Expect the unexpected: Harnessing Sentence Completion for Sarcasm Detection

Aditya Joshi[1,2,3], Samarth Agrawal[2],
Pushpak Bhattacharyya[2], and Mark J Carman[3]

[1] IITB-Monash Research Academy, Mumbai, India
[2] Indian Institute of Technology Bombay, Mumbai, India
[3] Monash University, Melbourne, Australia
✉ {adityaj,samartha,pb}@cse.iitb.ac.in
✉ mark.carman@monash.edu

Abstract. The trigram '*I love being*' is expected to be followed by positive words such as '*happy*'. In a sarcastic sentence, however, the word '*ignored*' may be observed. The expected and the observed words are, thus, incongruous. We model sarcasm detection as the task of detecting incongruity between an observed and an expected word. In order to obtain the expected word, we use Context2Vec, a sentence completion library based on Bidirectional LSTM. However, since the exact word where such an incongruity occurs may not be known in advance, we present two approaches: an All-words approach (which consults sentence completion for every content word) and an Incongruous words-only approach (which consults sentence completion for the 50% most incongruous content words). The approaches outperform reported values for tweets but not for discussion forum posts. This is likely to be because of redundant consultation of sentence completion for discussion forum posts. Therefore, we consider an oracle case where the exact incongruous word is manually labeled in a corpus reported in past work. In this case, the performance is higher than the all-words approach. This sets up the promise for using sentence completion for sarcasm detection.

Key words: Sarcasm detection; sentence completion; sentiment analysis; LSTM

1 Introduction

Sarcasm is defined as "*the use of irony to mock or convey contempt*[1]". For example, the sentence '*I love being ignored*' is sarcastic. Automatic sarcasm detection is the task of predicting whether or not a given text contains sarcasm. Several statistical approaches have been proposed for sarcasm detection [1] [2] [3]. In addition, rule-based approaches based on evidences of sarcasm have also done well [4] [5] [6]. This paper presents another rule-based technique. Our technique is novel in its application of sentence completion for sarcasm detection.

[1] Source: Oxford Dictionary

As an introduction to the technique, consider the sarcastic sentence '*I love being ignored*'. A likely word to follow the trigram '*I love being*' would be a positive sentiment word such as '*happy*'. However, in the sarcastic sentence, the word '*ignored*' occurs. The word '*ignored*' in the sarcastic sentence is semantically distant from an expected word such as '*happy*'. This (dis)similarity can be used as an indicator of incongruity which is central to sarcasm, as per linguistic studies [7][8]. In order to obtain the expected word at a given position, we harness automatic sentence completion. Sentence completion predicts the most likely word at a given position in a sentence [9]. For our experiments, we use context2vec, a sentence completion toolkit [10]. Thus, our paper deals with the question:

Because incongruity in sarcasm is a phenomenon where the unexpected is observed, can sarcasm be detected using sentence completion?

A key assumption here is that a sentence completion toolkit trained on a large, general-purpose corpus follows the language model for non-sarcastic text. The assumption is reasonable because the sentence completion model is likely to have learned the language model for non-sarcastic text since sarcasm is an infrequent phenomenon.

It must be noted that the exact observed word where the incongruity occurs ('*ignored*' in the example above) is not known in advance. Hence, a sentence contains multiple candidate words of incongruity, out of which the incongruity is observed in case of specific word(s). We refer to these words as the '*incongruous word(s)*'. Therefore, our approaches vary in terms of the candidate incongruous words that are considered.

The novelty of this paper is as follows:

1. Using sentence completion for sarcasm detection
2. Experimentation with short text (where candidate incongruous words are a small set of words), long text (where candidate incongruous words are a large set of words), and an oracle case (where the exact incongruous word is known)

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 presents the motivation behind using sentence completion. Section 4 presents two approaches: an all-words approach, and an incongruous words-only approach. As stated earlier, the two approaches differ in terms of candidate incongruous words. Section 5 gives the experiment setup while Section 6 presents the results. We discuss an oracle case scenario in Section 7 to validate the strength of our hypothesis. Finally, we analyze the errors made by our system in Section 8 and conclude the paper in Section 9.

2 Related Work

The majority of the past work in statistical sarcasm detection has used sarcasm specific features such as punctuations, emoticons or sarcasm-indicating n-grams

[1][2][11][12][3]. For example, [1] present a semi-supervised algorithm that first extracts sarcasm-indicating n-grams and then use them as features for a classifier. [11] use features based on number of sentiment flips, positive/negative subsequences, in addition to such n-grams. [3] include features such as audience information, twitter familiarity, etc.

Recent work in sarcasm detection employs features that capture contextual information such as an author's background or their conversational context, etc. [13][14][15][16][17]. Formulations beyond classifiers have also been considered. For example, [17] use sequence labeling algorithms to predict sarcasm in individual utterances in a dialogue. On the other hand, [16] use them to predict sarcasm of the last utterance in a dialogue with automatic labels in the rest of the sequence. However, in our case, we do not use any contextual information from the author or the conversation. This means that a hyperbolic sentence such as '*X is the best President ever!*' (where the sarcasm cannot be understood based on the text alone) is beyond the scope of our approach.

In addition to the above, several rule-based techniques based on intuitive indicators of sarcasm have been reported. [6] predict a tweet as sarcastic if sentiment in the text of the tweet contradicts with the sentiment of a hashtag in the tweet. [4] predict a tweet as sarcastic if sentiment of the tweet does not match with sentiment of past tweets by the author of the tweet towards the entities in the tweet. Similarly, [5] use a set of nine rules to predict if a given simile (for example, '*as exciting as a funeral*') is sarcastic. [12] capture sarcasm as a combination of positive verbs followed by negative situation phrases. Our approach is rule-based as well.

Our work is the first to employ sentence completion for the purpose of sarcasm detection. Sentence completion approaches based on word embeddings have been reported [18][19]. However, they are only for sentence completion and not for sarcasm detection. They restrict themselves to completing sentences. We propose and validate the hypothesis that a 'language model incongruity' as experienced by a sentence completion module can be useful for sarcasm detection. We use context2vec [10] as the sentence completion library. The distinction between these sentence completion approaches is beyond the scope of this paper because the focus is to use one of them for sarcasm detection and demonstrate that it works.

3 Motivation

As stated in the previous section, in the sarcastic example '*I love being ignored*', the word '*ignored*' is observed at a position where positive sentiment words would be expected. Hence, the word '*ignored*' is the exact incongruous word. Specifically, if context2vec [10] were consulted to complete the sentence '*I love being []*' where [] indicates the position for which the most likely word is to be computed, the word '*happy*' is returned. Word2vec similarity between '*happy*' and '*ignored*' is 0.0204, for certain pre-trained word2vec embeddings. This low value of similarity between the expected and observed words can be harnessed as an indicator for sarcasm. In the rest of the paper, we refer to the word present at a given position as the '**observed word**' ('*ignored*' in the example above) where the most likely word at the position as returned by sentence completion is the '**expected word**' ('*happy*' in the example above).

However, a caveat lies in determination of the candidate incongruous words for which sentence completion will be consulted. For example, the sentence '*I could not make it big in Hollywood because my writing was not bad enough*' is sarcastic because of the incongruous word '*bad*' which is at the penultimate position in the sentence. In the absence of the knowledge of this exact incongruous word, it is obvious that an algorithm must iterate over a set of candidate incongruous words. Hence, we present two approaches: one which iterates over all words and another which restricts to a subset of words. The first approach is called the all-words approach, while the second is incongruous words-only approach. These approaches are described in detail in the next section. The '*oracle case*' for our algorithm is a situation where the incongruous word is exactly known. We validate that our algorithm holds benefit even for the oracle case, in Section 7.

4 Approach

We present two approaches that use sentence completion for sarcasm detection: (a) an "all-words" approach, and (b) "incongruous words-only" approach. As stated earlier, in the absence of the knowledge about the exact position of incongruity, our technique must iterate over multiple candidate positions. For both the approaches, the following holds:

Input: A text of length l
Output: Sarcastic/non-sarcastic
Parameters:

- Similarity measure $sim(w_i, w_k)$ returning the similarity between words w_i and w_k
- Threshold T (a real value between minimum and maximum value of $sim(w_i, w_k)$)

4.1 All-words approach

As the name suggests, this approach considers all content words[2] as candidate incongruous words. This approach is as follows:

[2] Content words are words that are not function words. We ignore function words in a sentence.

```
$min \leftarrow \infty$
for $p = 1$ to $l$ do:
        % compute expected word:
    $e_p \leftarrow context2vec(w_1, ..., w_{p-1}, [], w_{p+1}, ..., w_l)$
        % check similarity to observed word:
    if $sim(e_p, w_p) < min$ then $min \leftarrow sim(e_p, w_p)$
if $min < T$ then predict *sarcastic*
```

Thus, for the sentence '*A woman needs a man like a fish needs a bicycle*[3]' containing five content words (out of which '*needs*' occurs twice), the sentence completion library will be consulted as follows:

1. A [] needs a man like a fish needs a bicycle.
2. A woman [] a man like a fish needs a bicycle.
3. A woman needs a [] like a fish needs a bicycle.
4. A woman needs a man like a [] needs a bicycle.
5. A woman needs a man like a fish [] a bicycle.
6. A woman needs a man like a fish needs a [].

4.2 Incongruous words-only Approach

A key shortcoming of the previous approach is that it may use similarity values for words which are not incongruous, since it makes six calls in case of the example given. For example, the first part of the sentence does not contain a language model incongruity and hence, the calls are redundant. Our second approach, the Incongruous words-only approach, reduces the set of words to be checked by sentence completion to half, thereby eliminating redundant comparisons as shown in the previous subsection. Incongruous words-only approach is as follows:

```
for $p = 1$ to $l$ do:
        % compute average similarity to words:
    $\bar{s}_p \leftarrow \frac{1}{l-1} \sum_{i \neq p} sim(w_i, w_p)$
        % choose positions with lowest averages:
$Incongruous \leftarrow \{i : \bar{s}_i \leq median(\bar{s}_1, ..., \bar{s}_l)\}$
$min \leftarrow \infty$
for $p \in Incongruous$ do:
        % compute expected word:
    $e_p \leftarrow context2vec(w_1, ..., w_{p-1}, [], w_{p+1}, ..., w_l)$
        % check similarity to observed word:
    if $sim(e_p, w_p) < min$ then $min \leftarrow sim(e_p, w_p)$
if $min < T$ then predict *sarcastic*
```

As seen above, we first select the required subset of words in the sentence. Beyond that, the approach is the same as the all-words approach. As a result, for the sentence '*A woman needs a man like a fish needs a bicycle*', '*fish*', '*needs*' and '*bicycle*' are returned as most incongruous Incongruous words-only. Hence, the sentence completion is now consulted for the following input strings:

[3] http://www.phrases.org.uk/meanings/414150.html

1. A woman [] a man like a fish needs a bicycle.
2. A woman needs a man like a [] needs a bicycle.
3. A woman needs a man like a fish [] a bicycle.
4. A woman needs a man like a fish needs a [].

We hope that this reduction in the set of candidate strings increases the chances of the algorithm detecting the incongruous word and hence, the sarcasm. We observe an interesting trend in short versus long text in terms of this reduction, as will be discussed in the forthcoming sections.

5 Experiment Setup

Since our approaches are contingent on the set of candidate phrases being considered, we consider two scenarios: short text where the set of words where incongruity has likely occurred is small, and long text where the set is large. Therefore, the two datasets used for the evaluation of our approaches are: (a) Tweets by [12] (2278 total, 506 sarcastic, manually annotated), and (b) Discussion forum posts by [20] (752 sarcastic, 752 non-sarcastic, manually annotated). We ignore function words when we iterate over word positions. They are not removed because such removal would disrupt the sentence, which is undesirable since we use sentence completion. We use a list of function words available online[4].

For both approaches, we repeat the experiments over a range of threshold values, and report the best results (and the corresponding threshold values). As similarity measures, we use (a) **word2vec similarity** computed using pre-trained embeddings given by the Word2Vec tool. These embeddings were learned on the Google News corpus[5], (b) **WordNet similarity** from WordNet::similarity by [21] (specifically, Wu-Palmer Similarity). The word2vec similarity in Incongruous words-only approach is computed in the same manner as word2vec similarity above. Since word2vec similarity may not be low for antonyms, we set the similarity measure for antonyms as 0. As stated earlier, for sentence completion, we use context2vec by [10]. It is a sentence completion toolkit that uses Bidirectional LSTM to predict a missing word, given a sentence. We use the top word returned by context2vec, as per the model trained on UkWac corpus[6].

We report our evaluation for two configurations:

1. *Overall Performance*: In the first case, we run the algorithm for a range of threshold values and report results for the complete dataset.
2. *Two-fold cross-validation*: Our algorithm is dependent on the value of the threshold. Hence, we divide the dataset into two splits and repeat the experiments in two runs: estimate the optimal threshold on a split, and report results for the other, and vice versa.

[4] `http://www.ranks.nl/stopwords`
[5] `https://code.google.com/archive/p/Word2Vec/`
[6] `http://u.cs.biu.ac.il/~nlp/resources/downloads/context2vec/`

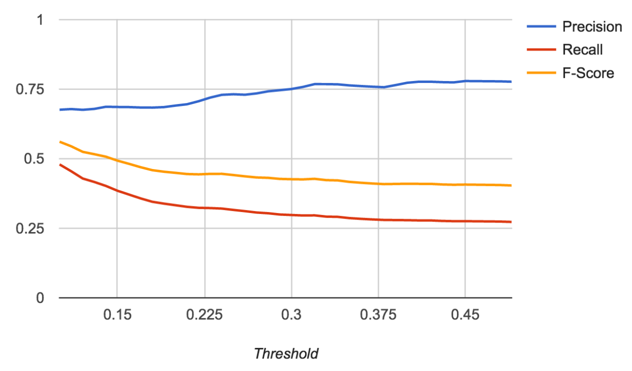

Fig. 1. Determining optimal value of threshold; Tweets, word2vec, All-words approach

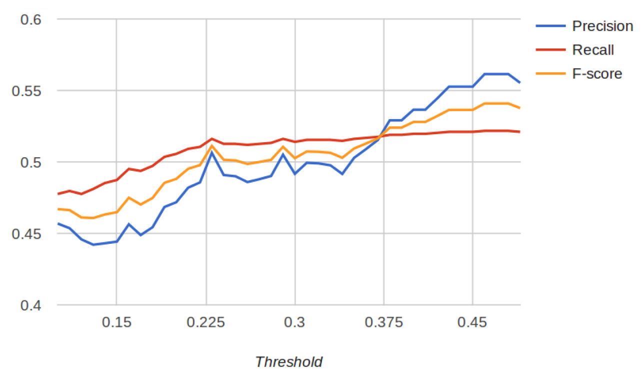

Fig. 2. Determining optimal value of threshold; Discussion Forum posts, word2vec, All-words approach

6 Results

In this section, we present an evaluation of our approaches, on the two datasets: the first consisting of short text (tweets), and the second consisting of long text (discussion forum posts). We first show the results for the complete dataset with optimal values of the threshold. We then repeat our experiments where the threshold is determined using a train-test split. These two configuations (overall performance and two-fold cross-validation) collectively validate the benefit of our approach.

6.1 Overall Performance

Table 1 shows the performance for tweets. Figure 1 shows how optimal thresholds are determined. When word2vec similarity is used for the all-words approach, an F-score of 54.48% is obtained. We outperform two past works [12,11] which have reported their values on the same dataset. The best F-score of 80.24% is obtained when WordNet is used as the similarity measure in our Incongruous words-only approach. We observe that the Incongruous words-only approach performs significantly better than the all-words approach. In case of word2vec similarity, the F-score increases by around 18%, and by around 9% in case of WordNet similarity. Also, the optimal threshold values are lower for the all-words approach as compared to the Incongruous words-only approach.

		P	**R**	**F**
Riloff et al. (2013)		62	44	51
Joshi et al. (2015)		77	51	61
Similarity	**T**	**P**	**R**	**F**
All-Words Approach				
Word2Vec	0.11	67.85	45.51	54.48
WordNet	0.11	67.68	74.84	71.08
Incongruous words-only Approach				
Word2Vec	0.42	68.00	77.64	72.50
WordNet	0.12	82.77	77.87	80.24

Table 1. Results of our approach for dataset of tweets by Riloff et al. (2013), compared with best reported values (Joshi et al. (2015) and Riloff et al. (2013)) on the same dataset

Table 2 shows the performance of our approaches for discussion forum posts, compared with past work by [11]. Note that [12] do not report performance on this dataset and are hence, not included in this table. Figure 2 shows how optimal threshold is determined. Note that similar trends are observed for other cases as well. In this case, our approaches do not perform as well as the past reported value in [11]. Also, unlike the tweets, the Incongruous words-only approach results in a degradation as compared to all-words approach, for discussion forum posts. This shows that while our approach works for short text (tweets), it does not work for long text (discussion forum posts). This is because the average length of discussion forum posts is higher than that of tweets. As a result, the all-words approach or even Incongruous words-only approach may introduce similarity comparison with irrelevant words (‘*man*’ and ‘*woman*’ in the example in Section 3).

		P	**R**	**F**
Joshi et al. (2015)		48.9	92.4	64
Similarity	**T**	**P**	**R**	**F**
All-Words Approach				
Word2Vec	0.48	56.14	52.17	54.08
WordNet	0.27	45.12	47.68	46.37
Incongruous words-only Approach				
Word2Vec	0.36	37.04	47.48	41.61
WordNet	0.15	42.69	48.18	45.27

Table 2. Results of our approach for dataset of discussion forum posts by Walker et al (2012), compared with best reported value on the same dataset

		P	**R**	**F**
Riloff et al. (2013)		62	44	51
Joshi et al. (2015)		77	51	61
Similarity Metric	**Best-T**	**P**	**R**	**F**
All words				
Word2Vec	(0.1, 0.1)	67.68	47.96	56.12
WordNet	(0.1, 0.1)	68.83	76.93	72.66
Incongruous words-only				
Word2Vec	(0.42,0.1)	63.92	77.64	70.09
WordNet	(0.14,0.12)	82.81	77.91	**80.28**

Table 3. Two-fold cross-validation performance of our approaches for the tweets dataset; Best-T values in parentheses are optimal thresholds as obtained for the two folds

		P	**R**	**F**
Joshi et al. (2015)		48.9	92.4	**64**
Similarity Metric	**Best-T**	**P**	**R**	**F**
All words				
Word2Vec	(0.48, 0.48)	56.20	52.17	54.10
WordNet	(0.37, 0.46)	43.13	48.04	45.45
Incongruous words-only				
Word2Vec	(0.19,0.25)	36.48	47.41	41.23
WordNet	(0.15,0.12)	28.34	48.04	35.33

Table 4. Two-fold cross-validation performance of our approaches for the discussion forum posts dataset; Best-T values in parentheses are optimal thresholds as obtained for the two folds

6.2 Two-fold cross-validation

Tables 3 and 4 show the two-fold cross-validation performance in case of tweets and discussion forum posts respectively. In each of the cases, past work that reports results on the same dataset is also mentioned: [12] and [11] report performance on the tweets dataset while [11] do so on the discussion forums dataset. The optimal values of threshold for the two folds are also reported since they cannot be averaged. Table 3 shows that the incongruous words-only approach outperforms past work and the all words approach. The best performance is 80.28% when incongruous words-only approach and WordNet similarity are used. Thus, in the case of tweets, our approaches perform better than past reported values.

Table 4 shows the corresponding values for the discussion forum posts. Unlike tweets, both our approaches do not perform as well as past reported values. The reported value of F-score is 64% while our approaches achieve a best F-score of 54.10%. This is likely because discussion forum posts are longer than tweets and hence, the set of candidate incongruous words is larger. This negative observation, in combination with the observation in case of tweets above, is an indicator of how the set of candidate incongruous words is a crucial parameter of the success of our approaches.

7 Discussion

Since our approaches perform well for short text like tweets but not for long text such as discussion forum posts, choosing the right set of candidate positions appears to be crucial for the success of the proposed technique. The Incongruous words-only is a step in that direction, but we observe that it is not sufficient in

Approach	**T**	**P**	**R**	**F**
All-words	0.29	55.07	55.78	55.43
Oracle	0.014	59.13	68.37	**63.42**

Table 5. Performance of the all-words approach versus the situation when the exact incongruous word is known

case of discussion forum posts. Hence, in this section, we consider an oracle case: the exact incongruous word case. This is the case where the exact incongruous word is known. Hence, we now compare our all-words approach with an '*exact incongruous word*' approach, when the exact incongruous word is known. In this case, we do not iterate over all word positions but only the position of the incongruous word. For the purpose of these experiments, we use the dataset by [22]. Their dataset consists of a word, a tweet containing the word and the sarcastic/non-sarcastic label. In case of sarcastic tweets, the word indicates the specific incongruous word. Table 5 compares the all-words approach with the only incongruous word approach. We observe that the F-score increases from 55.43% to 63.42% when the exact incongruous word is known. This shows that our approaches can be refined further to be able to zone in on a smaller set of candidate incongruous words.

It is never possible to know the exact incongruous word in a sentence. Therefore, future approaches that follow this line of work would need to work towards reducing the set of candidate incongruous words.

8 Error Analysis

Some errors made by our approaches are due to the following reasons:

1. **Absence of WordNet senses**: For a certain input sentence, the word '*cottoned*' is returned as the most likely word for a position. However, no sense corresponding to the word exists in WordNet, and so the word is ignored.
2. **Errors in sentence completion**: The sarcastic sentence '*Thank you for the input, I'll take it to heart*[7]' is incorrectly predicted as non-sarcastic. For the position where the word '*input*' is present, the expected word as returned by context2vec is '*message*'.

9 Conclusion & Future Work

This paper describes how sentence completion can be used for sarcasm detection. Using context2vec, a sentence completion toolkit, we obtain the expected word at a given position in a sentence, and compute the similarity between the observed word at that position and the expected word. Since the position of the

[7] This tweet is labeled as sarcastic in the dataset by [12]

incongruous (observed) word may not be known, we consider two approaches: (a) All-words approach in which context2vec is invoked for all content words, (b) Incongruous words-only approach where context2vec is invoked only for 50% most incongruous words. We present our experiments on two datasets: tweets and book snippets, and for two similarity measures: word2vec similarity, and WordNet similarity. Our approach outperforms past reported work for tweets but not for discussion forum posts, demonstrating that sentence completion can be used for sarcasm detection of short text. Finally, we validate the benefit of our approach for an oracle case where the exact incongruous word is known. Our approach results in a 8% higher F-score as compared to the all-words approach. Our error analysis shows that absent WordNet senses and errors in sentence completion results in errors by our approach.

Our findings set up the promise of sentence completion for sarcasm detection. This work can be extended by incorporating the current technique as a set of features for a statistical classifier. Since our approaches do not perform well for discussion forum posts, our approach must be refined to arrive at a good subset of candidate incongruous words.

References

1. O. Tsur, D. Davidov, and A. Rappoport, “Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews.” in *ICWSM*, 2010.
2. A. Reyes, P. Rosso, and T. Veale, “A multidimensional approach for detecting irony in twitter,” *Language Resources and Evaluation*, vol. 47, no. 1, pp. 239–268, 2013.
3. A. Joshi, V. Tripathi, K. Patel, P. Bhattacharyya, and M. Carman, “Are word embedding-based features for sarcasm detection?” *EMNLP*, 2016.
4. A. Khattri, A. Joshi, P. Bhattacharyya, and M. J. Carman, “Your sentiment precedes you: Using an author’s historical tweets to predict sarcasm,” in *WASSA*, 2015, p. 25.
5. T. Veale and Y. Hao, “Detecting ironic intent in creative comparisons.” in *ECAI*, vol. 215, 2010, pp. 765–770.
6. D. Maynard and M. A. Greenwood, “Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis,” in *LREC*, 2014.
7. R. W. Gibbs, *The poetics of mind: Figurative thought, language, and understanding.* Cambridge University Press, 1994.
8. S. L. Ivanko and P. M. Pexman, “Context incongruity and irony processing,” *Discourse Processes*, vol. 35, no. 3, pp. 241–279, 2003.
9. G. Zweig and C. J. Burges, “The microsoft research sentence completion challenge,” Technical Report MSR-TR-2011-129, Microsoft, Tech. Rep., 2011.
10. O. Melamud, J. Goldberger, and I. Dagan, “context2vec: Learning generic context embedding with bidirectional lstm,” in *CONLL*, 2016, pp. 51–61.
11. A. Joshi, V. Sharma, and P. Bhattacharyya, “Harnessing context incongruity for sarcasm detection,” in *ACL-IJCNLP*, vol. 2, 2015, pp. 757–762.
12. E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, “Sarcasm as contrast between a positive sentiment and negative situation.” in *EMNLP*, 2013, pp. 704–714.

13. A. Rajadesingan, R. Zafarani, and H. Liu, "Sarcasm detection on twitter: A behavioral modeling approach," in *ICWSM*. ACM, 2015, pp. 97–106.
14. B. C. Wallace, D. K. Choe, and E. Charniak, "Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment." in *ACL (1)*, 2015, pp. 1035–1044.
15. Z. Wang, Z. Wu, R. Wang, and Y. Ren, "Twitter sarcasm detection exploiting a context-based model," in *WISE*. Springer, 2015, pp. 77–91.
16. A. Joshi, V. Tripathi, P. Bhattacharyya, and M. Carman, "Harnessing sequence labeling for sarcasm detection in dialogue from tv series 'friends'," *CoNLL*, p. 146, 2016.
17. A. Silvio, B. C. Wallace, H. Lyu, and P. C. M. J. Silva, "Modelling context with user embeddings for sarcasm detection in social media," *CoNLL 2016*, p. 167, 2016.
18. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
19. Q. Liu, H. Jiang, S. Wei, Z.-H. Ling, and Y. Hu, "Learning semantic word embeddings based on ordinal knowledge constraints," in *ACL-IJCNLP*, 2015.
20. M. A. Walker, J. E. F. Tree, P. Anand, R. Abbott, and J. King, "A corpus for research on deliberation and debate." in *LREC*, 2012, pp. 812–817.
21. T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet:: Similarity: measuring the relatedness of concepts," in *Demonstration papers at HLT-NAACL 2004*. Association for Computational Linguistics, 2004, pp. 38–41.
22. D. Ghosh, W. Guo, and S. Muresan, "Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words," in *EMNLP*, 2015.