Context-Based Sarcasm Detection Model in Chinese Social Media Using BERT and Bi-GRU Models

Chenghao Jia[1*] and Hongying Zan[1]

[1] *College of Computer Science and Artificial Intelligence, Zhengzhou University, Zhengzhou, China*
[*] *ch_jia@stu.zzu.edu.cn*

Abstract

The emergence of web text has brought new challenges to the research of sentiment analysis. At present, most sarcasm detection tasks are not suitable for situational sarcasm which needs to identify contextual information, and the semantics of Chinese sarcasm is more complex than that of English sarcasm. Therefore, this paper proposes a Chinese social media sarcasm detection model combining BERT and Bi-GRU to solve these problems in the background of the increasing use of sarcasm on the Internet. First, we use the bi-directional encoder representations from transformers (BERT) to obtain the word vectors fused with the context. Then, we use bi-directional gate recurrent unit (Bi-GRU) to extract the semantic features and contextual information again. Finally, we get the sarcasm probability of the online comments by the sigmoid function. In this model, the topic background is used as a part of the contextual information to detect sarcasm, so as to improve the accuracy of situational sarcasm detection. At the same time, by extracting semantic features twice, the problem of more complicated Chinese sarcasm semantics is solved.

Keywords

Sarcasm Detection, Bi-GRU, BERT, Social Media

1. Introduction

Sentiment analysis is a significant part of natural language processing (NLP), and enterprises and governments increasingly need sentiment analysis to provide reference for public opinion [1]. For the government, accurately analyzing the public sentiment from the user's language is of great value for the government to serve the people and control public opinion. For enterprises, knowing customers' preferences will help them improve their products and services. Moreover, if the computer can accurately understand the sentiment of the text, it can ease users' emotional pressure and it has medical value. However, with the vigorous development of social media, many Web clients use sarcasm to express their views and emotions on the Internet. For example, Web clients often use “gou tou bao ming”, “[doge]” for joking but not its superficial meaning. Sarcasm is defined as a rhetorical device whose purpose is to belittle someone, or to say something in the opposite sense. In the experiment of Edison et al. [2], the types of irony in the dataset are divided into three categories: verbal irony, situational irony, and other types of verbal irony. Verbal irony refers to words that express the opposite of literal meaning, while situational irony refers to words that are judged by context.

Traditional sentiment analysis tasks can not correctly identify the real sentiment of sarcasm, which will seriously affect the accuracy of sentiment analysis. Therefore, how to accurately identify sarcasm is a difficult problem in the field of sentiment analysis. Accurate sarcasm detection can not only greatly improve the accuracy of enterprises and governments in judging online public opinion trends, but also help computers understand the sentiment of the language at a deeper level, laying the foundation for a more accurate and friendly human-computer interaction environment. Joshi et al. summarized two trends in the direction of sarcasm detection: pattern discovery and role of context [3]. Finding sarcastic patterns was an early trend in sarcasm detection. Pattern-based approaches attempt to identify sarcasm through linguistic and pattern features. In these tasks, the traditional machine learning algorithms are used extensively. For example, Kreuz et al. used support vector machine (SVM) to detect sarcasm [4]. Bamman and Smith used binary logistic regression [5]. Using context is a lately trend in sarcasm detection. The term context here refers to any information beyond common sense, as well as information beyond the text. Wallace et al. highlighted the need for context for sarcasm detection [6]. Silvio Amir

et al. proposed a new method based on convolutional networks in 2016, while allowing the model to learn user-specific context [7]. The result showed that the performance is improved by 2%. Joshi et al. summarized three types of contexts: author-specific context, conversation context, and topical context [3]. Because the topic context of each comment can be easily obtained on social media such as Weibo and Twitter, a few topics are probably going to summon sarcasm more normally than others. This means that sarcasm detection can be easier and more straightforward through topic context in practice. Based on the above reasons, this paper mainly considers the topic context.

Raffel et al. verified the generally held opinion that utilizing a denoising objective commonly results in better downstream task performance compared to a language modeling objective [8]. Edison et al. proposed that using more pre-training schemes can make the model that rely solely on pre-trained embedding performs better [2]. Their work both demonstrated that pre-training models can improve performance. Compared with other pre-training models, the bi-directional encoder representations from transformers (BERT) model has a stronger ability to fuse text. Therefore, this paper proposes a sarcasm detection model combining BERT and bi-directional gate recurrent unit (Bi-GRU). First, the comment text and topic background are converted into dynamic vectors fused with semantics through the BERT pre-training model. Then, the output of BERT is extracted from the semantic features and context information again through the Bi-GRU. Finally, the sarcasm probability can be judged by extracting deep semantics.

The rest of this paper is listed as follows: The second section lists previous works. The third part introduces how to realize the Chinese sarcasm detection model combining BERT and Bi-GRU. At last, the fourth section draws a conclusion.

2. Related Works

2.1. Previous work on sarcasm detection

In previous work on sarcasm detection, sarcasm detection methods can be divided into: rule-based methods, statistical methods, and deep learning-based methods.

Rule-based methods attempt to identify sarcasm through specific evidence. For example, Bharti et al. proposed two rules-based classifiers: a sarcasm detection model based on the occurrence of the interjection word and Parsing-based lexicon generation algorithm (PBLGA) [9]. Rule-based methods need to construct a large number of dictionaries and rule sets, which requires people to spend a lot of time collecting punctuation marks, keywords, central words, demonstrative words and other features. This method puts the recognized object into the corresponding dictionary for matching, which requires multiple corrections and matching. This method is also very dependent on rules and specific languages, fields and styles. It is difficult to cover all types of sarcasm and is only suitable for small-scale data. Moreover, when encountering new problems, it is necessary to construct a new dictionary and set new rules .

Statistical methods, which hope to detect sarcasm through the characteristics of statements, are mainly achieved by using machine learning algorithms. Riloff et al. considered hyperbole, ellipsis and imbalance in their set of features [10]. Reyes et al. used naive bayes and decision trees to detect verbal irony in short web text [11]. However, statistical methods still rely too much on manual annotation and domain knowledge to construct normalized texts. It makes mobility poor and consumes a lot of manpower and material resources.

As architectures based on deep learning technologies become more popular, deep learning-based methods are gradually being applied to the task of sarcasm detection. For example, Ghosh and Veale compared their deep learning architectures with recursive support vector machines, and the result showed that the deep learning architectures bring improvements [12]. Edison et al. used training methods that combine multi-layer bi-directional long short term memory (Bi-LSTM) and pre-trained word embedding [2].

Above works detect sarcasm through pattern discovery. Even though Yi et al. used deep learning techniques [13], they essentially detect sarcasm by linguistic features. In fact, sarcasm detection can not solely depend on the characteristics of language, but on the context. Sarcasm that is only recognized based on linguistic features may overlook contextual features, which will lead to a decrease in accuracy.

Many modern researchers also pay attention to context and use deep learning technology to complete the detection of contextual sarcasm. Hazarika et al. studied the context including user information and topic information [14], and user information includes the user's style and the user's behavior. Kolchinski et al. studied the role of author-specific context in sarcasm detection [15].

2.2. Previous work on pre-training model

Natural language pre-training models are divided into static models and dynamic models. Static models include word2vec [16], glove and so on. Dynamic models include embedding from language models (ELMo) [17], generative pre-training (GPT), BERT and so on. Although the field of NLP has been greatly developed due to static pre-training technology, the static word vector can not characterize poly semantic words well. For example, the word "apple" can be understood as fruit or Apple Company due to the different contexts, and it may also represent the title of a book or the name of a film. If we use static word embedding technology, computer will use one vector to represent these three meanings. In 2018, the emergence of ELMo effectively solved the problem of poly semantic words. Then, new pre-training models such as GPT and BERT are proposed one after another, especially the BERT model, which provides a powerful pre-training tool for many tasks in the field of NLP. This is an important breakthrough in the field of NLP. The network structure of ELMo pre-training model uses a two-layer Bi-LSTM that can adjust word embeddings based on current contextual information dynamically, but its feature fusion capability is weaker than BERT. Although GPT adopts the Transformer model with strong feature extraction ability, it only pays attention to the above information, but abandons the latter information, which makes it impossible to fully combine the semantic information of the context. BERT makes up for the problems of ELMo and GPT. It uses word-level vectors to extract semantics from the context, and there is no problem of poly semantic words.

3. Proposed algorithm

The sarcasm detection model combining with BERT and Bi-GRU is shown in Figure 1. The model has 5 layers: the text preprocessing layer, the word embedding layer, the BERT layer, the Bi-GRU layer and the decision layer. The text preprocessing layer removes the part of the text that has nothing to do with semantics, and only retains the text containing semantic information by segmenting, cleaning, and standardizing the data in the text dataset. Then, we use the pre-training model to get the static vectors of the corresponding words. The BERT layer further trains and learns the static vectors corresponding to the topic background and the static vectors corresponding to the comment text, and obtains the dynamic vectors fused with their respective context. The two groups of dynamic vectors output by the BERT layer are linearly connected as the input of the Bi-GRU layer. The Bi-GRU layer extracts the context information of comments that fuse their topic background. The final states of the two directions of the Bi-GRU are concatenated with each other and pass through a fully connected linear layer or multiple fully connected linear layers. The output of the last linear layer is input through a sigmoid function, which yields the estimated probability of sarcasm.

3.1. The text preprocessing layer and the word embedding layer

There are a lot of noise data in the original comments and topic background text, such as expressions, a series of punctuation marks, etc. First, we use regular expressions to clear the text. We treat the whole review text as a sentence, then divide the sentence by character and delete the stop words. The same is true of the theme background text. The marked sentences only contains the characters with semantic information. We make sure that the maximum length of the sentence does not exceed the set sentence length minus 2, because the remaining 2 positions are used to store the [CLS] flag and the [SEP] flag. After preprocessing, each character of the sentence will be mapped to the vector one by one. Finally, we get the static vectors. These word vectors are used as one of the inputs of the BERT pre-training model. After the BERT layer, the dynamic word vectors fused with the context will be obtained.

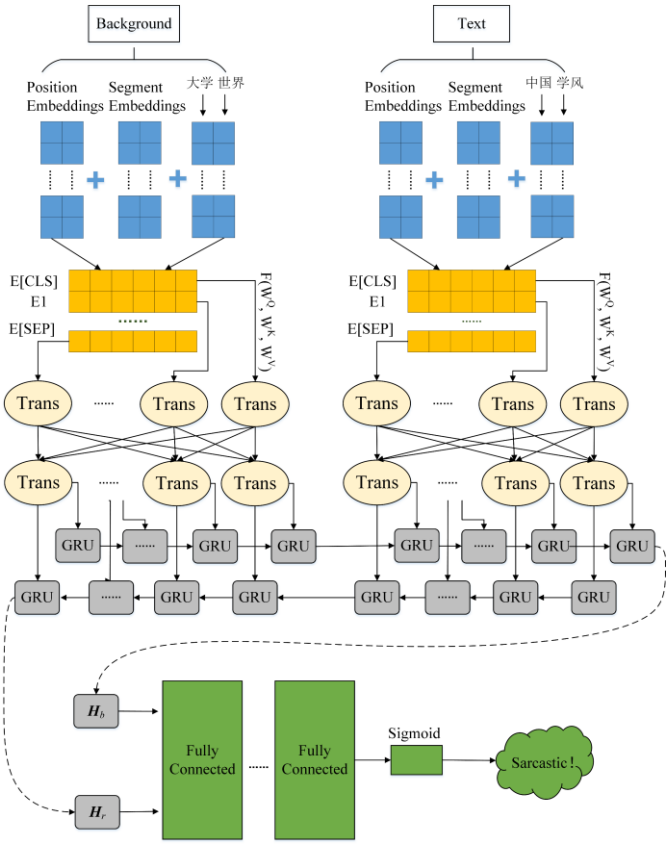

Figure 1: Model combined with BERT and Bi-GRU

3.2. The BERT layer

The BERT [18] model adopts the transformer model with strong ability to integrate text, and considers the importance of each word to other words by combining the attention mechanism. The vectors pre-trained by the BERT model work better. The network structure of BERT model shown in Figure 2. $e_1, \ldots, e_n$ are the input vectors of the BERT model. $T_1, \ldots, T_n$ are the output vectors of the BERT model.

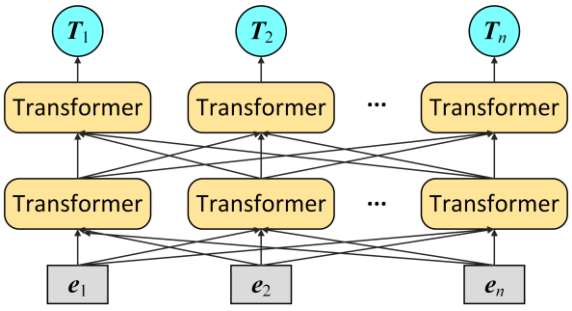

Figure 2: Network structure of BERT model

The formation of the input vector e_i is shown in Figure 3, which is formed by adding up the corresponding elements of three different vectors. The first vector of each sentence is the [CLS] flag, which can be used for downstream classification tasks. The [SEP] flag is used as a separator for different sentences. We treat the comment text as a sentence, so we only use a sentence vector. The topic background is the same as the comment text, which is also treated as a sentence. P_E is the position vector that records the position of the participle in the sentence, and the calculation formula is shown:

$$P_E(pos, 2i) = \sin\left(pos/10000^{2i/d}\right) \quad (1)$$

$$P_E(pos, 2i+1) = \cos\left(pos/10000^{2i/d}\right) \quad (2)$$

Where *pos* represents the position of the word segmentation in the sentence, $2i$ and $2i + 1$ represent the even and odd dimensions of the word vector respectively and d represents the dimension of the input vectors, which is also the dimension of the output vectors.

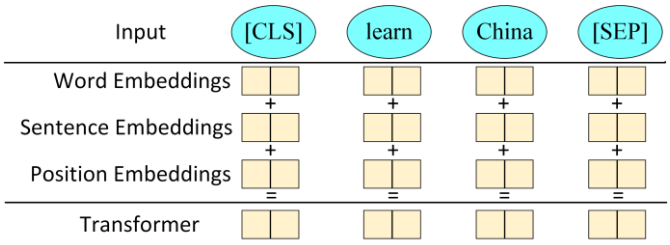

Figure 3: Input structure of BERT model

The transformer structure of the BERT model is shown in Figure 4. The transformer model includes encoder and decoder. The encoder is composed of a stack of N = 6 identical layers. The decoder is also composed of a stack of N = 6 identical layers.

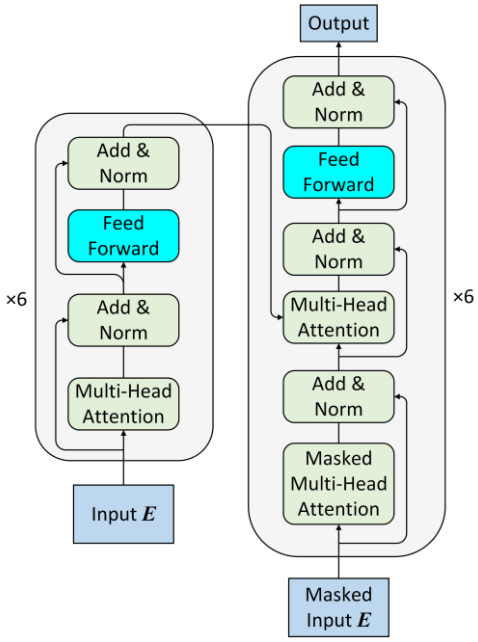

Figure 4: Structure of Transformer model

BERT is based on matrix calculations in practice. We need to concatenate all the inputs into a vector matrix $E = \{e_1, \ldots, e_n\}$. The multi-head attention mechanism consists of 8 self-attention mechanisms. The input of self-attention is three different vector matrices: Query matrix (Q), Key matrix (K) and Value matrix (V). They are calculated by multiplying vector matrix E and matrix W^Q, W^K and W^V respectively. The calculation formula for calculating self-attention mechanism is shown:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \tag{3}$$

Where $Q \in R^{n \times d_K}$, $K \in R^{n \times d_K}$, $V \in R^{n \times d_V}$ and d_K is vector dimension. The Softmax function normalizes each row vector after calculating $QK^T / \sqrt{d_K}$, in order to calculate the weight of a word to other words.

The output X of the multi-head attention mechanism can be calculated by formula (4) and formula (5):

$$X = MultiHead(Q, K, V) = Concat\left(Attention_1, \ldots, Attention_i, \ldots, Attention_g\right) \cdot W^O \tag{4}$$

$$Attention_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{5}$$

Where W^O represents the parameter matrix, which is the parameter matrix to be learned during training, and W_i^Q, W_i^K and W_i^V represent W^Q, W^K and W^V in the i-head attention mechanism respectively.

Z is obtained from the output X of multi-head attention after residual and layer normalization operations. Z is the input of the fully connected feedforward network (FFNN). FFNN consists of two fully connected layers. The calculation formula is shown:

$$FFNN(Z) = \max\left(0, ZXW_1^F + b_1\right) \cdot W_2^F + b_2 \tag{6}$$

Where W_1^F and W_2^F represent the parameter matrix of the FFNN layer, and b_1 and b_2 are bias vectors. They are the parameters that need to be learned during training.

After the output of the fully connected feedforward neural network is subjected to residual and layer normalization operations, the result is the input of the next encoder. The input of the first encoder is the sentence word vector matrix, the input of the subsequent encoder is the output of the previous encoder, and the last encoder output is the encoded matrix, which will act on each decoder. The calculation process of decoder is similar to that of encoder, but a layer of masked multi-head attention mechanism is added and the output masked matrix is used as one of the inputs of the next sub-layer. Denote the output matrix of the last layer of BERT as $T_r=\{T_1, \ldots, T_n\}$, the row and column dimensions of the T_r matrix are the same as the BERT input matrix, and each row vector represents the unambiguous depth vector of the word segmentation, which is used as the input of the downstream task.

3.3. The Bi-GRU layer and the decision layer

Gated recurrent unit (GRU) [19] is a variant of long-term and short-term memory (LSTM). GRU and LSTM belong to the advanced model of recurrent neural network (RNN). Due to the serious disappearing gradient problem in RNN processing sequences, the perception ability of the later nodes is lower than that of the earlier nodes. To tackle the issue of vanishing gradient, Hinton et al. [20] proposed the LSTM model. As a variant of LSTM, GRU is also very suitable for sequence data processing. It also uses the "gate mechanism" to memorize the information of the previous nodes, thus solving the problem of disappearing gradient. The GRU model is a simplified version of the LSTM model. The "gate" structure used by GRU is different from that of LSTM. GRU merges the input gate and the forget gate into an update gate, which only contains two gate structures: the reset gate and the update gate. And linear self-update is not built on additional memory states, but linearly accumulated on hidden states and regulated by the gate structure. The reset gate determines how to combine the previous information and the current input. The update gate determines how much previous information is retained. Compared with LSTM, GRU has fewer parameters and reduces the risk of overfitting. Because most social media texts are short texts, the GRU model is more suitable than the LSTM model. The GRU network model is shown in Figure 5.

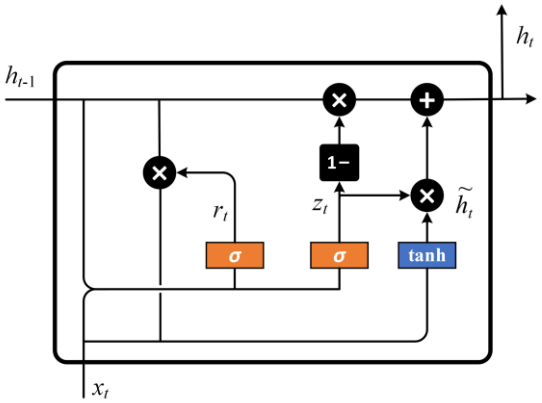

Figure 5: Network structure of GRU model

Where x is the input data, h is the output of the GRU unit, r is the reset gate, and z is the update gate. r and z together complete the calculation from the previous hidden state (h_{t-1}) to the new hidden state (h_t). The update gate simultaneously controls the current input data x_t and the previous memory information h_{t-1}. Finally, it outputs z_t that is a number between 0 and 1. The calculation formula is shown:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{7}$$

Where z_t determines the transfer degree from h_{t-1} to the next state, 0 represents complete abandonment and 1 represents complete retention. In formula (7), σ is the sigmoid function, W_z is the weight of update gate, b_z is the offset.

The reset gate controls the importance of h_{t-1} to the result $\widetilde{h}_t$. If the previous memory h_{t-1} is completely irrelevant to the new memory, the reset gate will remove the influence of the previous memory. The calculation formula is shown:

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{8}$$

Generate new memory information $\widetilde{h}_t$ according to the update door. The calculation formula is shown:

$$\widetilde{h}_t = \tanh(W_r[h_{t-1}, x_t] + b_r) \tag{9}$$

The output at the current time is h_t. The calculation formula is shown:

$$h_t = (1 - z_t)h_{t-1} + z_t\widetilde{h}_t \tag{10}$$

GRU only gets the above information, ignoring the following information. However, the semantic information of sentences is related to the following information as well as the above information. Therefore, this paper uses the Bi-directional GRU neural network, which can simultaneously obtain the above information and the following information and heighten the accuracy of feature extraction. Moreover, Bi-GRU has the benefits of small dependence on word vectors, low complexity and fast response time. The Bi-GRU [21] model is composed of two GRU networks superimposed. There are two GRU units in the opposite directions in each time step. Each word vector of the BERT layer output matrix Tr is input to the positive and negative Grus of each time step, respectively. We input the BERT vectors of the comment text together with the BERT vectors of the topic background to the Bi-GRU layer, so that the output in each direction takes the topic background into account.

The final states of the two directions of the Bi-GRU are concatenated with each other and pass through a fully connected linear layer or multiple fully connected linear layers. The output of the last linear layer is input through a sigmoid function, which yields the estimated probability of sarcasm.

3.4. Implementation Issues

Firstly, the text is preprocessed by the text preprocessing layer and the word embedding layer, and the noiseless word segmentation vector is obtained. Second, we respectively input the static vector of the comment text and the static vector of the topic background to the BERT layer to obtain the dynamic vector of their own text sentiment. Next, we input the dynamic vectors of the comment text together with the dynamic vectors of the topic background to the Bi-GRU layer. Then, the output vector of the topic context is obtained. Finally, we get the estimated probability of sarcasm through the decision layer.

4. Conclusion

The sarcasm detection model proposed in this paper combines BERT and Bi-GRU, and uses the BERT model to obtain the dynamic word vectors of the comment text and topic background respectively, which enables the model to better understand the text semantics. Then, the model uses Bi-GRU to get the result that is fused with topic context, and extract semantic features again to strengthen the model's understanding of semantics. Through the analysis in the third section, the effectiveness of the method is verified. In future work, we will consider using BERT's optimized model and new pre-training models such as transformer-XL network (XLNet) to replace BERT to improve the accuracy. On the other hand, since there is no authoritative Chinese sarcasm dataset at present, the sarcasm corpus can only be manually annotated by each research unit, and due to subjective factors, the quality of the corpus can not be guaranteed. In the future, an appropriate experimental data set will be an important direction.

5. References

[1] Ravi, K. and V. Ravi, A survey on opinion mining and sentiment analysis: tasks, approaches and applications[J]. Knowledge-based systems, 2015. 89: p. 14-46.

[2] Marrese-Taylor, E., et al., IIIDYT at SemEval-2018 Task 3: Irony detection in English tweets[J]. arXiv preprint arXiv:1804.08094, 2018.

[3] Joshi, A., P. Bhattacharyya, and M.J. Carman, Automatic sarcasm detection: A survey[J]. ACM Computing Surveys (CSUR), 2017. 50(5): p. 1-22.

[4] Kreuz, R. and G. Caucci. Lexical influences on the perception of sarcasm[C]. in Proceedings of the Workshop on computational approaches to Figurative Language. 2007.

[5] Bamman, D. and N.A. Smith. Contextualized sarcasm detection on twitter[C]. in Ninth international AAAI conference on web and social media. 2015.

[6] Wallace, B.C., L. Kertz, and E. Charniak. Humans require context to infer ironic intent (so computers probably do, too)[C]. in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2014.

[7] Amir, S., et al., Modelling context with user embeddings for sarcasm detection in social media[J]. arXiv preprint arXiv:1607.00976, 2016.

[8] Raffel, C., et al., Exploring the limits of transfer learning with a unified text-to-text transformer[J]. arXiv preprint arXiv:1910.10683, 2019.

[9] Bharti, S.K., K.S. Babu, and S.K. Jena. Parsing-based sarcasm sentiment recognition in twitter data[C]. in 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). 2015. IEEE.

[10] Riloff, E., et al. Sarcasm as contrast between a positive sentiment and negative situation[C]. in Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

[11] Reyes, A., P. Rosso, and T. Veale, A multidimensional approach for detecting irony in twitter[J]. Language resources and evaluation, 2013. 47(1): p. 239-268.

[12] Muresan, S., et al., Identification of nonliteral language in social media: A case study on sarcasm[J]. Journal of the Association for Information Science and Technology, 2016. 67(11): p. 2725-2737.

[13] Tay, Y., et al., Reasoning with sarcasm by reading in-between[J]. arXiv preprint arXiv:1805.02856, 2018.

[14] Hazarika, D., et al., Cascade: Contextual sarcasm detection in online discussion forums[J]. arXiv preprint arXiv:1805.06413, 2018.

[15] Kolchinski, Y.A. and C. Potts, Representing social media users for sarcasm detection[J]. arXiv preprint arXiv:1808.08470, 2018.

[16] Mikolov, T., et al., Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[17] Peters, M., et al., Deep Contextualized Word Representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[18] Devlin, J., et al., Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[19] Chung, J., et al., Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

[20] Hinton, G.E. Learning distributed representations of concepts[C]. in Proceedings of the eighth annual conference of the cognitive science society. 1986. Amherst, MA.

[21] Cho, K., et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.