Tweet Sarcasm Detection Using Deep Neural Network

Meishan Zhang[1], **Yue Zhang**[2] and **Guohong Fu**[1]
1. School of Computer Science and Technology, Heilongjiang University, China
2. Singapore University of Technology and Design
mason.zms@gmail.com,
yue_zhang@sutd.edu.sg,
ghfu@hotmail.com

Abstract

Sarcasm detection has been modeled as a binary document classification task, with rich features being defined manually over input documents. Traditional models employ discrete manual features to address the task, with much research effect being devoted to the design of effective feature templates. We investigate the use of neural network for tweet sarcasm detection, and compare the effects of the continuous automatic features with discrete manual features. In particular, we use a bi-directional gated recurrent neural network to capture syntactic and semantic information over tweets locally, and a pooling neural network to extract contextual features automatically from history tweets. Results show that neural features give improved accuracies for sarcasm detection, with different error distributions compared with discrete manual features.

1 Introduction

Sarcasm has received much research attention in linguistics, psychology and cognitive science (Gibbs, 1986; Kreuz and Glucksberg, 1989; Utsumi, 2000; Gibbs and Colston, 2007). Detecting sarcasm automatically is useful for opinion mining and reputation management, and hence has received growing interest from the natural language processing community (Joshi et al., 2016a). Social media such as Twitter exhibit rich sarcasm phenomena, and recent work on automatic sarcasm detection has focused on tweet data.

Tweet sarcasm detection can be modeled as a binary document classification task. Two main sources of features have been used. First, most previous work extracts rich discrete features according to the *tweet content* itself (Davidov et al., 2010; Tsur et al., 2010; González-Ibánez et al., 2011; Reyes et al., 2012; Reyes et al., 2013; Riloff et al., 2013; Ptáček et al., 2014), including lexical unigrams, bigrams, tweet sentiment, word sentiment, punctuation marks, emoticons, quotes, character ngrams and pronunciations. Some of these work uses more sophisticated features, including POS tags, dependency-based tree structures, Brown clusters and sentiment indicators, which depend on external resources. Overall, ngrams have been among the most useful features.

Second, recent work has exploited contextual tweet features for sarcasm detection (Rajadesingan et al., 2015; Bamman and Smith, 2015). Intuitively, the *history behaviors* for a tweet author can be a good indicator for sarcasm. Rajadesingan et al. (2015) exploit a behavioral approach to model sarcasm, using a set of statistical indicators extracted from both the target tweet and relevant history tweets. Bamman and Smith (2015) study the influences of tweet content features, author features, audience features and environment features, finding that contextual features are very useful for tweet sarcasm detection.

So far, most existing sarcasm detection methods in the literature leverage discrete models. While on the other hand, neural network models have gained much attention for related tasks such as sentiment analysis and opinion extraction, achieving the best results (Socher et al., 2013; dos Santos and Gatti, 2014; Vo and Zhang, 2015; Zhang et al., 2016). Success on these tasks shows potentials of neural network on sarcasm detection (Amir et al., 2016; Ghosh and Veale, 2016; Joshi et al., 2016b). There are two main advantages of using neural models. First, neural layers are used to induce features automatically,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2449–2460, Osaka, Japan, December 11-17 2016.

making manual feature engineering unnecessary. Such neural features can capture long-range and subtle semantic patterns, which are difficult to express using discrete feature templates. Second, neural models use real-valued word embedding inputs, which are trained from large scale raw texts, and are capable of avoiding the feature sparsity problem of discrete models.

In this paper, we exploit a deep neural network for sarcasm detection, comparing its automatic features with traditional discrete models. First, we construct a baseline discrete model that exploits the most typical features in the literature, including the features on the target tweet content and the features on historical tweets of the author, achieving competitive results as compared to the previous best systems.

Second, we build a neural model, with two sub neural networks to capture tweet content and contextual information, respectively. The two-component structure closely corresponds to the two feature sources of the baseline discrete model. We model the tweet content with a gated recurrent neural network (GRNN) (Cho et al., 2014b; Cho et al., 2014a), and use a gated pooling function for feature extraction. To model the salient words from the contextual tweets, we use pooling to extract features directly.

Results on a tweet datasets show that the neural model achieves significantly better accuracies compared to the discrete baseline, demonstrating the advantage of the automatically extracted neural features in capturing global semantic information. Further analysis shows that features from history tweets are as useful to the neural model as to the discrete model. We make our source code publicly available under GPL at `https://github.com/zhangmeishan/SarcasmDetection`.

2 Related Work

Features. Sarcasm detection is typically regarded as a classification problem. Discrete models have been used and most existing research efforts have focused on finding effective features. Kreuz and Caucci (2007) studied lexical features for sarcasm detection, finding that words, such as interjections and punctuation, are effective for the task. Carvalho et al. (2009) demonstrated that oral or gestural expressions represented by emoticons and special keyboard characters are useful indicators of sarcasm. Both Kreuz and Caucci (2007) and Carvalho et al. (2009) rely on unigram lexical features for sarcasm detection. More recently, Lukin and Walker (2013) extended the idea by using n-gram features as well as lexicon-syntactic patterns.

External sources of information have been exploited to enhance sarcasm detection. Tsur et al. (2010) applied features based on semi-supervised syntactic patterns extracted from sarcastic sentences of Amazon product reviews. Davidov et al. (2010) further extracted these features from sarcastic tweets. Riloff et al. (2013) identified a main type of sarcasm, namely contrast between a positive and negative sentiment, which can be regarded as detecting sarcasm using sentiment information. There has been work that comprehensively studies the effect of various features (González-Ibánez et al., 2011; González-Ibánez et al., 2011; Joshi et al., 2015).

Recently, contextual information has been exploited for sarcasm detection (Wallace et al., 2015; Karoui et al., 2015). In particular, contextual features extracted from history tweets by the same author has shown great effectiveness for tweet sarcasm detection (Rajadesingan et al., 2015; Bamman and Smith, 2015). We consider both traditional lexical features and the contextual features from history tweets under a unified neural network framework. Our observation is consistent with prior work: both sources of features are highly effective for sarcasm detection (Rajadesingan et al., 2015; Bamman and Smith, 2015).. To our knowledge, we are among the first to investigate the effect of neural networks on this task (Amir et al., 2016; Ghosh and Veale, 2016; Joshi et al., 2016b).

Corpora. With respect to sarcasm corpora, early work relied on small-scale manual annotation. Filatova (2012) constructed a sarcasm corpus from Amazon product reviews using crowdsourcing. Davidov et al. (2010) discussed the strong influence of hashtags on sarcasm detection. Inspired by this, González-Ibánez et al. (2011) used sarcasm-related hashtags as gold labels for sarcasm, creating a tweet corpus by treating tweets without such hashtags as negative examples. Their work is similar in spirit to the work of Go et al. (2009), who constructed a tweet sentiment automatically by taking emoticons as gold sentiment labels.

The method of González-Ibánez et al. (2011) was adopted by Ptáček et al. (2014), who created a

sarcasm dataset for Czech. More recently, both Rajadesingan et al. (2015) and Bamman and Smith (2015) followed the method for building a sarcasm corpus. We take the corpus of Rajadesingan et al. (2015) for our experiments.

Neural network models. Although only very limited work has been done on using neural networks for sarcasm detection, neural models have seen increasing applications in sentiment analysis, which is a closely-related task. Different neural network architectures have been applied for sentiment analysis, including recursive auto-encoders (Socher et al., 2013), dynamic pooling networks (Kalchbrenner et al., 2014), deep belief networks (Zhou et al., 2014), deep convolutional networks (dos Santos and Gatti, 2014; Tang et al., 2015) and neural CRF (Zhang et al., 2015). This line of work gives highly competitive results, demonstrating large potentials for neural networks on sentiment analysis. One important reason is the power of neural networks in automatic feature induction, which can potentially discover subtle semantic patterns that are difficult to capture by using manual features. Sarcasm detection can benefit from such induction, and several work has already attempted for it (Amir et al., 2016; Ghosh and Veale, 2016; Joshi et al., 2016b). This motivates our work.

3 Baseline Discrete Model

We follow previous work in the literature, building a strong discrete baseline model using features from both the *target tweet* itself and its *contextual tweets*. The structure of the model is shown in Figure 1(a), which consists of two main components, modeling the target tweet and its contextual tweets, respectively. In particular, the **local component** (the left of Figure 1(a)) is used to extract features $\mathbf{f}$ from the target tweet content, and the **contextual component** (the right of Figure 1(a)) is used to extract contextual features $\mathbf{f}'$ from the history tweets of the author. Based on $\mathbf{f}$ and $\mathbf{f}'$, a logistic regression is used to obtain the output:

$$\mathbf{o} = \mathrm{softmax}\big(W_o \cdot (\mathbf{f} \oplus \mathbf{f}')\big), \tag{1}$$

where the matrix W_o is the model parameter matrix, $\mathbf{o}$ is the output two-bit sarcasm/non-sarcasm vector, and $\oplus$ denotes vector concatenation.

3.1 The Local Component

Given an input tweet $w_1, w_2 \cdots w_n$, we extract a set of sparse discrete feature vectors $\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n$ by instantiating a set of feature templates over each word, respectively. In particular, we follow Rajadesingan et al. (2015) and use three feature templates, including the current word w_i, the word bigram $w_{i-1}w_i$ and the word trigram $w_{i-2}w_{i-1}w_i$. The final local feature vector $\mathbf{f}$ is the sum of $\mathbf{f}_i$ from all words: $\mathbf{f} = \sum_{i=1}^{n} \mathbf{f}_i$.

3.2 The Contextual Component

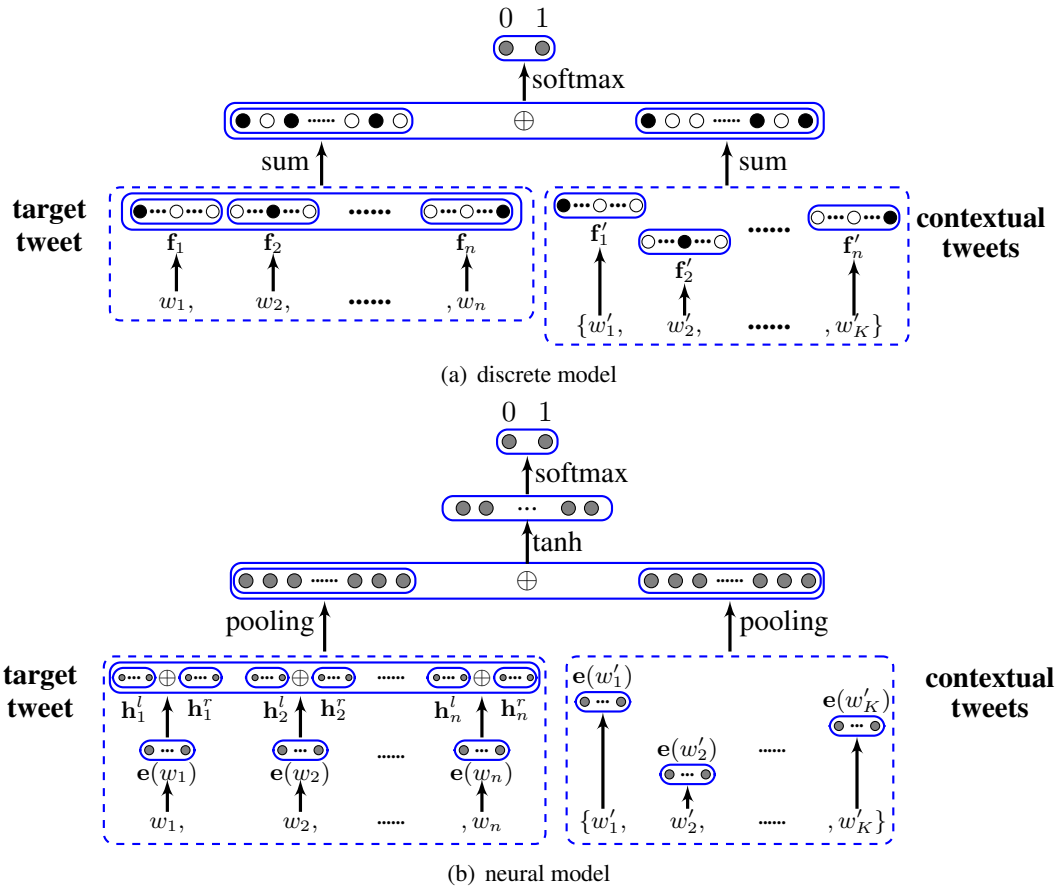
We follow Bamman and Smith (2015) for defining the features of the contextual tweets. In particular, a set of salient words are extracted from history tweets of the target tweet author, which can reflect the tendency of the author in using irony or sarcasm towards certain subjects.

First, we extract a number of most recent history tweets by using Twitter API[1], setting the maximum number of history tweets to 80.[2] The words in the history tweets are sorted by their *tf-idf* values. To estimate *tf* and *idf*, we regard the set of history tweets for a given tweet as one document, and use all the tweets in the training corpus to generate a number of additional documents. We choose a fixed-number of contextual tweet words with the highest *tf-idf* values for contextual features.

Denoting the set of words extracted from contexts as $\{\mathbf{w}'_1, \mathbf{w}'_2, \cdots, \mathbf{w}'_K\}$, where K is a hyper-parameter set manually, we use a single feature template w_i the extract a sparse feature vector $\mathbf{f}'_i$ for each word. The final contextual feature is the sum of all unigram features: $\mathbf{f}' = \sum_{i=1}^{K} \mathbf{w}'_i$. The set of baseline features, adopted from Rajadesingan et al. (2015) and Bamman and Smith (2015), are simple yet effective, giving highly competitive accuracies in our experiments.

[1] https://dev.twitter.com

[2] We use the data shared by Rajadesingan et al. (2015) directly, following their setting of history tweets.



(a) discrete model

(b) neural model

Figure 1: Discrete and neural models for tweet sarcasm detection (• denotes 1, ○ denotes 0, and • denotes a real-value feature).

4 Proposed Neural Model

In contrast to the discrete model, the neural model explores low-dimensional dense vectors as input. Figure 1(b) shows the overall structure of our proposed neural model, which has two components, corresponding to the **local** and the **contextual** components of the discrete baseline model, respectively. The two components use neural network structures to extract dense real-valued features $\mathbf{h}$ and $\mathbf{h}'$ from the local and history tweets, respectively, and we add a non-linear hidden layer to combine the neural features from the two components for classification. The output nodes can computed by:

$$\begin{aligned} \mathbf{c} &= \tanh(W_c(\mathbf{h} \oplus \mathbf{h}') + \mathbf{b}_c), \\ \mathbf{o} &= \text{softmax}(W_o\mathbf{c}), \end{aligned} \tag{2}$$

where the matrices W_c and W_o, and the vector $\mathbf{b}_c$ are model parameters.

As Figure 1 shows, the neural model is designed in such a way so that the correspondence between the model and the discrete baseline is maximized at the level of feature sources, for the convenience of direct comparison.

4.1 The Local Component

As shown on the left of Figure 1(b), we use a bi-directional gated recurrent neural network (GRNN) to model a tweet. The input layer of the network, represented by $\mathbf{x}_i$ at each position of the input tweet, is the concatenation of three consecutive word vectors, with the current word $\mathbf{w}_i$ in the center. With respect to the source of information, it is similar to the trigram feature templates of the baseline discrete model. Formally, at each word location i, the input vector is $\mathbf{x}_i = [\mathbf{e}(w_{i-1}), \mathbf{e}(w_i), \mathbf{e}(w_{i+1})]$, where $\mathbf{e}$ is a function to obtain dense embeddings for words based on a matrix E, which is a model parameter.

A recurrent neural network is used to capture sequential features automatically, hence giving semantic information over the whole input tweets. Compared with the vanilla recurrent neural network structure, gated recurrent neural networks such as long-short-term-memory (Hochreiter and Schmidhuber, 1997) apply gate structures to effectively reduce the issues of exploding and diminishing gradients (Pascanu et al., 2013; Yao et al., 2015), and therefore have been widely used as a more effective form of recurrent neural networks.

We exploit two efficient GRNNs to obtain a left-to-right ($\mathbf{h}_1^l \mathbf{h}_2^l \cdots \mathbf{h}_n^l$), and a right-to-left hidden node sequence ($\mathbf{h}_n^r \mathbf{h}_{n-1}^r \cdots \mathbf{h}_1^r$), respectively. Taking the left-to-right GRNN as an example, the hidden node vectors $\mathbf{h}_i^l$ are computed by:

$$\mathbf{h}_i^l = (\mathbf{1} - \mathbf{z}_i^l) \odot \mathbf{h}_{i-1}^l + \mathbf{z}_i^l \odot \tilde{\mathbf{h}}_i^l$$
$$\tilde{\mathbf{h}}_i^l = \tanh(W_1^l \mathbf{x}_i + U_1^l(\mathbf{r}_i^l \odot \mathbf{h}_{i-1}^l) + \mathbf{b}_1^l)$$
$$\mathbf{z}_i^l = \text{sigmoid}(W_2^l \mathbf{x}_i + U_2^l \mathbf{h}_{i-1} + \mathbf{b}_2^l)$$
$$\mathbf{r}_i^l = \text{sigmoid}(W_3^l \mathbf{x}_i + U_3^l \mathbf{h}_{i-1}^l + \mathbf{b}_3^l)$$

where the $\mathbf{z}_i^l$ and $\mathbf{r}_i^l$ are two gates, and $\odot$ denotes Hadamard product. W_1^l, U_1^l, W_2^l, U_2^l, W_3^l, U_3^l, $\mathbf{b}_1^l$, $\mathbf{b}_2^l$ and $\mathbf{b}_3^l$ are model parameters.

We use the same method to obtain $\mathbf{h}_i^r$ in the reverse direction, with the corresponding model parameters $W_1^l, U_1^l, \cdots\cdots, \mathbf{b}_3^l$ being W_1^r, U_1^r, W_2^r, U_2^r, W_3^r, U_3^r, $\mathbf{b}_1^r$, $\mathbf{b}_2^r$ and $\mathbf{b}_3^r$, respectively. After both hidden node sequences are computed, we concatenate the bi-directional hidden nodes at each position, obtaining $\mathbf{h}_1 \mathbf{h}_2 \cdots \mathbf{h}_n$ ($\mathbf{h}_i = \mathbf{h}_i^l \oplus \mathbf{h}_i^r$).

We apply a gated pooling function over the variable-length sequence $\mathbf{h}_1 \mathbf{h}_2 \cdots \mathbf{h}_n$ to project these GRNN hidden node features into a global feature vector $\mathbf{h}$. Formally, the pooling function is defined by $\mathbf{h} = \sum_{i=1}^n \alpha_i \odot \mathbf{h}_i$, where the α_i values are computed automatically by $\alpha_i \infty \exp\big(\tanh(W_g \mathbf{h}_i + \mathbf{b}_g)\big)$ with the constrain of $\sum_1^n \alpha_i = 1$, W_g and $\mathbf{b}_g$ are model parameters. Here α_is are vectors that control the bit-wise combination between the hidden vectors $h_i, i \in [1 \cdots n]$.

The gated pooling add to the degree of flexibility in the interpolation compared with max, min and average pooling techniques, which are commonly used to extract features from variable length vector sequences. For example, when all α_is are equal, the resulting pooling effect is the same as the average pooling function. This gated pooling mechanism is similar in spirit to the attention method of Bahdanau et al. (2014), but is used for each element in the operated vectors rather than the full vectors.

Note that our baseline discrete model and neural model have highly similar structures, differing mainly in the use of manual discrete features and automatic neural features. In particular, the discrete feature vector $\mathbf{f} = \sum_{i=1}^n \mathbf{f}_i$ can be regarded as being obtained by using a *sum* pooling function $\mathbf{f} = \sum_{i=1}^n \alpha_i \odot \mathbf{f}_i$, where $\alpha_i = \mathbf{1}$ ($i \in [1, n]$). Here $\mathbf{f}_i$ is a discrete feature vector with manual feature engineering. The neural model obtains $\mathbf{f}$ also by pooling, with α_i being trained automatically. Different from $\{\mathbf{f}_i\}$, the features $\{\mathbf{h}_i\}$ are obtained through automatical feature extraction via GRNN, rather than manual combination of one-hot features.

4.2 The Contextual Component

We follow the discrete baseline, using the same contextual tweet words extracted from history tweets for contextual features. Different from target tweet words, contextual tweet words are separate words without structures, and therefore it is unnecessary to use structured neural networks such as GRNNs to model them. As a result, we directly apply the gated pooling function to project their embedding vectors into a fixed-dimensional feature vector $\mathbf{h}'$.

5 Training

We use supervised learning with a training objective to minimize the cross-entropy loss over a set of training examples $(x_i, y_i)|_{i=1}^N$, plus with a l_2-regularization term,

$$L(\theta) = -\sum_{i=1}^{N} \log p_{y_i} + \frac{\lambda}{2} \parallel \theta \parallel^2,$$

where θ is the set of model parameters, and p_{y_i} is the model probability of the gold-standard output y_i, which is computed by using logistic regression over the output vector $\mathbf{o}$ in Eq (1) and (2) for the discrete and neural models, respectively.

Online AdaGrad (Duchi et al., 2011) is used to minimize the objective function for both discrete and neural models. All the matrix and vector parameters are initialized by uniform sampling in $(-0.01, 0.01)$. The initial values of the embedding matrix E can be assigned either by using the same random initialization as the other parameters, or by using word embeddings pre-trained over a large-scale tweet corpus. We obtain pre-trained tweet word embeddings using *GloVe* (Pennington et al., 2014)[3]. Embeddings are fine-tuned during training, with E belonging to model parameters.

6 Experiments

6.1 Experimental Settings

6.1.1 Data

We use the dataset of Rajadesingan et al. (2015) to conduct our experiments, collected by querying the Twitter API using the keywords #sarcasm and #not, and filtering retweets and non-English tweets automatically. In total, Rajadesingan et al. (2015) collected 9,104 tweets that are self-described as sarcasm by the authors. We stream the tweet corpus using the tweet IDs they provide.[4]

We remove the #sarcasm and #not hashtags from the tweets, assigning to them the *sarcasm* output tags for training and evaluation. General tweets that are *non-sarcastic* are also obtained using the tweet IDs shared by Rajadesingan et al. (2015). For each tweet, a set of history tweets are extracted using Twitter API, for obtaining of contextual tweet words. We remove the #sarcasm and #not hashtags of the history tweets also, in order to avoid predicting sarcasm by using explicit clues.

Our models are evaluated on a balanced and an imbalanced dataset, respectively, where the balanced dataset includes equal sarcastic and non-sarcastic tweets, and the imbalanced dataset has a sarcasm:non-sarcasm ratio of 1:4.

6.1.2 Evaluation

We perform ten-fold cross-validation experiments and exploit the overall accuracies of sarcasm detection as the major evaluation metric. We report the macro F-measures as well, considering the data imbalance. Concretely, for both sarcasm and non-sarcasm, we compute their precisions, recalls and F-measures, respectively, and then we report the averaged F-measure. To tune the model hyper-parameters, we choose 10% of the training dataset as the development corpus.

6.2 Hyper-parameters

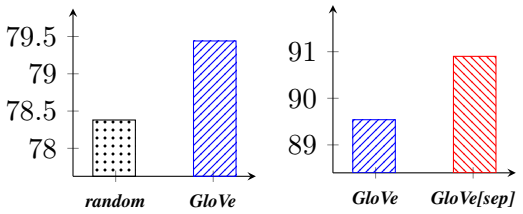
There are several important hyper-parameters in our models, and we tune their values using the development corpus. For both the discrete and neural models, we set the regularization weight $\lambda = 10^{-8}$ and the initial learning rate $\alpha = 0.01$. For the neural models, we set the size of word vectors to 100, the size of hidden vectors in GRNNs to 100, and the size of the non-linear combination layer to 50. One exception is the maximum number of history tweet words, which is 100 as default in align with Bamman and Smith (2015). We will show that the performance is still increasing when the number becomes larger in the next subsection.

6.3 Development Results

We conduct development experiments to study the effect of pre-trained word embeddings for the neural models, as well as the effect of the contextual information for both sparse and neural models. These experiments are performed on the balanced dataset.

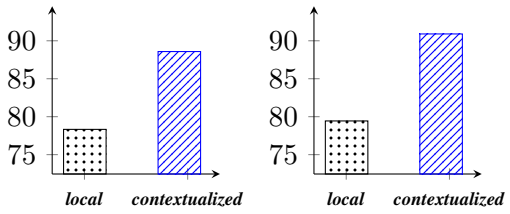
[3] http://nlp.stanford.edu/projects/glove/

[4] We are unable to obtain all the sarcastic tweets, due to modified authorization status of some tweets.



(a) local (b) contextualized

Figure 2: Influence of word representations.



(a) discrete (b) neural

Figure 3: Influence of contextual features.

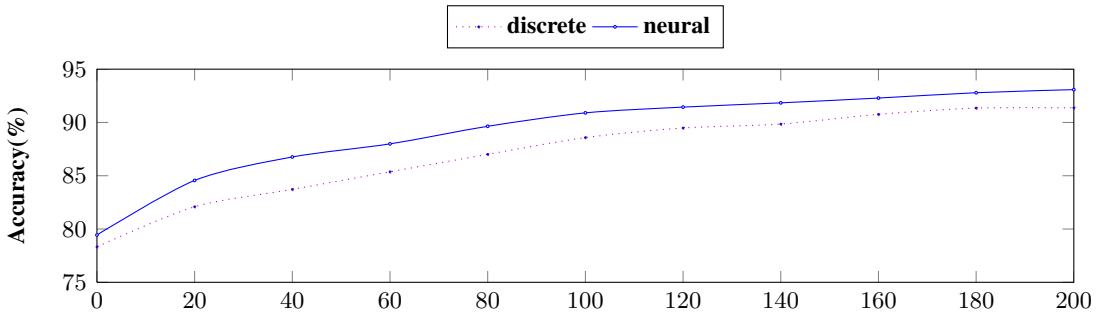


Figure 4: Developmental results with respect to different number of history tweet words.

6.3.1 Initialization of Word Embeddings

We use the neural model with only local features to evaluate the effect of different word embedding initialization methods. As shown in Figure 2(a), a better accuracy is obtained by using *GloVe* embeddings for initialization compared with random initialization. The finding is consistent with previous results in the literature on other NLP tasks, which show that pre-trained word embeddings can bring better accuracies (Collobert et al., 2011; Chen and Manning, 2014).

6.3.2 Differentiating Local and Contextual Word Embeddings

We can obtain embeddings of contextual tweet words using the same looking-up function as target tweet words, thereby giving each word a unique embedding regardless whether it comes from the target tweet to classify or its history tweets. However, the behavior of contextual tweet words should intuitively be different, because they are used as different features. An interesting research question is that whether separate embeddings lead to improved results. We investigate the question by using two embedding look-up matrices E and E', for target tweet words and contextual tweet words, respectively. The result in Figure 2(b) confirms our assumption, showing an improved accuracy by separating the two types of embeddings for each word.

Based on the above observation, we use *GloVe* embeddings for initialization in our final neural models, and use separate embedding matrices for target and contextual tweet words.

6.3.3 Effect of Contextual Features

Previous work has shown the effectiveness of contextual features for discrete sarcasm detection models (Rajadesingan et al., 2015; Bamman and Smith, 2015). Here, we study their effectiveness under both discrete and neural settings. The results are shown in Figure 3. It can be seen that contextual tweet information is highly useful under the neural setting also, which is consistent with previous work for the discrete models.

In more detail, we look at the performance with different maximum number of contextual words, in order to see the potential of contextual features. Figure 4 shows the development results, with the number range from 0 to 200, where 0 denotes the local model. As shown, the performance is consistently increasing with the increase of maximum contextual word number, although in this work we choose this

Model	Balanced		Imbalanced	
	Accuracy	F-measure	Accuracy	F-measure
Local				
baseline	78.55	78.53	86.45	75.14
neural	**79.29**‡	**79.36**‡	**87.25**‡	**77.37**‡
Riloff et al. (2013)	77.26	–	78.40	–
baseline l_1	78.56	–	81.63	–
Contextualized				
baseline	88.10	88.11	91.15	85.92
neural	**90.74**‡	**90.74**‡	**94.10**‡	**90.26**‡
SCUBA++	86.08	–	89.81	–

Table 1: Final results of our proposed models, where ‡ denotes a p-value below 10^{-3} compared to the baseline by pairwise t-test.

value by 100 to align with (Bamman and Smith, 2015).

6.4 Final Results

Table 1 shows the final results of our proposed models on both the balanced and the imbalanced datasets. The neural models show significantly better accuracies compared to the corresponding discrete baselines. Take the balanced data for example. Using only local tweet features, the neural model achieves an accuracy of 78.55%, significantly higher compared to the accuracy of 79.29% by the discrete model. Using also context tweet features, the accuracy of the neural model goes up to 90.74%, showing the strength of the history information. The F-measure values are consistent with the accuracies. These results demonstrate large advantages for the neural models on the task.

One interesting finding is that although the accuracies of imbalanced dataset are higher than those of balanced one, the micro F-measure values are on the contrary. The most possible reason could be the label bias of the imbalanced dataset, because detailed results show that the F-measures of sarcasm decrease significantly on the imbalanced dataset. According the final results, we can find both neural and contextual features can make up the F-measure gaps between balanced and imbalanced datasets, which further demonstrates the advantages of our final model.

We also compare the neural model with other sarcasm detection models in the literature. Shown in Table 1, Riloff et al. (2013) is lexicon-based model based on target tweet words only, identifying sarcasm by checking whether both positive and negative sentiment exist. *SCUBA++* shows the best results of Rajadesingan et al. (2015), using a contextualized model. Our baseline model gives higher accuracies compared to this state-of-the-art model, despite using similar features. One reason can the use of different optimization. For example, *baseline-l_1* shows the accuracy of our baseline using l_1 regularization instead of l_2, which yields variations of up to 1%. Nevertheless, the main purpose of the comparison is to show that our baseline is comparable to the best systems. Bamman and Smith (2015) report an accuracy of 75.4% on a balanced dataset, which is lower than our result. However, they performed evaluation on a different set of data, thus the results are not directly comparable.

6.5 Analysis

In order to better understand the differences between neural and manual features, we compare the discrete and neural models in more details on the test dataset. We focus on the balanced setting, and compare the contextualized models.

6.5.1 Error Characteristics

Figure 5 shows the output sarcasm probabilities of both models on each test tweet, where the x-axis represents the discrete model and the y-axis represents the neural model. The shapes + and • represent the gold-standard sarcasm and non-sarcasm labels, respectively. A probability value above 0.5 corresponds

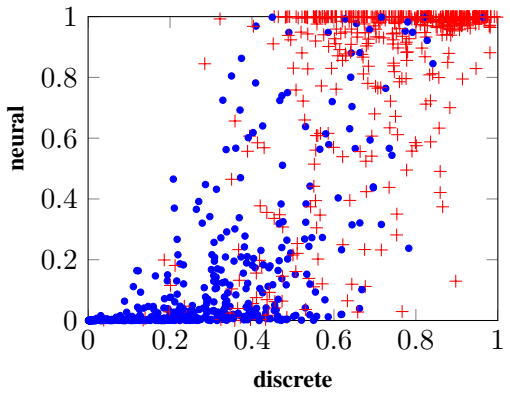


Figure 5: Error characteristic comparison, where + denotes sarcasm and • denotes non-sarcasm.

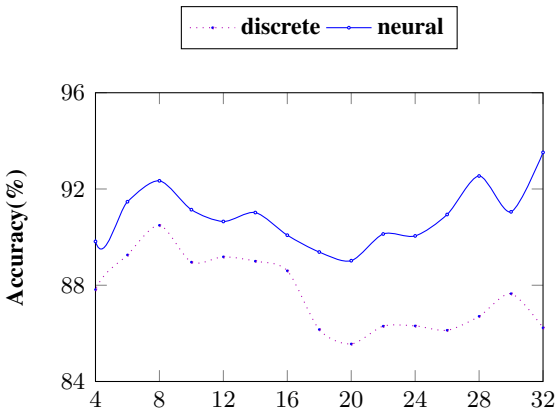


Figure 6: Accuracies against tweet length.

sarcasm	non-sarcasm
I guess finally knowing what it could have been makes me better .	so happy my brother has so many good people to help him with his move next weekend.
trying to fix my car is exactly what i wanna be doing on a saturday night.	Never go a day without telling your parents you love them

Table 2: Examples which the neural model predicted correctly but the discrete model incorrectly.

to the *sarcastic* output label. Intuitively, "+"s on the right half and "•"s on the left half of the figure show the examples that the discrete model predicts correctly, and "+"s on the top half and "•"s on the bottom half of the figure show the examples that the neural model predicts correctly.

As shown in the figure, most "+"s are in the top-right area, and most "•"s are in the bottom-left area, which indicates that the accuracies of both models are reasonably high. On the other hand, the samples are more scattered along the x-axis. This shows that the neural model is more confident in its predictions for most examples, demonstrating the discriminate power of the automatic neural features as compared with the manual discrete features.

6.5.2 Impact of Tweet Length

The GRNN neural model can potentially capture non-local syntactical and semantic information. We verify this assumption by comparing the accuracies of the neural and discrete models with respect to the tweet length. As shown in Figure 6, the neural model consistently outperforms the discrete model with respect to the tweet length. For longer tweets, the accuracies of the discrete model drops significantly, but those of the neural model remains stable.

6.5.3 Example Outputs

Table 2 shows some example sentences that the neural model predicted correctly, but the discrete model predicted incorrectly. Understanding sarcasm in the positive case requires global semantic information, which is better captured by non-local features from the recurrent neural network model. For the two cases with *non-sarcasm* gold labels, there are surface features such as "so happy", "so many" and "never", which are useful indicators of sarcasm for the discrete model. These features are local and can occur in both sarcasm and non-sarcasm tweets, thereby reducing the confidence of the discrete model (as shown in Figure 5) and can cause relatively more mistakes.

7 Conclusion

We constructed a deep neural network model for tweet sarcasm detection. Compared with traditional models with manual discrete features, the neural network model has two main advantages. First, it is free

from manual feature engineering and external resources such as POS taggers and sentiment lexicons. Second, it leverages distributed embedding inputs and recurrent neural networks to induce semantic features. The neural network model gave improved results over a state-of-the-art discrete model. In addition, we found that under the neural setting, contextual tweet features are as effective for sarcasm detection as with discrete models.

Acknowledgments

We thank the anonymous reviewers from COLING 2016, ACL 2016, NAACL 2016, AAAI 2016 and EMNLP 2015 for their constructive comments, which helped to improve the final paper. This work is supported by National Natural Science Foundation of China (NSFC) grants 61602160 and 61672211, Natural Science Foundation of Heilongjiang Province (China) grant F2016036, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mario J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. In *CONLL 2016*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014EMNLP*, pages 1724–1734.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

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*, pages 107–116.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014*, pages 69–78.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *LREC*, pages 392–398.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th WASSA*, pages 161–169.

Raymond W Gibbs and Herbert L Colston. 2007. *Irony in language and thought: A cognitive science reader*. Psychology Press.

Raymond W Gibbs. 1986. On the psycholinguistics of sarcasm. *Journal of Experimental Psychology: General*, 115(1):3.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th ACL*, pages 581–586.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd ACL*, pages 757–762.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016a. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016b. Are word embedding-based features useful for sarcasm detection? In *EMNLP*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd ACL*, pages 655–665. Association for Computational Linguistics.

Jihen Karoui, Benamara Farah, Véronique MORICEAU, Nathalie Aussenac-Gilles, and Lamia Hadrich-Belguith. 2015. Towards a contextual pragmatic model to detect irony in tweets. In *Proceedings of the 53rd ACL*, pages 644–650.

Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4.

Roger J Kreuz and Sam Glucksberg. 1989. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of Experimental Psychology: General*, 118(4):374.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 30–40.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*, pages 1310–1318.

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

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014*, pages 213–223, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

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*, WSDM '15, pages 97–106.

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

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

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 *EMNLP*, pages 704–714.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the EMNLP*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. 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)*, pages 1014–1023, July.

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*.

Akira Utsumi. 2000. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics*, 32(12):1777–1806.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of IJCAI*, BueNos Aires, Argentina, August.

Byron C. Wallace, Do Kook Choe, and Eugene Charniak. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd ACL*, pages 1035–1044.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the EMNLP*, pages 612–621.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Shusen Zhou, Qingcai Chen, Xiaolong Wang, and Xiaoling Li. 2014. Hybrid deep belief networks for semi-supervised sentiment classification. In *Proceedings of COLING 2014*, pages 1341–1349. Dublin City University and Association for Computational Linguistics.