proposed a computational model of linguistic humor in puns and two information-theoretic measures, ambiguity and distinctiveness, which derived from a simple model of sentence processing. Aaron et al. (2016) considered the semantic relationship between pun and its phonologically similar target, and leveraged automatically learned phone edit probabilities. N-Hance (Sevgili et al., 2017) assumed every pun has a particularly strong association with exactly one other word in the context, and calculated the PMI between every pair of words. Idiom Savant (Doogan et al, 2017) used Google n-grams, word2vec, and CMU Pronouncing Dictionary to recognize heterographic pun. JU_CSE_NLP (Pramanick and Das, 2017) trained a hidden Markov model and cyclic dependency network, using features from a part-of-speech tagger and a syntactic parser. ECNU (Xiu et al., 2017) applied a supervised approach to pun detection by using WordNet and word2vec embeddings. These above methods mostly design useful features, and adopt rule-based or machine learning methods to recognize the heterographic pun. Here we use an end-to-end computable network for recognizing heterographic pun, combined with CMU pronunciation dictionary and spelling relationship.

Compared with pun recognition, pun generation has attracted little attention in the past decades. Hempelmann et al. (2003) modeled the factors to generate the pun and obtain the measures for pun evaluation based on the theory. Richie et al. (2005) generated pun based on the analysis of humor language. Hong et al. (2009) used to automatic extract the semantic patterns for puns generation and store the knowledge base in template form by using phonetic and semantic linguistic resources.

Some researchers focus on the application of pun in humor. Yang D et al. (2015) recognized the humor and humor anchor on the Pun of the Day dataset by incongruity, ambiguity, interpersonal effect, and phonetic style these useful semantic features. Taylor and Mazlack (2004) proposed an N-gram method with the syntactic context to detect puns for humorous effect in English jokes. Then Taylor (2009) carried out humor recognition depending on Ontological Semantics in the transforming content. Bertero and Fung (2016) used LSTM model to recognize humor based on the phonetic and textual features with the Big Bang Theory Corpus.

Bahdanau et al. (2014) proposed the attention mechanism to select the reference words in an initial language before translating other language for solving the machine translation problem. Yang et al. (2016) applied attention mechanism to settle the document-level classification. There are also some NLP tasks which employing attention mechanism, including machine comprehension (Seo et al., 2017; Yu et al., 2018), text classification (Du et al., 2017; Tan et al., 2018; Shen et al., 2018; Xu et al., 2018; Shen et al., 2018), question answering (Lu et al., 2016; Gui et al., 2017), generation (Zhang et al., 2018; Kim et al., 2018; Fan et al., 2018). In conclusion, attention model can enable to mining the latent and semantic information for improving the performance of NLP tasks.

Hence, we propose an end-to-end approach Pronunciation Spelling Understanding Gated Attention network (PSUGA) for recognizing heterographic pun, combined with CMU pronunciation dictionary, different levels of embedding and attention mechanism.

## 3 Methods

In this section, we propose our model as Pronunciation Spelling Understanding Gated Attention network (PSUGA). This model can improve the performance by considering both the phonological semantic representation and spelling relationship with character-level, word-level, tag-level, positon-level and contextual embeddings for heterographic puns recognition.

The overall architecture of our model is shown in Figure 1. It consists of two main components: a hierarchical attention convolutional neural network (CNN) as Pronunciation Understanding network, and a multi-level embedding attention network as Spelling Understanding network for text. These two components are combined by a gated attention strategy to recognize the heterographic pun. We describe the details of these two components in the following subsections.

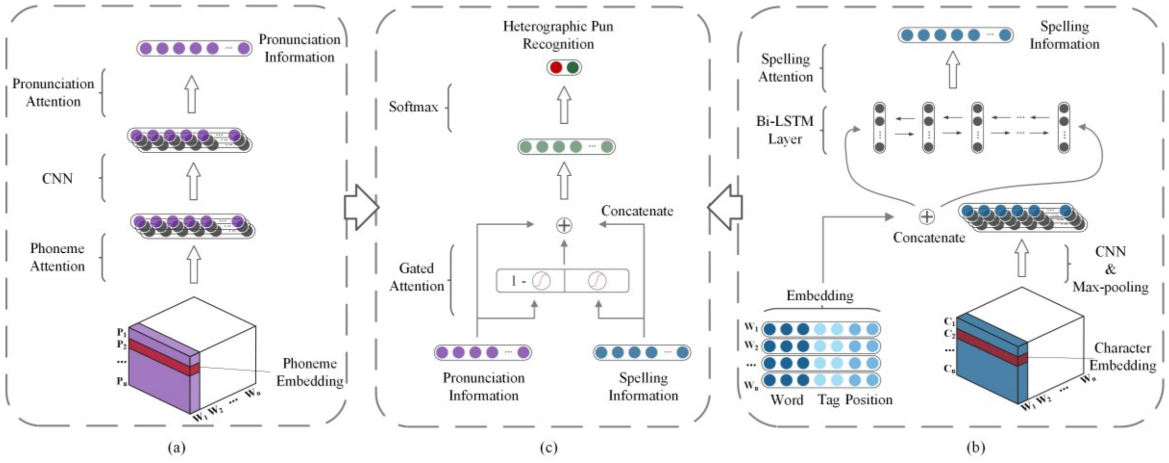

**Figure 1: The Framework of Pronunciation Spelling Understanding Gated Attention Network (PSUGA)**

### 3.1 Pronunciation Understanding Network(PUN)

Heterographic pun are mostly fabricated by replacing phonetically similar words. For example, "*What did the grape when it got stepped on? Nothing but it let out a little whine.*" The pun word is "whine", while the similar pronunciation word is "wine" which is associated with the previous word "grape". This example illustrates that one of the main challenges for heteographic pun recognition is identifying multi word expressions between the phonetically similar words.

In order to obtain phoneme representation, we introduce CMU pronouncing dictionary[1] (CMUdict), which developed by Carnegie Mellon University for designing speech synthesizer. It contains 134000 entries. Each entry is pronounced by Arpabet, which is composed of 23 vowels and 31 consonants. For example, the word "Hello" is pronounced as "HH AHO L OW1", where the number indicates stress.

A word can contain a number of phonetic symbols in CMUdict. We use all the pronunciation of an entry for the

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

speech extension, and match any pronunciation as the speech extension of a word. There are two types of speech extensions: the same pronunciation and the similar pronunciation. The former is to look up the word of same sound with this entry in CMUdict, while the latter is to make statistics on pronunciation substitution between original word and target word. Here, we apply the substitution matrix between vowels and vowels, consonants and consonants proposed by Jaech et al. (2016). If the pronunciation is found in CMUdict after phoneme substitution, it can be used as the speech extensions of the original word.

**Phoneme Embedding Layer**. Phoneme embedding layer is responsible for mapping each word to a high-dimensional vector space. Let $\{x_1,\ldots, x_M\}$ represents the words in the input sentence, where $x_i=\{p_1,\ldots,p_N\}$ represents the phonemes of the word $x_i$ and $p_N \in R^d$ is the d dimensional vector of the N-th phoneme in the target word $x_i$.

**Phoneme Attention Mechanism**. Traditional model cannot capture the important parts in sentences. In order to address this problem, we design an attention mechanism that enables model to concentrate on salient phonemes with respect to a specific word. To make full use of pronunciation information, this model uses the phoneme embedding as input to extract phoneme-specific attention signal. The formulas are as follows.

$$u_t = \tanh(w \cdot p_a + b)$$

$$\alpha_{ti} = \frac{\exp(u_{pi}^T p_{vi})}{\sum_i \exp(u_{pi}^T p_{vi})} \qquad (1)$$

$$h_t = \sum_i \alpha_{pi} p_{ti}$$

where $p_t$ is the phoneme embedding, $p_a$ and $p_v$ are the weighted matrices in the training process, $h_t$ is the phoneme vector after encoding as input of phoneme embedding layer.

**Convolutional Layer**. A convolution layer is first used to extract local n-gram features. It involves a filter $w \in R^{ld}$, which is applied to a window of l words to produce a new feature. For example, a window of words $h_{i:i+l-1}$ generates the feature map ct as follows,

$$c_t = f(w^\circ h_{i:i+l-1} + b) \qquad (2)$$

where f is ReLU function, w and b denote the weight matrix and bias respectively, $^\circ$ is a convolutional operator which can be considered as 1D inputs to the CNN, l is the length of the filter, h is the phoneme vector after phoneme attention layer, d is the dimension of the phoneme vector, $d_l$ is the dimension of the feature map $c_t$.

**Pronunciation Attention Mechanism**. We also apply an attention mechanism, which drives the model to discover the heterographic pun and use the feature map as input to extract pun-specific attention signal. The formulas are as follows.

$$u_o = \tanh(w \cdot c_a + b)$$

$$\alpha_o = \frac{\exp(u_{ci}^T c_{vi})}{\sum_i \exp(u_{ci}^T c_{vi})} \qquad (3)$$

$$r_o = \sum_i \alpha_{ci}\, c_{ti}$$

where $c_t$ is the convolutional feature map, $c_a$ and $c_v$ are the weighted matrices in the training process, $r_o$ is the pronuncation understanding vector after encoding, the dimension of $r_o$ is $d_l$.

## 3.2 Spelling Understanding Network(SUN)

Heterographic pun can rely on spelling to create the ambiguous effect. For example, "*When the church bought gas for their annual barbecue, proceeds went from the sacred to the propane.*" This pun exploits the spelling similarity between the surface sign "propane" and the latent target "profane".

This task has some challenges, in particular mapping clever near spelling into the intended form. For example, "Tom Swify" (Lessard and Levison, 1992) is one type of pun often found in the data set. Meanwhile, a large of puns is descriptions of things that have been seen and heard in the past. Moreover, heterographic pun is related with the part of speech and position. Miller points out the candidate pun words mainly consist of nouns, verbs, adjectives and adverbs in each pun at the end of the sentence. Our model introduces the useful information as different levels of embedding in the Spelling Understanding Network.

**Character Embedding Layer**. Our model accepts a sequence of encoded characters as input following Zhang et al. (2015). The encoding is prescribing an alphabet for the input language, and then quantizes each character according to 1-of-m encoding. The alphabet consists of 70 characters, including 26 English letters, 10 digits, 33 other characters and the new line character.

Character embedding layer is responsible for mapping each word to a high-dimensional vector space. Let $\{x_1,\ldots, x_M\}$ represent the words in the input sentence. Following Kim (2014), we obtain the character-level embedding of each word for the intended form with heteographic pun by using Convolutional Neural Networks (CNN). Characters are embedded into vectors, which can be considered as 1D inputs to the CNN model, and the size is the input channel size of CNN model. Then the outputs of the CNN are max-pooled to get a fixed-size vector for each word, where the dimension of outputs is $d_c$.

**Word Embedding Layer**. Word embedding layer also maps each word to a high-dimensional vector space for the latent semantic representation. Here we use pre-trained word vectors, GloVe (Pennington et al., 2014) to obtain the fixed word embedding of each word for recognizing heterographic puns. The dimension of GloVe is $d_w$.

**Tag Embedding Layer**. Part of speech information is important clue for heterographic pun recognition. Here we define five types of tags: noun, verb, adjective, adverb and other. Tag embedding layer is also capable for representing each word to a high-dimensional semantic vector space. The dimension of tag embedding is $d_t$.

**Position Embedding Layer**. Position information can represent the distance for recognizing heterographic pun. Position embedding layer is also able to represent each term into a high-dimensional semantic vector space. The dimension of tag embedding is $d_p$.

The concatenation of the character-level, word-level, tag-level and position-level embedding vectors is pass to a concatenate operation. The output vector is $Q \in R^{(d_c+d_w+d_t+d_p)\times M}$, where the dimension is $(d_c+d_w+d_t+d_p)$.

**Contextual Embedding Layer.** We leverage a Long Short-Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997) on top of the embeddings provided by the preceding layers to model the temporal interactions between words. LSTM was proposed by Hochreiter and Schmiduber (1997), which had been widely adopted for text processing. There are three gates and one cell in LSTM: an input gate $i_t$, a forget gate $f_t$, an output gate $o_t$ and a memory cell $c_t$. They are all vector in $\mathbb{R}^d$. The equations of transition are:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1}) \quad (4)$$
$$\tilde{c_t} = \tanh(W_c x_t + U_c h_{t-1})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t}$$
$$h_t = o_t \odot \tanh(c_t)$$

where $x_t$ is an input vector at the current time step,$\sigma$ is the sigmoid function and $\odot$ is the element-wise multiplication operation, $W_{\{i,f,o,c\}}, U_{\{i,f,o,c\}}, V_{\{i,f,o,c\}}$ are learned weight parameters, $h_t$ is the hidden state vector. In LSTM, the hidden state $h_t$ only encodes the front context in a forward direction but not consider the backward context.

We apply a LSTM in both directions, and concatenate the outputs of the two LSTMs (Bi-LSTM) by Graves et al. (2013) to better learn the context with heterographic pun. It has a forward LSTM $\vec{h}$ and a backward LSTM $\overleftarrow{h}$ to concatenate the hidden states of two LSTMs as the representation of the corresponding word. By doing this, the forward and backward contexts can take into account simultaneously.

Hence, we obtain $E \in R^{2d_e \times M}$ from the input concatenation of the character, word, tag and position vectors Q, where e is the neural units of LSTM model. Note that each column vector of E is 2de-dimensional because the outputs concatenate the forward and backward LSTMs, each with $d_u$ dimensional output.

**Spelling Attention Mechanism**. In order to discover the important information in spelling understanding, we design an attention mechanism, which enables model to concentrate on salient spelling. To make full use of spelling information, this model uses the contextual embedding as input to extract spelling-specific attention signal. The formulas are as follows.

$$u_f = \tanh(w \cdot e_a + b)$$
$$\alpha_{fi} = \frac{\exp(u_{ei}^T e_{vi})}{\sum_i \exp(u_{ei}^T e_{vi})} \quad (5)$$
$$s_f = \sum_i \alpha_{ei}\, e_{fi}$$

where $e_f$ is the contextual embedding combining of character-level, word-level and tag-level embeddings, $e_a$ and $e_v$ are the weighted matrices in the training process, $s_f$ is the spelling vector after encoding as the input of contextual embedding layer, the dimension of $s_f$ is $2d_e$.

**Gated Attention Mechanism**. After learning the pronunciation understanding and spelling understanding networks, the next step is to combine them to get the integrated information. We carry out two strategies as following.

**ST1: Averaged Aggregation strategy**. We assume that the information of pronunciation part and spelling part are of same importance. The integrated information is computed as:

$$R_{st1} = \alpha * r_o + (1 - \alpha) * s_f \quad (6)$$

where $\alpha$ is 0.5, the dimension of the attention gate is same as $r_o$ and $s_f$.

**ST2: Weighted Aggregation strategy**. We assume that the pronunciation information and spelling information are of different importance. We leverage gated attention to model the confidence of clues provided by the two parts. The value of the attention gate is computed as:

$$g = \sigma(w[r_o; s_f] + b) \quad (7)$$

where w is the weight matrix and b is the bias term, $\sigma$ is the sigmoid function accepting vector as input.

We use 1-g and g as the combination weights of pronunciation side and spelling side to assemble $r_o$ and $s_f$. Here the attention gate can be seen as a sentinel to control information combination between pronunciation part and spelling part. Note that the dimension of the attention gate is same as $r_o$ and $s_f$, and the integrated information is computed by weighted sum as:

$$R_{st2} = (g^\circ r_o) + ((1 - g)^\circ s_f\,) \quad (8)$$

where $^\circ$ stands for element-wise multiplication operation, g is the combination weight, $R_{st2}$ represents the final integrated information of various parts.

**Output Layer**. Follow previous work, we formulate heteographic pun recognition as a classification problem. In the output layer, we combine $r_o$, $s_f$, and $R_{st2}$ as input to a softmax classifier:

$$z_{out} = \mathrm{Softmax}(W \cdot [r_o; s_f; R_{st2}] + b) \quad (9)$$

where $z_{out} \in R^C$ is the vector of predicted probability for heterographic pun. Here C is the number of classes of pun labels, W and b are parameters of the classification layer, activation function is softmax.

## 3.3 Model Training

Here we introduce the learning and optimization details of our framework. The model can be trained in an end-to-end way by backpropagation, where objective function (loss function) is the cross-entropy loss. Here y is the target distribution and $\hat{y}$ is the predicted distribution. The goal of training model is to minimize the loss function between y and $\hat{y}$ for all examples. The formula is as following.

$$\mathrm{loss}(\theta) = -\sum_i \sum_j y_i^j \log \hat{y_i}^j + \lambda \|\theta\|^2 \quad (10)$$

where i is the index of sentence, j is the index of class, $\lambda$ is the regularization parameter and $\theta$ indicates all parameters.

For optimization method, we adopt mini-batch stochastic gradient descent (SGD) to minimize the objective function. Meanwhile, we add dropout strategy to prevent the co-adaptation of the parameters to settle the overfitting problem.

## 4 Experiments

In this section, we first introduce the dataset and evaluation metrics. Then we examine the performance of our PSUGA model comparison with existing methods. Finally, we give the detailed analysis of our proposed model.

### 4.1 Dataset and Evaluation Metrics

We conduct experiments on the widely used Pun of the Day[2] and SemEval2017 Task7[3] dataset.

**Pun of the Day**. This corpus only contains pun content in the beginning. Then it collects the negative samples from Yahoo!Answer, AP News, New York Times and Proverb to balance the distribution of positive and negative samples for the domain discrepancy.

**SemEval2017 Task7**. This corpus includes homographic and heterographic pun for recognizing and interpreting pun. We mainly focus on heterographic pun detection in pronunciation and spelling. Each text includes at least one pun. Table 1 shows a detailed statistical distribution of our datasets.

**Table 1: Statistics on SemEval2017 Task7 and Pun of the Day.**

| Dataset | Positive | Negative |
|---|---|---|
| Task 7 (Heterographic Pun) | 1271 | 509 |
| Task 7 (Homographic Pun) | 1607 | 643 |
| Pun of the Day | 2423 | 2403 |

We apply the following standard criteria precision, recall, accuracy and F1-score for evaluation, which also adopted as metrics in SemEval2017 task7 for heterographic pun recognition.

**Training Details**. In our experiments, our model is tuned with 5-fold cross validation. We apply the GloVe embedding (Pennington et al., 2014) and the dimension is 100. The size of units in LSTM is 100 and dropout is 0.5. We use learning rate decay and early stop in the training process.

### 4.2 Comparison with Existing Methods

We compare our model with many state-of-the-art methods as follows. The good performing models in SemEval2017 task7 are N-Hance, Idiom Savant, JU_CSE_NLP and ECNU (Miller et al., 2017).

- CNN: CNN without Pronunciation Spelling Understanding Gated Attention network.
- LSTM: LSTM without Pronunciation Spelling Understanding Gated Attention network.
- Bi-LSTM: Bi-LSTM without Pronunciation Spelling Understanding Gated Attention network.
- Bi-LSTM-Attention: Bi-LSTM with single attention mechanism.
- N-Hance (Sevgili et al., 2017): N-Hance system calculates the PMI between every pair of words and uses a certain threshold method to distinguish the pun.
- Idiom Savant (Doogan et al., 2017): This method uses Google n-grams, word2vec, and CMU Pronunciation Dictionary to recognize heterographic pun.
- JU_CSE_NLP (Pramanick and Das, 2017): It trains a hidden Markov and cyclic dependency network, using features from a part-of-speech tagger and syntactic parser.
- ECNU (Xiu et al., 2017): It applies a supervised machine learning approach to detect pun by using WordNet and word2vec embedding.
- PSUGA: Here we both consider the Pronunciation Understanding with hierarchical attention mechanism and phoneme embedding and Spelling Understanding with different level embeddings. They are concatenated by the gated attention mechanism.

The performance is shown in Table 2. Our PSUGA model uses Pun of the Day as the training set to obtain all the parameters, and test in the SemEval2017 task7 dataset for hetegraphic pun recognition. From the results, we have several observations that:

**Table 2: Performance of all the methods.**

| Models | P(%) | R(%) | A(%) | F1(%) |
|---|---|---|---|---|
| CNN | 81.57 | 80.73 | 78.61 | 81.15 |
| LSTM | 82.42 | 81.56 | 80.61 | 81.99 |
| Bi-LSTM | 82.71 | 82.34 | 80.82 | 82.52 |
| CNN-Attention | 81.98 | 81.26 | 79.44 | 81.62 |
| Bi-LSTM-Attention | 83.49 | 82.71 | 81.30 | 83.10 |
| N-Hance | 77.25 | 93.00 | 75.45 | 84.40 |
| Idiom Savant | 87.04 | 81.90 | 78.37 | 84.39 |
| JU_CSE_NLP | 73.67 | 94.02 | 71.74 | 82.61 |
| ECNU | 78.07 | 67.61 | 63.33 | 72.47 |
| PSUGA | 87.92 | 85.04 | 82.91 | 86.46 |

(1) LSTM has the better performance for heterographic pun recognition compared with CNN (81.99% vs. 81.15%). It shows that LSTM is able to find the global useful features for classification than CNN model. Meanwhile Bi-LSTM outperforms LSTM (82.52% vs. 81.99%), which demonstrated the two parallels of Bi-LSTM has ability to learn more context information.

[2] Pun of the Day: http://www.punoftheday.com/

[3] SemEval2017 Task7:http://alt.qcri.org/semeval2017/task7/

(2) Bi-LSTM-Attention performs slightly better than Bi-LSTM (83.10 % vs. 82.52%). Meanwhile CNN-Attention outperforms CNN (81.62 % vs. 81.15%). It shows that the attention mechanism can capture the semantic important parts to recognize heterographic pun. As we know, attention mechanism enables to improve the performance of NLP tasks.

(3) Among all the approaches, our method behaves best. It achieves the best performance (86.46% of F1) on SemEval2017 task7 and beats the best reported approach N-Hance by 2.06% on F1. It shows that the pronunciation information, spelling information and gated attention mechanism can largely affect the performance for this task.

(4) Compared with other methods, our model tends to achieve the higher precise score and accuracy score but lower recall. We argue it is the different types of additional information, which causes this phenomenon. Our model is able to learn pronunciation and spelling features, and apply the attention mechanism for providing more coherent and related clues. While for rule based methods, they usually need to fit the pun pattern by heuristic rules for obtaining high recall compared with our model.

## 4.3 Detailed Analysis

We conduct extra experiments to do detailed analysis as following.

*4.3.1 Analysis of PSUGA model* . In this subsection, we design a series of models to verify the effectiveness of our PSUGA model. First, we do not introduce the pronunciation, spelling and attention mechanism. We design N-Pron-N-Sp model which only using Bi-LSTM for recognizing heterographic pun. Then we implement our model Y-Pron-N-Sp (PUN), which considers pronunciation embedding and hierarchical attention mechanism based on CNN. Next, we design N-Pron-Y-Sp (SUN) model that employing character-level, word-level, tag-level, position-level and contextual-level embeddings to learn the spelling information based on Bi-LSTM and attention model. Then, we design our model Y-Pron-Y-Sp (PSUGA). Table 3 shows the performance of all the models.

**Table 3: Analysis of the PSUGA model.**

| Models | P(%) | R(%) | A(%) | F1(%) |
|---|---|---|---|---|
| N-Pron-N-Sp | 82.71 | 82.34 | 80.82 | 82.52 |
| Y-Pron-N-Sp (PUN) | 83.94 | 83.13 | 81.74 | 83.53 |
| N-Pron-Y-Sp (SUN) | 85.26 | 83.92 | 82.33 | 84.58 |
| Y-Pron-Y-Sp (PSUGA) | 87.92 | 85.04 | 82.91 | 86.46 |

(1) From Table 3, we can see that N-Pron-N-Sp model achieves the worse performance, which justifies the intuition of exploiting the pronunciation information, spelling information and gated attention mechanism to recognize the heterographic pun.

(2) For Y-Pron-N-Sp and N-Pron-Y-Sp model, they both outperform N-Pron-N-Sp and underperform Y-Pro-Y-Sp. It proves the effectiveness and necessity of the pronunciation part, spelling part, and the gated attention mechanism.

(3) Among all the models, Y-Pron-Y-Sp (PSUGA) beats the three baselines and achieves the best performance. The reason is that our model considers the hierarchical attention mechanism with CMU pronounce embedding, different level of embeddings with spelling information, and effective gated attention mechanism.

*4.3.2 Impact of Feature Combinations* . In this section, we exploit the effects of feature combinations. The experiments are conducted in the same setting. Results are show in table 4:

According to the results, we find that: (1) Phoneme embedding provides complement improvement for understanding the pronunciation information. (2) Character embedding is effective for understanding spelling. It proves that "Tome Swify", "-ed" these structures can discover the heterogarphic pun. (3) Tag embedding can improve the performance because the part of speech information is useful for this task. (4) While the effectiveness of position embedding is low. An intuition explanation is that when using Bi-LSTM to represent sentence, position information might have been encoded by some extent to the distributed representations for reducing the effect.

**Table 4: Performance on Features Combination**

| Models | Features | F1(%) |
|---|---|---|
| PUN | Word Embedding | 81.62 |
| | Phoneme Embedding | 83.53 |
| SUN | Word Embedding | 83.10 |
| | +Character Embedding | 83.82 |
| | + Tag Embedding | 83.56 |
| | +Position Embedding | 83.37 |
| | +Contextual Embedding | 84.09 |
| | All | 84.58 |

*4.3.3 Impact of Different Combination Strategies* . We come up with the single model and combination model to exploit the effects of different attention strategies. Single model is the basic model to introduce the Pronunciation Understanding network (PUN) and Spelling Understanding network (SUN). Combination model is to assemble PUN and SUN with different strategies. Highway network which proposed by Srivastava et al. (2015) is an effective combination network. ST1 and ST2 are two attention strategies have been described in Section 3.2. Results are shown in table 5.

We find that: (1) all the combination models best the single model PUN and SUN, which justifies the intuition of combination strategy to recognize heterographic pun. (2) Among the combination models, both ST1 and ST2 achieve better results than concatenate and highway. It shows that the attention combination strategy can improve the performance for this task. (3) Besides, ST2 beats ST1 by a large margin (0.89% on F1), showing the advantages of using gated attention strategy to assemble information.

**Table 5: Performance on Different Combination Strategies**

| Models | | F1(%) |
|---|---|---|
| Single | PUN | 83.53 |
| | SUN | 84.58 |
| Combination | PSUGA with concatenate | 85.09 |
| | PSUGA with highway | 85.42 |
| | PSUGA with ST1 | 85.57 |
| | PSUGA with ST2 | 86.46 |

*4.3.4 Attention Visualization* . Attention weights of two examples are visualized in Figure 2.

In Exp.1, "dentist", "hates" and "orifice" are of higher attention weights, which implies a pun related to part of speech and position. The weight of gated attention strategy is high for spelling understanding, which shows "orifice" and "office" has much similar spelling. In Exp.2, "grape" and "whine" are the words with much attraction in this sentence. The attention weight of pronunciation understanding is high, which implies pronunciation information can capture the semantic learning between "whine" and "wine".

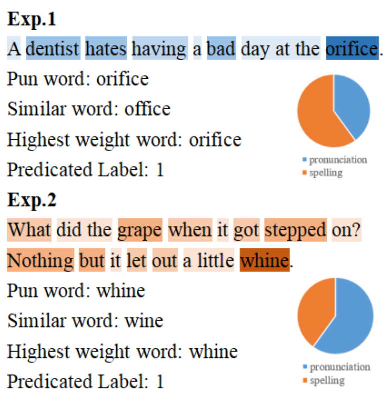
**Exp.1**
A dentist hates having a bad day at the orifice.
Pun word: orifice
Similar word: office
Highest weight word: orifice
Predicated Label: 1
pronunciation
spelling
**Exp.2**
What did the grape when it got stepped on?
Nothing but it let out a little whine.
Pun word: whine
Similar word: wine
Highest weight word: whine
Predicated Label: 1
pronunciation
spelling

**Figure 2: Attention visualization. The word is more important if the color is darker. The pie chart indicates the average value of the gated attention mechanism.**

*4.3.5 Error Analysis* . We also analyze the sentence where our model failed to predict the correct labels of heterographic pun.

Exp.3 When the church bought gas for their annual barbecue, proceeds went from the sacred to the propane.

For example, the true pun label is "negative", but our model predicted its label as "positive". For predicting the label, ambiguity is the main reasons affected the performance. It needs deep semantic analysis of the sentence to identify the ambiguity out. Meanwhile, some background knowledge would be required to predict the label correctly.

## 5 Conclusion and Future Work

In conclusion, we propose an automatic computational PSUGA model combined with Pronunciation-based hierarchical attention mechanism, Spelling-based different levels embeddings and gated attention strategy to settle with confused and ambiguous meaning. Experimental results show that our method achieves significant improvement over exiting methods and sets a new state-of-the-art performance.

In future work, we would like to find an appropriate way in incorporation the external background knowledge for recognizing the heterographic pun. We also focus on deep semantic analysis, interpretation and generation with heterographic pun. Those are all promising jobs we can pursue in the future.

## 6. ACKNOWLEDGMENTS

This work is partially supported by grant from the Natural Science Foundation of China (No.61632011, 61572102, 61772103, 61702080, 61602079), the Ministry of Education Humanities and Social Science Project (No.16YJCZH12), the Fundamental Research Funds for the Central Universities (DUT18ZD102), the National Key Research Development Program of China (No. 2016YFB1001103), and China Postdoctoral Science Foundation (No. 2018M631788).

## REFERENCES

Tristan M and Mladen T. Towards the automatic detection and identification of English puns[J]. European Journal of Humour Research, 2016 ,4(1):59–75.
Tanaka K. The pun in advertising: A pragmatic approach[J]. Lingua, 1992, 87(1-2): 91-102.
Miller T, Gurevych I. Automatic disambiguation of English puns[C]//ACL. 2015: 719-729.
Yang D, Lavie A, Dyer C, et al. Humor Recognition and Humor Anchor Extraction[C]// EMNLP. 2015: 2367-2376.
Hempelmann CF. Computational humor: Beyond the pun?[J]. The Primer of Humor Research. Humor Research, 2008, (8): 333-360.
Zwicky A, Zwicky E. Imperfect puns, markedness, and phonological similarity: With fronds like these, who needs anemones[J]. Folia Linguistica, 1986, 20(3-4): 493-503.
Robinson, T. The British English example pronounciation dictionary [M], 1996.
Kim Binsted. Machine humour: An implemented model of puns. PH.D. University of Edinburgh, 1996.
Kao J T, Levy R, Goodman N D. A Computational Model of Linguistic Humor in Puns[J]. Cognitive science, 2015.
Jaech A, Koncel-Kedziorski R, Ostendorf M. Phonological Pun-derstanding[C]. Proceedings of NAACL-HLT. 2016: 654-663.
Sevgili Ö, Ghotbi N, Tekir S. N-Hance at SemEval-2017 Task 7: A Computational Approach using Word Association for Puns[C]//Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017: 436-439.
Doogan S, Ghosh A, Chen H, et al. Idiom Savant at Semeval-2017 Task 7: Detection and Interpretation of English Puns[C]//Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017: 103-108.
Pramanick A, Das D. JU_CSE_NLP at SemEval 2017 Task 7: Employing Rules to Detect and Interpret English Puns[C]//Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017: 432-435.
Xiu Y, Lan M, Wu Y. ECNU at SemEval-2017 Task 7: Using Supervised and Unsupervised Methods to Detect and Locate English Puns[C]//Proceedings of

the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017: 453-456.

Hempelmann C F. Paronomasic puns: Target recoverability towards automatic generation[J]. Dissertation Abstracts International, 2003 ,64(11):4029.

Ritchie. Computational mechanisms for pun generation[J]. Proceedings of the 10th European Workshop on Natural Language Generation, 2005: 8-10.

Hong B A, Ong E. Automatically extracting word relationships as templates for pun generation[C]. The Workshop on Computational Approaches To Linguistic Creativity. Association for Computational Linguistics, 2010:24-31.

Julia M. Taylor and Lawrence J. Mazlack. Computationally recognizing wordplay in jokes. In Proceedings of the 26th Annual Conference of the Cognitive Science Society[C] (CogSci 2004), 2004:1315–1320.

Julia M Taylor. Computational detection of humor: A dream or a nightmare? the ontological semantics approach[C]. In Proceedings of the 2009 ACM International Joint Conference onWeb Intelligence and Intelligent Agent Technology, 2009, (03): 429–432.

Dario Bertero and Pascale Fung. A Long Short-Term Memory Framework for Predicting Humor in Dialogues[J]// Proceedings of NAACL-HLT 2016: 130-135.

Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer Science, 2014.

Yang Z, Yang D, Dyer C, et al. Hierarchical Attention Networks for Document Classification[C]//HLT-NAACL. 2016: 1480-1489.

Tan Z, Wang M, Xie J, et al. Deep Semantic Role Labeling with Self-Attention[C]. AAAI. 2018.

Shen T, Zhou T, Long G, et al. Disan: Directional self-attention network for rnn/cnn-free language understanding[C]. AAAI, 2018.

Du J, Xu R, He Y, et al. Stance Classification with Target-specific Neural Attention[C]// Twenty-Sixth International Joint Conference on Artificial Intelligence. 2017:3988-3994.

Seo M, Kembhavi A, Farhadi A, et al. Bidirectional attention flow for machine comprehension[J]. ICLR, 2017.

Yu A W, Dohan D, Luong M T, et al. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension[J]. ICLR, 2018.

Gui L, Hu J, He Y, et al. A question answering approach to emotion cause extraction[J]. arXiv preprint arXiv:1708.05482, 2017.

Lu J, Yang J, Batra D, et al. Hierarchical question-image co-attention for visual question answering[C]//Advances In Neural Information Processing Systems. 2016: 289-297.

Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification[C]//Advances in neural information processing systems. 2015: 649-657.

Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.

Pennington J, Socher R, Manning C. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.

Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

Graves A, Jaitly N, Mohamed A. Hybrid speech recognition with deep bidirectional LSTM[C]//Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013: 273-278.

Srivastava R K, Greff K, Schmidhuber J. Highway networks[J]. arXiv preprint arXiv:1505.00387, 2015.

Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

Xu C, Paris C, Nepal S, et al. Cross-Target Stance Classification with Self-Attention Networks[J]. arXiv preprint arXiv:1805.06593, 2018.

Zhang H, Goodfellow I, Metaxas D, et al. Self-Attention Generative Adversarial Networks[J]. arXiv preprint arXiv:1805.08318, 2018.

Shen T, Zhou T, Long G, et al. Bi-directional block self-attention for fast and memory-efficient sequence modeling[J]. arXiv preprint arXiv:1804.00857, 2018.

Kim J, Kong D, Lee J H. Self-Attention-Based Message-Relevant Response Generation for Neural Conversation Model[J]. arXiv preprint arXiv:1805.08983, 2018.

Fan A, Lewis M, Dauphin Y. Hierarchical Neural Story Generation[J]. arXiv preprint arXiv:1805.04833, 2018.

Miller T, Hempelmann C F, Gurevych I. SemEval-2017 Task 7: Detection and interpretation of English puns[C]//Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). Vancouver, BC. 2017.

Pennington J, Socher R, Manning C. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.