Sarcasm Detection with Self-matching Networks and Low-rank Bilinear Pooling

Tao Xiong*
Ant Financial Security and Risk Management
Hangzhou, China
weilue.xt@alibaba-inc.com

Peiran Zhang†
Alibaba Group
Hangzhou, China
peiran.zpr@alibaba-inc.com

Hongbo Zhu
Alibaba Group
Hangzhou, China
xiaofeng.zhb@alibaba-inc.com

Yihui Yang*
Afterpay
Melbourne, Australia
yihui.y@outlook.com

ABSTRACT

Sarcasm is sophisticated linguistic expression and is commonly observed on social media and e-commerce platforms. Failure to detect sarcastic expressions in natural language processing tasks, such as opinion mining and sentiment analysis, leads to poor model performance. Traditional approaches rely heavily on discrete handcrafted features and will incur enormous human costs. It was not until recent that scholars began to employ neural networks to address these limitations and have achieved new state-of-the-art performance. In this work, we propose a novel self-matching network to capture sentence *"incongruity"* information by exploring word-to-word interactions. In particular, we calculate the joint information in each word-to-word pair in the input sentence to build a self-matching attention vector, based on which we attend the sentence and build its representation vector. Such a network allows sentence to match within itself word by word and cater to the words of conflict sentiments. In addition, we incorporate a bi-directional LSTM network into our proposed network to retain compositional information. We concatenate incongruity information and compositional information through a Low-rank Bilinear Pooling method to control for potential information redundancy without losing discriminative power. Experiment results on publicly available datasets demonstrate that our model significantly outperforms extant baselines on standard evaluation metrics including precision, recall, $F1$ score and accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**.

KEYWORDS

Sarcasm detection, self-matching, Bi-LSTM, co-attention

ACM Reference Format:
Tao Xiong, Peiran Zhang, Hongbo Zhu, and Yihui Yang. 2019. Sarcasm Detection with Self-matching Networks and Low-rank Bilinear Pooling. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3308558.3313735

*This work was finished when the first and the fourth authors were was working at Alibaba Group.
†Corresponding author.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313735

1 INTRODUCTION

Sarcasm, which refers to *"a sharp and often satirical or ironic utterance designed to cut or give pain"*[1], is not unusual in online communities such as social media and e-commerce platforms. Sarcasm detection is significant for many language understanding tasks, such as opinion mining and sentiment classification [27]. According to Ghosh and Veale, human's ability to detect sarcasm mainly relies on social cues such as sentiment, scenario and people's intentions [10]. However, as a computational task, using learning models to detect such a complicated linguistic expression is challenging. The sophistication of automatic sarcasm detection mainly comes from the difficulty in identifying its proper context dependency, such as humorous scenario [11], or discriminating the contradictory between literal expression and author's underlying intention, such as irony and metaphor [35]. Following are some sarcastic examples collected from Twitter by Riloff et al. [34]:

- *"A wonderful day of starting work at 6am"*.
- *"I love finals week! #justkidding #stressed"*
- *"I am working hard to be this poor"*.
- *"Nothing makes me happier than getting on the highway and seeing break lights light up like a Christmas tree"*.
- *"Oh thank GOD, our entire office email system is down"*.

These examples demonstrate a strong disparity between authors' intended emotions and true contexts. For example, *"starting work at 6am"* and *"finals week"* are less likely to be enjoyed. Joshi et al. defines such a disparity as *"incongruity"*, referring to contradictory implications between words and phrases [16].

Sarcasm is a widely studied topic in cognitive science and linguistics [7, 8, 14]. However, the study of sarcasm in computational linguistics community is still at its nascent stage [10]. Traditional machine learning based sarcasm detection approaches typically consider it as a binary classification task and emphasize the importance of exploring predictive discrete features [42], including both content features (e.g. n-grams, sentiment lexicons, incongruity) [29, 33, 34], and contextual features (e.g. historical behavior, contextual environment) [2, 30]. These approaches incur nonnegligible research efforts on feature engineering and the modeling process is usually inefficient. In recent times, scholars began to explore neural features based on word embeddings [26, 28] and cultivate deep

[1]Defined by Merriam-Webster dictionary at https://www.merriam-webster.com.

networks, such as Convolutional Neural Networks(CNN), Gated Recurrent Units (GRU), Long Short Term Memory (LSTM) networks, and Attention networks to improve modeling efficiency as well as model performance [1, 10, 35, 42].

Current state-of-the-art sarcasm learning models mainly rely on identifying and exploiting the incongruity patterns embedded in sentences [34, 35]. For example, Riloff et al. address a common incongruity pattern that a sentence contains a positive sentiment which is adhere to a negative scenario or vice versa [34]. A few other studies utilize the incongruity information between target sentence and authors' historical behaviors [1] or contextual information[11]. Approaches utilizing historical information or contextual information rely on extra inputs other than the target sentence itself. In the most recent state-of-the-art model, Tay et al. suggest that the incongruity can be identified by comparing word-to-word interactions and they compute the embedding similarity between every word-to-word pair in the input sentence to search for conflict sentiments [35].

In this work, we aim to predict sarcasm by using sentence information exclusively to save data collecting efforts. We follow the word-to-word comparison approach suggested by Tay et al. [35] to search for incongruity within input sentence. However, Tay et al. compute the joint information between words simply by using the inner product between word embeddings [35]. In our proposed self-matching network, we improve the word-to-word comparison approach by proposing a novel self-matching network based on the "co-attention" mechanism suggested by Lu et al. [23]. The original co-attention mechanism is designed to attend two input feature maps simultaneously, and we employ a similar approach to attend embeddings simultaneously to capture the joint information between every word-to-word pair in the input sentence. Such a comprehensive comparison mechanism allows us to fully capture any potential incongruity information generated by the sentiment conflicts between words. In addition, as Tay et al. suggest, compositional and sequential information of sentence is also important for sarcasm detection and existing state-of-the-art models rely on deep and sequential neural networks such as GRU and LSTM [35]. Therefore, we also incorporate sentence compositional information as prediction input by employing a bi-directional LSTM encoder [13], to complement the self-matching network. Our proposed network generates two sentence representations separately: one captures incongruity information, the other captures compositional information. We do not input them directly into the final prediction layer because of potential information redundancy. Instead, we propose a bilinear pooling method based on Low-rank Bilinear Pooling (LRBP) [21] to reduce input dimension without losing sentence representation's discriminative power.

We evaluate model performance by conducting extensive experiments on several publicly available datasets. Experiment results demonstrate two major findings: first, our proposed self-matching network outperforms most of extant neural network based sarcasm detection models on standard evaluation metrics including precision, recall, $F1$ score and accuracy, and on some datasets, we notice that our model improves $F1$ score and accuracy of existing state-of-the-art performance by over 10%. Second, sequential networks such as LSTM are known for its power on capturing compositional and sequential information, we compare the self-matching network with and without the bi-directional LSTM encoder to evaluate the effectiveness of cultivating compositional information on sarcasm detection task. Experiment results demonstrate that in most circumstances, employing a sequential neural network encoder to complement the self-matching network helps improve model performance.

The rest of this paper is organized as follows: in section 2, we briefly review extant related studies; in section 3 we describe our proposed model architecture in detail, specifically, we illustrate how self-matching network, bi-directional LSTM and the low-rank bilinear pooling method works; in section 4 we demonstrate our experimental settings and discuss experiment results; we conclude this paper in section 5.

2 RELATED WORKS

Sarcasm is a widely studied linguistic phenomenon by cognitive science and linguistic scholars [7, 8, 14], the study of sarcasm in the computational linguistic community is still at the beginning[10]. Extant studies on sarcasm detection can be classified into three main streams: (1) rule based approaches; (2) machine learning approaches based on discrete features; and (3) neural network approaches.

Rule based approaches are the most straightforward methods for sarcasm detection. Researchers attempt to identify sarcastic expressions based on typical linguistic rules and representative indicators. Among these obvious rules and routines, hash-tags attached by authors themselves, such as *#sarcasm*, *#not true*, *#just kidding*, are widely explored by previous studies [9, 25, 37]. Veale and Hao suggest that some unique simile types, such as sentences containing the pattern of "as xx as a xx" (e.g., *"as private as a park-bench"*, *"as useful as a chocolate teapot"*), can be a strong predictor for ironic expressions [38]. In social contexts, sarcasm is usually expressed verbally with heavy tonal stress or certain body gestures such as eye rollings [5], these pieces of behavioral information are unable to be recovered in automatic sarcasm detection though, certain syntactic and semantic text patterns, such as hyperboles identification [5] and interjections and punctuations [8] can still be utilized.

Rule based approaches solely rely on fixed patterns. In order to make improvements, scholars began to exploit a wide range of discrete features including pattern features and semantic features. Pattern features such as n-grams features and occurrence patterns, usually relate to words' syntactic patterns without considering words' implications [3, 16, 17, 22, 31, 33]. For example, Reyes and Rosso use n-grams, which are sequences combining 2 up to 7 words, to search for a set of recurrent words carrying sarcasm information [31]; Barbieri et al. propose an Italian sarcasm detection model based on bag-of-words [3]. Semantic features rely on words' implications and meanings. For example, ambiguity is an important aspect of sarcasm because sarcastic expressions usually present words with ambiguous meanings [4, 32]. Scholar also explore the relationship between contexts and behaviors [19, 30, 40]. Among these various semantic features, incongruity based features are the most widely explored ones, incongruity information is usually identified by searching for negative words in positive context or vice versa [3, 6, 12]. It is worth pointing out that traditional machine learning approaches usually employ various pattern features and semantic

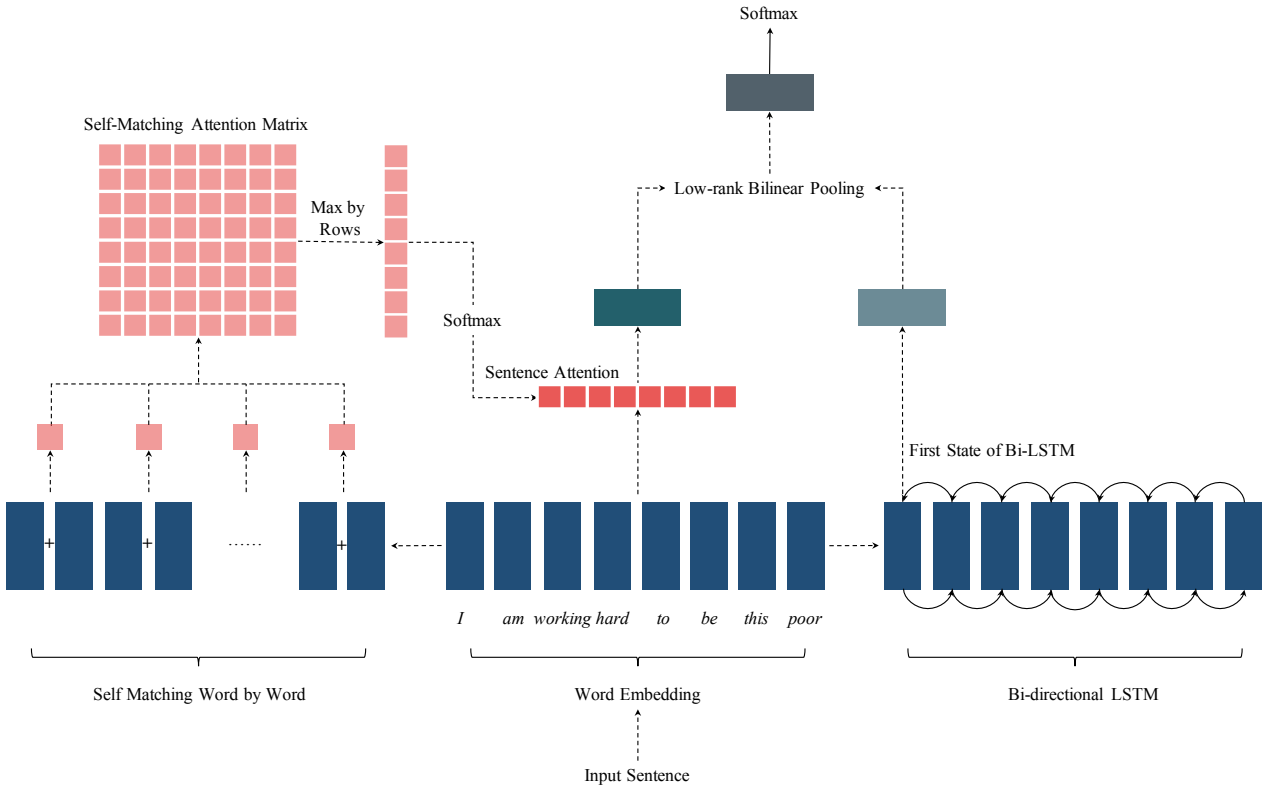

Figure 1: Model architecture. The left part is the self-matching network which is used to generate an attended sentence feature vector based on the self-matching process to search for incongruity information; and the right part is a bi-directional LSTM network used to capture sentence compositional information. We concatenate these two sentence feature vectors by using a low-rank bilinear pooling method to reduce the dimension of the final sentence representation vector.

features simultaneously. Joshi et al. provide a more comprehensive overview on studies based on feature engineering[15] .

Discrete feature based approaches incur nonnegligible research efforts on feature engineering. In recent years, scholars began to employ neural network approaches to detect sarcastic expressions. Joshi et al. are among the very first few studies demonstrating the effectiveness of word embeddings on sarcasm classification tasks [18]. Zhang et al. design a bi-directional Gated Recurrent Neural Network (GRNN) based on neural features, and they also explore how to automatically learn content and context embeddings to make improvements[42]. Amir et al. employ a Convolutional Neural Network (CNN) model to automatically learn *"user embeddings"* from users' historical behavior (e.g. previous tweets), and they detect sarcasm by searching for the incongruity between user embeddings and target content embeddings [1]. Ghosh and Veale combine Deep Neural Networks (DNN) with Long Short-Term Memory(LSTM) to capture sentences' sequential information[10]. In their following study, Ghosh and Veale incorporate user's mood information as well as contextual information into a neural architecture containing two CNN layers and one LSTM layer and achieve improved model accuracy [11] .

Neural network approaches achieve the state-of-the-art performance for sarcasm detection, according to Tay et al., sarcasm detection relies on semantic relationships between words and phrases, yet most sequential neural networks such as CNN, RNN and LSTM, still lack the capability to capture the incongruity and contrast between words, especially when those words are located relatively far from each other. Therefore, they propose a neural network to attend a sentence representation by comparing word-to-word embeddings, and an LSTM encoder to capture sequential information. The so-called *intra-attention* mechanism allows their model to search for conflict sentiments as well as maintain compositional information. This model achieves the new state-of-the-art performance on sarcasm detection [35].

Inspired by the concept of searching for words' contrast and incongruity [29, 33–35], we propose a novel Self-matching Sarcasm Detection model (SMSD) based on a modified "co-attention" mechanism [23]. The major difference between our approach and Tay et al. on capturing word-to-word information is that Tay et al. use inner product operation to capture the joint information between words [35], however, we believe that the inner product operation dilutes information across different words and it only captures words' co-occurrence patterns without considering their sentiment information. As an improvement, we calculate the joint information by introducing a parameter matrix between two word embedding vectors, it helps capture the correlation between words

and is more sensitive on attending sentiment conflicts. We will compare our experiment results with [35] to validate our model performance.

3 MODEL DESCRIPTION

In this section, we describe our proposed model in detail. Model architecture is depicted in Figure 1.

3.1 Sentence Representation

We begin by training a word embedding space $\mathbf{E} \in \mathbb{R}^{V \times k}$, where V is the size of vocabulary and k is the size of dimension for each word vector. For any word in our vocabulary, we assign a vector $\mathbf{e} \in \mathbb{R}^k$ as its feature vector. Feature vectors are associated with rows in the embedding space $\mathbf{E}$. For any sentence s, we build a feature map by combining its word embeddings: $\mathbf{S} = [\mathbf{e}_1^{\mathrm{T}}, \mathbf{e}_2^{\mathrm{T}}, \ldots, \mathbf{e}_n^{\mathrm{T}}]$, $\mathbf{S} \in \mathbb{R}^{k \times n}$, where n is the number of words in the input sentence.

3.2 Self-matching Network

The purpose of our proposed self-matching network is to generate a attended feature vector for the input sentence: $\boldsymbol{f}_a = \mathbf{S} \cdot \boldsymbol{a}$, where $\boldsymbol{a} \in \mathbb{R}^n$ is the self-matched attention vector. As previous studies have found that semantic incongruity is a strong predictor for sarcasm [16], therefore, attention vector $\boldsymbol{a}$ is designed to capture sentence incongruity. In this section, we discuss the specifics of how we calculate the self-matched attention vector.

In the recent state-of-the-art model, Tay et al. propose that the incongruity embedded in a sentence mainly comes from conflict sentiments between words, and they take a deep exploration into this concept by considering the interaction between every word-to-word relationship. In their proposed *"intra-attention"* network, they calculate inner product value for every word-to-word pair as the interaction information. For example, the interaction information $w_{i,j}$ between word i and j is computed as:

$$w_{i,j} = \mathbf{e}_i \cdot \mathbf{e}_j^{\mathrm{T}} \tag{1}$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are word embeddings for i and j [35]. In our self-matching network, we employ a similar approach to search for potential incongruity by comparing word-to-word information. However, using inner product operation to compute joint information absorbs and dilutes too much information across different words. Arithmetically, inner product operation aims to compare the similarity between two feature vectors, it lacks the ability to take sentiment information into consideration. Word embedding aims to capture the occurrence patterns between words, not exclusively cater to the sentiment. For example, in a pre-trained word embedding space, the inner product between the vector for *"Loud"* and the vector for *"Silent"* usually is larger than the inner product between *"Loud"* and *"Noisy"*, yet the inner product between *"Loud"* and *"New"* also is likely to be significant because these two words are less likely to have similar occurrence patterns. Therefore, the simple inner product operation might not be able to fully capture sentiment conflicts between words. In the self-matching network, we introduce a parameter matrix based on "co-attention" approach to address this limitation.

Lu et al. propose a "co-attention" network to address the Visual Question Answering (VQA) task [23]. In particular, they introduce an affinity matrix $\mathbf{C}$ to attend input picture feature map $\mathbf{V}$ and text question representation $\mathbf{Q}$. $\mathbf{C}$ is calculated by:

$$\mathbf{C} = tanh(\mathbf{Q} \cdot \mathbf{W}_a \cdot \mathbf{V}) \tag{2}$$

where $\mathbf{W}_a$ contains attention weights based on which Lu et al. adopt a joint activation approach (e.g. maximize by rows and columns) to adjust attention weights for $\mathbf{V}$ and $\mathbf{Q}$ simultaneously. We modify this approach by introducing a weight matrix between word-to-word pair to improve the flexibility of capturing joint information between words.

Specifically, for word pair $(\mathbf{e}_i, \mathbf{e}_j)$, we compute the joint feature vector $w_{i,j}$ as follows:

$$w_{i,j} = tanh(\mathbf{e}_i \cdot \mathbf{M}_{i,j} \cdot \mathbf{e}_j^{\mathrm{T}}) \tag{3}$$

where $\boldsymbol{w}_{i,j} \in \mathbb{R}$ measuring the joint information between word i and word j, and $\mathbf{M}_{i,j} \in \mathbb{R}^{k \times k}$ is a parameter matrix.

We build a self-matching information matrix $\mathbf{W}$ based on all joint information $w_{i,j}, i, j \in (1, 2, ..., n)$:

$$\mathbf{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{2,2} & \cdots & w_{n,n} \end{pmatrix} \tag{4}$$

After having $\mathbf{W}$, we use a maximization activation approach to calculate the self-matched attention vector $\boldsymbol{a}$. We first calculate an intermediate vector $\boldsymbol{m} \in \mathbb{R}^n$ by maximizing elements in $\mathbf{W}$ by rows:

$$m_i = max(w_{i,1}, w_{i,2}, \ldots, w_{i,n}), \forall i \in (1, 2, \cdots, n) \tag{5}$$

and we input $\boldsymbol{m}$ into a standard softmax function to calculate $\boldsymbol{a}$: $\boldsymbol{a} = Softmax(\boldsymbol{m})$. Softmax function is adopted for the purpose of normalization.

3.3 Bi-direction LSTM

In our experiment we first propose an approach by inputing the attended sentence feature vector generated by the self-matching network $\boldsymbol{f}_a$ into the prediction layer (denoted as SMSD). However, we notice that the self-matching network only aims to capture incongruity information contained in sentence, it lacks the ability to cultivate sentence's compositional information, which as suggested by previous studies, is also critical for sarcasm detection [35]. Therefore, as a complement to the self-matching network, we propose a second approach which employs a standard Bi-directional LSTM encoder (Bi-LSTM)[13] to capture the compositional information for each input sentence (denoted as SMSD-BiLSTM). Bi-LSTM contains a forward LSTM operation ($\overrightarrow{LSTM}$) which reads the clause information from word 1 to word n; and a backward LSTM operation ($\overleftarrow{LSTM}$) which reads the clause information from word n to word i. Hidden states generated from forward LSTM and backward LSTM are concatenated as the hidden states for Bi-LSTM [2].

For each input sentence, we feed its original word embeddings into the Bi-LSTM encoder and we define the hidden state at the ith

[2]LSTM is based on a non-linear parameterized gating mechanism, as it's one of the most popular neural networks in NLP community, we skip the technical detail of LSTM and Bi-LSTM in this paper for brevity.

time step as:

$$\boldsymbol{h}_i = [\overrightarrow{LSTM}(\mathbf{e}_i), \overleftarrow{LSTM}(\mathbf{e}_i)], \forall i \in (1, 2, \cdots, n) \tag{6}$$

where n is the maximum number of words in the input sentence, $\mathbf{e}_i$ is the feature vector for the corresponding word and $\boldsymbol{h}_i \in \mathrm{R}^d$ is the hidden output of the bi-directional LSTM encoder at time-step i. Hyperparameter d is the dimension of the output vector.

Instead of using all hidden states generated from the Bi-LSTM encoder, we follow the approach proposed by Tay et al. and only adopt the first hidden state as the output of the Bi-LSTM encoder [35][3]. We define the sentence feature vector generated by Bi-LSTM encoder as follows:

$$\boldsymbol{f}_l = \boldsymbol{h}_1 \tag{7}$$

where $\boldsymbol{f}_l \in \mathbb{R}^d$, and d is a hyperparameter.

3.4 Low-rank Bilinear Pooling

We have two sentence feature vectors: $\boldsymbol{f}_a$ generated by the self-matching network, and $\boldsymbol{f}_l$ generated by the Bi-LSTM encoder. We can simply concatenate these two feature vectors for the final prediction. Self-matching network aims to capture word-to-word interactions and it is not intended to capture sequential information, but it might still contain compositional information because of the self-matching operation between adjacent words. Therefore, the two feature vectors $\boldsymbol{f}_a$ and $\boldsymbol{f}_l$ are of high possibility to have redundant sentence information and it is unnecessary to input all of them into the final prediction. In this case, we employ a Low-rank Bilinear Pooling (LRBP) method based on Hadamard product to reduce the dimension of the final input vector to control for potential information redundancy without reducing feature vector's discriminative power [21].

Considering the two feature vectors $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{y} \in \mathbb{R}^M$, standard bilinear pooling method uses a quadratic expansion of linear transformation to pool information from input vectors [21] as follows:

$$f_i = \boldsymbol{x}^{\mathrm{T}} \cdot \mathbf{W}_{p,i} \cdot \boldsymbol{y} + g_i \tag{8}$$

where $\mathbf{W}_{p,i} \in \mathbb{R}^{N \times M}$ is a weight matrix for output f_i and g_i is the bias. If L is the number of the output features, the total number of parameters including bias is $L \times (N \times M + 1)$. As an improvement, Kim et al. propose a Low-rank Bilinear Pooling (LRBP) method to reduce the rank of the weight matrix $\mathbf{W}_p$ with less parameters to be learned. Specifically, they decompose the matrix $\mathbf{W}_{p,i}$ as $\mathbf{W}_{p,i} = \mathbf{U}_i \cdot \mathbf{V}_i^{\mathrm{T}}$, where $\mathbf{U}_i \in \mathbb{R}^{N \times d}$ and $\mathbf{V}_i \in \mathbb{R}^{M \times d}$. Based on this concept, they propose the following equation to calculate a output feature vector $\boldsymbol{f} \in \mathbb{R}^c$:

$$\boldsymbol{f} = \mathbf{P}^{\mathrm{T}} \cdot (\mathbf{U}^{\mathrm{T}} \cdot \boldsymbol{x} \circ \mathbf{V}^{\mathrm{T}} \cdot \boldsymbol{y}) + \boldsymbol{g} \tag{9}$$

where $\circ$ represents the Hadamard product, the input of which are two matrices of the same dimension, and it produces another matrix of the same dimension where each element (i, j) in the output matrix is the arithmetic product of elements (i, j) in the original two matrices. $\mathbf{P} \in \mathbb{R}^{d \times c}$, $\mathbf{U} \in \mathbb{R}^{k \times c}$ and $\mathbf{V} \in \mathbb{R}^{d \times c}$ are parameters to be learned and $\boldsymbol{g} \in \mathbb{R}^c$ is bias, c and d are hyperparameters.

[3]Tay et al. adopts standard LSTM encoder in their network and use the last hidden state as the output. In this paper we adopt the Bi-LSTM encoder, therefore, the first hidden state contains the final information of the Bi-LSTM operation.

Table 1: Descriptive statistics

Dataset	Train	Test	Total
Reddit(/r/movies)	13,515	1,504	15,019
Reddit(/r/technology)	12,136	1,349	13,485
IAC-V1	1,794	200	1,794
IAC-V2	4,078	454	4532
Tweets(Riloff)	1,368	588	1,956
Tweets(Ghosh)	51,189	3,742	54,931

In this work, we follow the concept of LRBP to pool information from two input feature vectors: $\boldsymbol{f}_a \in \mathbb{R}^k$ and $\boldsymbol{f}_l \in \mathbb{R}^d$. Considering the fact that sentence representation vector usually doesn't employ very large number of dimensions, we simplify the approach by removing the parameter matrix $\mathbf{P}$ in equation (9) to reduce the parameters to be learned and do not further reduce the input dimension. We calculate the final projection feature vector for the input sentence as:

$$\boldsymbol{f} = \mathbf{U}^{\mathrm{T}} \cdot \boldsymbol{f}_a \circ \mathbf{V}^{\mathrm{T}} \cdot \boldsymbol{f}_l + \boldsymbol{g} \tag{10}$$

where $\boldsymbol{f} \in \mathbb{R}^c$, and c is a hyperparameter.

We use $\boldsymbol{f}$ as the final feature vector for the input sentence and we input $\boldsymbol{f}$ into a standard softmax classification layer to make the sarcasm prediction:

$$\boldsymbol{p}_i = Softmax(\mathbf{W}_f \cdot \boldsymbol{f} + \boldsymbol{b}), \tag{11}$$

where $\boldsymbol{p}_i \in \mathbb{R}^2$ representing the probability of whether the input sentence is sarcastic or not, and $\mathbf{W}_f \in \mathbb{R}^{2 \times c}$, $\boldsymbol{b} \in \mathbb{R}^2$ are parameters to be learned.

3.5 Training Objective

For the final sarcasm classification task, we optimize a standard cross-entropy loss function:

$$J(\theta) = -\sum_{i=1}^{N} [y_i \cdot log p_i + (1 - y_i) \cdot log(1 - p_i)] + \lambda \cdot R \tag{12}$$

where N is the size of training dataset, y_i is the true label for sentence i. $\theta = \{\mathbf{M}_{i,j}, \mathbf{U}, \mathbf{V}, \boldsymbol{g}, \mathbf{W}_f, \boldsymbol{b}\}$ are model parameters. $R = \|\theta\|_{L2}$ is the standard L2 regularization term and λ is a hyperparameter measuring the weight of regularization term. d is the hyperparameter in the bi-directional LSTM encoder referring to the number of dimension of the output feature vector. c is the hyperparameter in the low-rank bilinear pooling process referring to the number of dimension of the final sentence representation vector.

4 EXPERIMENTS

4.1 Datasets

To validate our model performance and make it comparable with baseline models, we follow previous studies and evaluate our model on two datasets collected from Reddit and another two datasets collected from IAC (Internet Argument Corpus) [10, 35, 42]. In addition, we notice that existent studies also test models on two popular Twitter datasets provided by Riloff et al. [34] and Ptáček et al. [29]. However, these two studies only provide Tweets IDs, and we are only able to recollect a comparable dataset with Riloff et al.[34]

but fail to match the dataset provided by Ptáček et al [29] [4]. As a complement, we test our model on a recent collected Twitter dataset provided by Ghosh and Veale [11]. Reddit and Twitter datasets mainly contain short texts, while IAC datasets focus on long texts. Descriptive statistics for all six datasets are summarized in Table 1.

- **Reddit**: Reddit is one of the world's largest online communities with millions of users and hundreds and thousands of topics. Provided by Khodak et al. [20], posts in Reddit dataset are annotated based on the '/s' tag left by authors themselves. We use two subsets: */r/movies* and */r/technology* that are tested by Tay et al. [35] to make our experiment results comparable [5].
- **IAC**(Internet Argument Corpus): the original corpus was collected by Walker et al. [39] from the online debate forum *4forums.com* to study political debates on online communities, Lukin and Walker annotate this corpus for sarcasm detection [24]. We test our model on the same two versions of the dataset used by Tay et al. [34], denoted as IAC-V1 and IAC-V2 respectively[6].
- **Tweets**: Twitter is the world's biggest microblogging platform where users can share any of their news, ideas and thoughts with their followers. Two twitter based datasets are widely used by NLP scholars: Riloff et al. provide a dataset with sentences manually annotated and checked, it is also known as the very first public tweets sarcasm dataset [34]; Ghosh and Veale collect a dataset in which tweets are labeled by hash-tags such as *#sarcasm*, *#sarcastic* and *#ironie* [29], and they also employ a feedback-based system which allows them to contact tweet authors to examine the validity of the sarcasm labels.

4.2 Baseline Models

We compare our model with following neural network based models to evaluate our model:

- **NBOW** is a baseline model using the average of word embedding vectors as sentence representation, which is adopted as the input of a standard logistic regression.
- **ATT-LSTM** is an attention based LSTM network proposed by Yang et al. [41], it employs an attention mechanism on all hidden states of the LSTM encoder. Although this model is not exclusively designed for sarcasm detection, it achieves outstanding performance on this task.
- **GRNN** is a bi-directional gated recurrent neural network model proposed by Zhang et al. [42], it employs a gated pooling method to extract content features from a gated recurrent neural network, and also employs a standard pooling method to extract contextual features. This model is among the very first models utilizing neural features for sarcasm detection and demonstrates improved model accuracy compared to traditional discrete feature engineering approaches.
- **CNN-LSTM-DNN** is proposed by Ghosh and Veale [10], it combines Convolutional Neural Network(CNN), Long Short-Term Memory(LSTM) and Deep Neural Network(DNN) into its model architecture. It employs two CNN layers with 1D convolution and two LSTM layers for feature engineering and employs a DNN layer for prediction. it was once the state-of-the-art model for sarcasm detection before Tay et al. [35].
- **SIARN** and **MIARN** are the state-of-the-art models for sarcasm detection [35] . It utilizes an intra-attention mechanism to overcome limitations from sequential neural networks and could better capture words' incongruities. **SIARN** employs a single-dimension intra-attention network and **MIARN** employs a multi-dimension intra-attention network.

4.3 Experimental Settings

To make our experiment results comparable with existent models, we align our experimental settings with Tay et al. [35]. We preprocess our datasets by removing words that appear only once in datasets, and sentences and documents with less than 5 tokens or contain hyper-links are also removed. Word embedding space in our model is generated by GloVe [28] with the dimension of word feature vector k fixed at 100 and we fine-tune the embedding during the training process. Considering the typical length of reviews and documents contained in our six datasets, we fix the length of each input sentence at (20, 20, 60) for datasets from Reddit, Twitter and IAC respectively.

All models are developed using TensorFlow and are trained on a GTX 1080Ti GPU. For the training process, we use the RMSProp optimizer[36] to optimize model parameters with learning rate equals to 0.01 and we train maximum 200 epochs for each model. The training batch size for all datasets is tuned amongst {64, 256, 512}. The L2 regularization is set to 10^{-5} for short texts contained in Reddit and Twitter datasets, and 10^{-3} for long texts covered in IAC datasets. Hyperparameter d in the Bi-LSTM encoder and hyperparameter c in the LRBP process are both fixed at 100.[7]

4.4 Experiment Results

We compare our model with baselines based on standard evaluation metrics including precision, recall, $F1$ score and the accuracy[8]. Experiment results on different datasets are reported in Table 2, 3 and 4.

On the two datasets collected from Reddit, we observe our models improve the state-of-the-art performance substantially. SMSD and SMSD-BiLSTM dominate previous baselines in precision, recall, $F1$ score and accuracy. The specific improvements differ in terms of evaluation metrics and datasets: on the Reddit(/r/movies) dataset, compared with the best performing baseline model, SMSD and SMSD-BiLSTM improve model precision by around 13%, improve recall by around 15-16%, improve the $F1$ score by around 11%

[4]We find the original Riloff dataset on a personal Github page and the dataset can be downloaded at https://github.com/przhang/Sarcasm-Detection. However, we are unable to find the Ptáček dataset from any reliable sources, and after 4 years passed, a substantial amount of original tweets are deleted and we are unable to collect them from Twitter.

[5]We download Reddit datasets from http://nlp.cs.princeton.edu/SARC/2.0/.

[6]We download IAC datasets from https://nlds.soe.ucsc.edu/sarcasm1 and https://nlds.soe.ucsc.edu/sarcasm2 respectively.

[7]As an important baseline, models proposed by Tay et al. [35] are not open-sourced yet, therefore, we align most of our hyperparameters with Tay et al. to make experiment results comparable.

[8]Precision, recall, $F1$ are calculated based on positive samples and accuracy is calculated by averaging both the accuracies in positive and negative samples.

Table 2: Experiment results on Reddit datasets. Best result in previous baselines is in *italic*, and best result in our approach is in bold. Experiment results of baseline models are collected from Tay et al. [35].

	Reddit(/r/movies)				Reddit(/r/technology)			
Model	Precision	Recall	$F1$	Acc	Precision	Recall	$F1$	Acc
NBOW	67.33	66.56	66.82	67.52	65.45	65.62	65.52	66.55
ATT-LSTM (Yang et al. [41])	68.11	67.87	67.94	68.37	68.20	68.78	67.44	67.22
GRNN (Zhang et al. [42])	66.16	66.16	66.16	66.42	66.56	66.73	66.66	67.65
CNN-LSTM-DNN (Ghosh and Veale [11])	68.27	67.87	67.95	68.50	66.14	66.73	65.74	66.00
SIARN (Tay et al. [35])	69.59	*69.48*	69.52	69.84	*69.35*	*70.05*	*69.22*	69.57
MIARN (Tay et al. [35])	*69.68*	69.37	*69.54*	*69.90*	68.97	69.30	69.09	*69.91*
SMSD	**83.31**	74.42	78.62	77.79	89.33	**82.83**	85.96	81.54
SMSD-BiLSTM	82.09	**76.01**	**78.93**	**78.52**	**91.09**	81.88	**86.24**	**81.62**

Table 3: Experiment results on IAC datasets. Best result in previous baselines is in *italic*, and best result in our approach is in bold. Results of baseline models are from Tay et al. [35]

	IAC-V1				IAC-V2			
Model	Precision	Recall	$F1$	Acc	Precision	Recall	$F1$	Acc
NBOW	57.17	57.03	57.00	57.51	66.01	66.03	66.02	66.09
ATT-LSTM (Yang et al. [41])	58.98	57.93	57.23	59.07	70.04	69.62	69.63	69.96
GRNN (Zhang et al. [42])	56.21	56.21	55.96	55.96	62.26	61.87	61.21	61.37
CNN-LSTM-DNN (Ghosh and Veale [11])	55.50	54.60	53.31	55.96	64.31	64.33	64.31	64.38
SIARN (Tay et al. [35])	*63.94*	63.45	62.52	62.69	72.17	71.81	71.85	72.10
MIARN (Tay et al. [35])	63.88	*63.71*	*63.18*	*63.21*	*72.92*	*72.93*	*72.75*	*72.75*
SMSD	73.68	58.82	65.42	63.00	**88.76**	64.81	**74.92**	67.40
SMSD-BiLSTM	**75.79**	**59.50**	**66.67**	**64.00**	82.73	**67.54**	74.37	**68.72**

Table 4: Experiment results on Tweets datasets. Best result in previous baselines is in *italic*, and best result in our approach is in bold. Results of baseline models are from Ghosh and Veale [11] and Tay et al. [35]

	Tweets(Riloff)				Tweets(Ghosh)			
Model	Precision	Recall	$F1$	Acc	Precision	Recall	$F1$	Acc
NBOW	71.28	62.37	64.13	79.23	-	-	-	-
ATT-LSTM (Yang et al. [41])	68.78	68.63	68.71	77.69	-	-	-	-
GRNN (Zhang et al. [42])	66.32	64.74	65.40	76.41	-	-	-	-
CNN-LSTM-DNN (Ghosh and Veale [11])	69.76	66.62	67.81	78.72	*73.30*	*71.70*	*72.50*	-
SIARN (Tay et al. [35])	*73.82*	*73.26*	*73.24*	*82.31*	-	-	-	-
MIARN (Tay et al. [35])	73.34	68.34	70.10	80.77	-	-	-	-
SMSD	95.70	**100.00**	97.80	**99.32**	76.39	**72.56**	**74.42**	**80.09**
SMSD-BiLSTM	**96.77**	98.90	**97.83**	**99.32**	**78.65**	68.80	73.40	78.37

and model accuracy by 8-9%. On the Reddit(/r/technology) dataset, SMSD and SMSD-BiLSTM achieve around 10-11% increasing on precision, 12-13% increasing on recall, 16% increasing on $F1$ score and 11% increasing on model accuracy.

On the two datasets collected from IAC, SMSD and SMSD-BiLSTM also achieve outstanding performance. On the IAC-V1 dataset, MIRAN proposed by Tay et al. [35] has the highest recall though, SMSD-BiLSTM achieves around 11% increasing on precision, 3% increasing on $F1$ score and 1% increasing on model accuracy. On the IAC-V2 dataset, MIARN has higher recall and accuracy, yet SMSD has higher precision and $F1$ score. In addition, we also notice that our models are more robust in model performance across two different IAC datasets. For example, the difference of model accuracy between these two datasets is 4.4% for the SMSD, 4.7% for SMSD-BiLSTM, while the difference for SIARN and MIARN are over 9%. Among these baselines, only GRNN has similar stability with our model, while its $F1$ score and accuracy are much lower than ours.

Moreover, SMSD and SMSD-BiLSTM achieve state-of-the-art performance on the two datasets collected from Twitter. On the Rillof dataset, SMSD and SMSD-BiLSTM almost achieve perfect model performance, the accuracy and $F1$ score for both of our

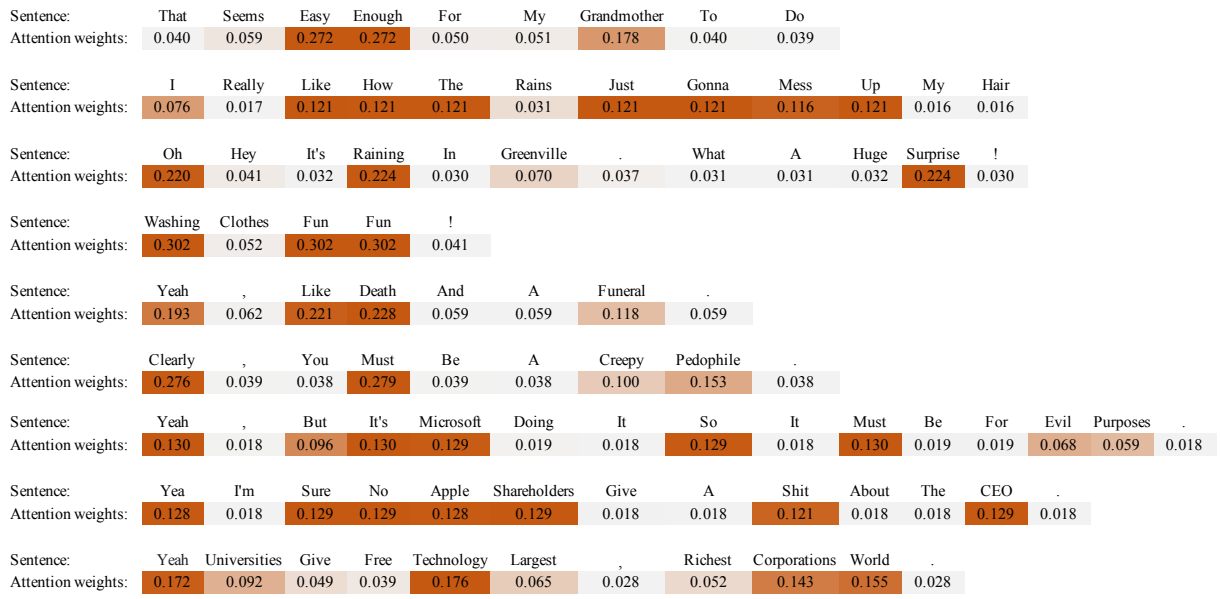

Figure 2: Visualization of Self-matching Results. For each sample, the first sentence shows the original input sentence, and the second sentence demonstrates the attention weights generated from the self-matching network, the darkness of the background varies according to the value of the corresponding attention weight.

models are over 97% [9]. The Ghosh dataset is not widely tested by extant studies yet, we only compare our model performance with the original paper [11] and we still find that our models outperform it, the SDSM-BiLSTM improves precision by around 5%, and SMSD improves recall, $F1$ score by 1% and 2% respectively.

As we have discussed in model description section, we design a self-matching network to compare word-to-word information aiming to search for potential incongruity information; and we also incorporate a bi-directional LSTM encoder to help capture sequential information. We report model performance with and without the Bi-LSTM separately to help us discriminate the value of these two components. As experiment results have demonstrated, in most circumstances, incorporating a bi-directional LSTM encoder will improve model performance. We test our models on six datasets and we find that on the IAC-V1 dataset, the SMSD-BiLSTM outperforms SMSD on four evaluation metrics. SMSD-BiLSTM generally have better performance than SMSD on two Reddit datasets except that SMSD has better precision on Reddit(/r/movies) and better recall on Reddit(/r/technology). These two models have comparable results on IAC-V2 and Tweets(Riloff) considering four evaluation metrics, except that SMSD-BiLSTM has higher accuracy and recall on IAC-V2. We notice that SMSD outperforms SMSD-BiLSTM on Tweets(Ghosh) in terms of recall, $F1$ score and accuracy. These comparison results indicate that the a bi-directional LSTM encoder indeed enriches model input by capturing sentence compositional information, and it helps to generate better sarcasm detection result.

4.5 Model Visualization

In this section we use a few model prediction results to showcase the mechanism of the self-matching network. The self-matching network attends word-to-word information simultaneously and the joint information are captured to attend the original sentence feature map. Therefore, words of conflict sentiments are more likely to be attended by our proposed model. In Figure 2 we depict a few sample sentences with their self-matched attention weights reported. We also demonstrate some bad cases in the prediction result to better understand under what circumstances our models would fail (see Figure 3)[10].

Examples demonstrate that the self-matching network is highly effective on attending words of incongruity information. For example. in the sentence *"That seems easy enough for my grandmother to do"* we notice that the phrase *"easy enough"* and the word *"grandmother"* have the highest attention weights; in sentence *"I really like how the rains just gonna mess up my hair"*, *"like"* and *"mess up"* have relatively high attention weights. These most attended words are usually of opposite sentiments compared to each other and are less likely to be found simultaneously in non-sarcastic expressions. The same pattern can be found across all the examples we provide in Figure 2. Therefore, the self-matching network has a strong power on capturing word pairs of incongruity information that can be taken as a strong indicator for the appearance of sarcasm.

[9]Indeed, the significant improvements on this dataset is beyond our expectation. As aforementioned, Rillof dataset is not provided officially and we collect the Rillof dataset from a personal Github link (as provided in previous section). We test the reliability of the dataset by checking the descriptive statistics of our collected dataset with the statistics reported in previous studies, such as Tay et al. [35] and we find no significant difference between these two datasets. We also notice that the Rillof dataset is unbalanced and evaluation metrics might be biased by prediction result on the dominant class. Therefore, we further examine the values of evaluation metrics from positive samples and negative samples respectively, and we find the results are highly similar.

[10]The authors want to thank two anonymous reviewers for the suggestion of providing a qualitative bad case analysis.

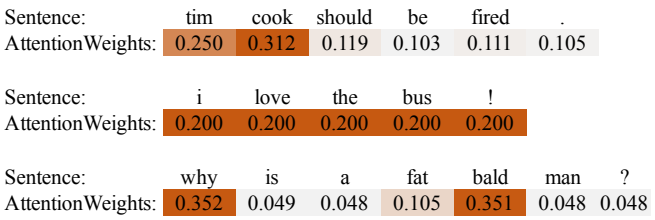

Figure 3: Bad case analysis. Three model prediction bad cases are reported, the first two samples are false negative, and the third sample is false positive.

In Figure 3 we report three bad cases: the first two are false negative and last one is false positive. The first case shows that our model fails to capture *tim cook* as a celebrity name and the sarcasm embedded in this sample replies heavily on external knowledge. In the second case, there are no conflict sentiments between those four words, and we have not further input to know that the author actually hates bus. In addition, the third false positive case *"why"* and *"bald"* have the highest attention weights, this sample sentence is a rhetorical question expression, which is highly similar with sarcastic expression. Therefore, our model is most likely to fail when the sarcasm relies on external knowledge or when the expression is a rhetorical question.

5 CONCLUSION

In this work we propose a novel self-matching network based on a modified co-attention mechanism to cater to sentence incongruity. The self-matching network allows us to capture the interaction between words to search for potential conflict sentiments. We also incorporate a bi-directional LSTM encoder to utilize sentence's compositional information. We conduct extensive experiments on six publicly available datasets. Experiment results demonstrate a strong evidence that our proposed model outperforms most of extant baselines in terms of standard evaluation metrics including precision, recall, $F1$score and accuracy. Our model even improve the state-of-the-art performance on some datasets dramatically. In addition, we also find supportive empirical evidence on utilizing sequential neural networks and low-rank bilinear pooling method can help improve sarcasm prediction result.

6 ACKNOWLEDGMENTS

This research is supported by Alibaba Group.

REFERENCES

[1] Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976* (2016).
[2] David Bamman and Noah A Smith. 2015. Contextualized Sarcasm Detection on Twitter. *ICWSM* 2 (2015), 15.
[3] Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2014. Italian irony detection in twitter: a first approach. In *The First Italian Conference on Computational Linguistics CLiC-it*. 28.
[4] Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 50–58.
[5] Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. 2015. Parsing-based sarcasm sentiment recognition in twitter data. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM, 1373–1380.
[6] Mondher Bouazizi and Tomoaki Ohtsuki. 2015. Opinion mining in Twitter: How to make use of sarcasm to enhance sentiment analysis. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE, 1594–1597.
[7] Elisabeth Camp. 2012. Sarcasm, pretense, and the semantics/pragmatics distinction. *Noûs* 46, 4 (2012), 587–634.
[8] John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes* 49, 6 (2012), 459–480.
[9] Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*. Association for Computational Linguistics, 107–116.
[10] Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*. 161–169.
[11] Aniruddha Ghosh and Tony Veale. 2017. Magnets for Sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 482–491.
[12] Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in Twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*. Association for Computational Linguistics, 581–586.
[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.
[14] Stacey L Ivanko and Penny M Pexman. 2003. Context incongruity and irony processing. *Discourse Processes* 35, 3 (2003), 241–279.
[15] Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)* 50, 5 (2017), 73.
[16] Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Vol. 2. 757–762.
[17] Aditya Joshi, Vaibhav Tripathi, Pushpak Bhattacharyya, and Mark J Carman. 2016. Harnessing Sequence Labeling for Sarcasm Detection in Dialogue from TV SeriesFriends'. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 146–155.
[18] Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are Word Embedding-based Features Useful for Sarcasm Detection? *arXiv preprint arXiv:1610.00883* (2016).
[19] Jihen Karoui, Farah Benamara, Véronique Moriceau, Nathalie Aussenac-Gilles, and Lamia Hadrich Belguith. 2015. Towards a contextual pragmatic model to detect irony in tweets. Association for Computational Linguistics (ACL).
[20] Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579* (2017).
[21] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Hadamard product for low-rank bilinear pooling. In *Proceedings of the 2017 International Conference on Learning Representation*.
[22] CC Liebrecht, FA Kunneman, and APJ van Den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not. (2013).
[23] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*. 289–297.
[24] Stephanie Lukin and Marilyn Walker. 2017. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. *arXiv preprint arXiv:1708.08572* (2017).
[25] DG Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC 2014 Proceedings*. ELRA.
[26] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, L Sutskever, and G Zweig. 2013. word2vec. *URL https://code. google. com/p/word2vec* (2013).
[27] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2, 1–2 (2008), 1–135.
[28] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[29] Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 213–223.
[30] Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 97–106.
[31] Antonio Reyes and Paolo Rosso. 2012. Making objective decisions from subjective data: Detecting irony in customer reviews. *Decision Support Systems* 53, 4 (2012), 754–760.

[32] Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering* 74 (2012), 1–12.

[33] Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation* 47, 1 (2013), 239–268.

[34] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 704–714.

[35] Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Reasoning with Sarcasm by Reading In-between. *arXiv preprint arXiv:1805.02856* (2018).

[36] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[37] Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. ICWSM-A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews.. In *ICWSM*. 162–169.

[38] Tony Veale and Yanfen Hao. 2010. Detecting ironic intent in creative comparisons.. In *ECAI*, Vol. 215. 765–770.

[39] Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A Corpus for Research on Deliberation and Debate.. In *LREC*. 812–817.

[40] Byron C Wallace, Eugene Charniak, et al. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 1035–1044.

[41] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.

[42] Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, The 26th International Conference on Computational Linguistics: Technical Papers*. 2449–2460.