Sarcasm Relation to Time: Sarcasm Detection with Temporal Features and Deep Learning

Md Saifullah Razali (gs47497@student.upm.edu.my)
Universiti Putra Malaysia https://orcid.org/0000-0001-9918-0061
Alfian Abdul Halin
Universiti Putra Malaysia
Yang-Wai Chow
University of Wollongong
Noris Mohd Norowi
Universiti Putra Malaysia
Shyamala Doraisamy
Universiti Putra Malaysia

Research Article

Keywords: Sarcasm Detection, Natural Language Processing, Deep Learning, Sentiment Analysis

Posted Date: August 26th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1902425/v1

Sarcasm Relation to Time: Sarcasm Detection with Temporal Features and Deep Learning

Md Saifullah Razali[1,2*], Alfian Abdul Halin[1†], Yang-Wai Chow[2†], Noris Mohd Norowi[1†] and Shyamala Doraisamy[1†]

[1*]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Jalan Universiti 1, Serdang, 43400, Selangor, Malaysia.

[2]Faculty of Engineering and Information Science, University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, New South Wales, Australia.

*Corresponding author(s). E-mail(s): gs47497@student.upm.edu.my;
Contributing authors: alfian@upm.edu.my; caseyc@uow.edu.au; noris@upm.edu.my; shyamala@upm.edu.my;
[†]These authors contributed equally to this work.

Abstract

This paper discusses a framework used to detect sarcasm in relation to time. It uses a set of deep learning extracted features (deep features) combined with a set of handcrafted features. The results of the experiments are positive in terms of Accuracy, Precision, Recall and F1-measure. The combination of features is classified using a few machine learning techniques for comparison purposes. Logistic Regression is found to be the best classification algorithm for this task with an accuracy of 89%. Furthermore, result comparison to recent works and the performance of each feature set are also shown as additional information.

Keywords: Sarcasm Detection, Natural Language Processing, Deep Learning, Sentiment Analysis

1 Introduction

The vast amount of data created on social media today is good for data analysis since they are very personal [1]. Organizations have been using these data to help them understand their audience better [2]. This field is called sentiment analysis [3].

On a different note, sarcasm is defined as a positive sentence with negative intention [4]. It is regarded as one of the most challenging issues in the Natural Language Processing (NLP) field [5]. Handling sarcasm correctly is crucial since it can change the polarity of a sentence [5, 6] and jeopardize a sentiment analysis. Traditional studies [4, 7] used rule-based techniques to solve this and more recent studies [8, 9] used learning to automatically detect the most useful features.

In this paper, the deep features are the product of a deep learning architecture combined with the temporal features which are the product of manual handcrafting. Tweets are used as the main source of input. Usually, different tones and gestures are used in normal day-to-day conversations. These features are not available in writing. Hence, writers that use sarcasm tend to be more creative. They came up with semantic clues to show their sarcasm in a given sentence [4, 7]. This kind of clue is what this work is trying to manipulate.

2 Motivation

Several NLP studies have used features discovered through deep learning architectures or manual handcrafting methods [10], never both. The first group depends too much on their deep learning architectures [8, 9] and the second group on manual handcrafting methods [4, 11, 12]. This calls for specific experiments to see if the two ways can be merged and assist each other.

3 Related work

Recent works in this domain use deep learning architectures [8, 9]. following its good reputation in solving NLP problems. For example, Poria et al. [8] used four datasets to extract four deep feature sets using Convolutional Neural Network (CNN). Then these feature sets are combined and classified by an SVM classifier. Another work from Ilić et al. [13] used a deep learning model based on character-level word representations derived from the Embeddings from Language Models (ELMo). ELMo is a representation technique which use vectors derived from a bidirectional Long Short Term Memory (LSTM) [14]. Both of these works used the same dataset [15]. This is also the dataset that is used in our work.

On the other hand, most older studies use rules to produce features. For example, Barbieri et al. [16] use frequency and rarity of words as their main features. This technique is also used by Bouazizi and Ohtsuki [17] with additional rules by extracting sarcastic word patterns. The rules include counting the number of positive/negative words in the tweet and counting the number of highly emotional positive/highly emotional negative words. Another study by Shmueli et al [18] used a seed phrase "being sarcastic" as in "I was being sarcastic". The seed is then used to collect other sarcastic instances. Ultimately, a new dataset is created to help solve the issue of dataset scarcity for sarcasm detection.

One of the most prominent studies is this domain by Riloff et al. [4] also used classifiers that looks for positive verbs found with negative situations. This research has inspired other researches that use rules [8, 12, 19].

Apart from rules and deep learning, other studies depended on user historical tweets to create their features. This idea was first experimented by [12] with thorough analysis on what are the most relevant features in a thread of tweets from the same user. The features is as shown in Figure 1. All the features has their own rules. For example, audience feature use historical communication between author of the tweet and the person that tweet is intended to. This idea is inspired by the work by Kreuz & Caucci [20] which claims that sarcasm is likely to happen between the people that are familiar with each other. This technique is followed by a few other more recent studies [21–23]. For example, Rajadesingan et al. [23] created a framework called Sarcasm Classification Using a Behavioral modeling Approach (SCUBA) where they classify user's behaviour using a similar approach as done by Bamman & Smith [12](using historical tweets) but added aspects like the difference in the length of the words between the user's current tweet and their past tweets. They have also experimented on whether there is a direct correlation between the availability of historical information with sarcasm detection performance, as shown in Figure 2.

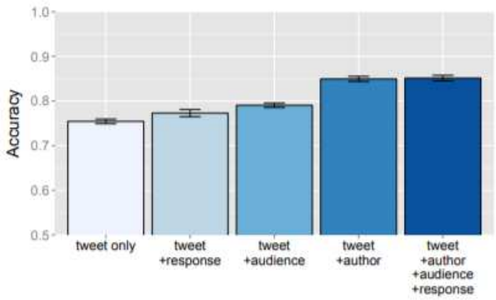

Fig. 1 Most Relevant Features for Sarcasm Detection according to Bamman & Smith[12]

In this work, the deep feature set is combined with a handcrafted set called temporal set. This second set is created after a thorough analysis on an assumption given by another work in the same domain [24]. The author of this work assumes that sarcastic sentences could be regarded as ill-meaning in a period of time but change to be good-meaning in a different period of time. The result

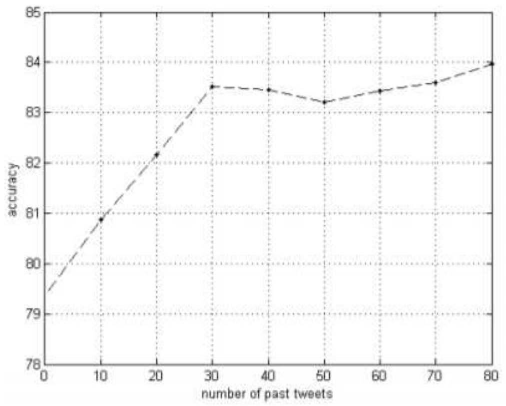

Fig. 2 Direct Correlation between Availability of Historical Information and Sarcasm Detection Performance according to Rajadesingan et al.[23]

of the experiments in this work shows that the assumption by Ebrahimi et al. [24] could be deemed right.

4 Proposed Method

This work with data acquisition and ends with evaluation. The overall framework can be seen in Figure 3.

When a person is speaking using sarcasm, they tend to use different abnormal tones [25, 26] or exaggerations [27]. In a written statement, these are translated into certain type writings. This work is focusing on the detection of such features. A temporal way of writing is one of those features. This is proven by our experiments in the later sections.

4.1 Data Acquisition

The dataset used as the input for the experiments in this paper is shared publicly [15]. It is an imbalanced dataset of 780,000 English tweets (130,000 sarcastic and 650,000 non-sarcastic). It is imbalance because the author believes that sarcastic utterances rarely happen in the real world in comparison to normal ones. The dataset is split into 80 percent for training and 20 percent for testing.

4.2 Data Preprocessing

Preprocessing is essential to any NLP study [22]. The preprocessed part of the inputs would not give any weight. In this work, five types of preprocessing techniques are used:

- Change to lowercase
- Punctuation removal

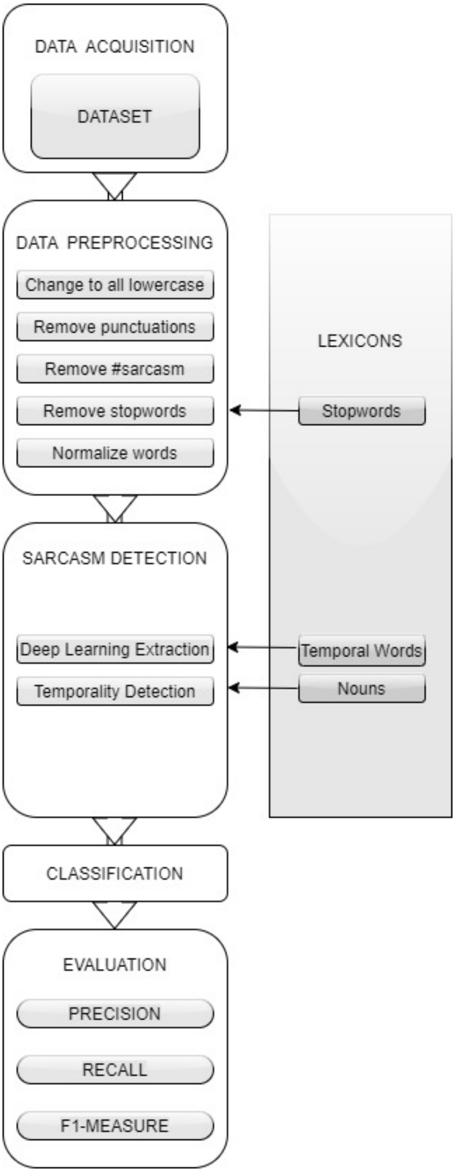

Fig. 3 Overall Framework of the Study

- "#sarcasm" removal
- Stopwords removal
- Normalize using lemmatization

4.3 Sarcasm Detection

4.3.1 Deep Learning Extraction

Many recent researches on sarcasm detection are moving towards deep learning [8, 9, 22, 28] given its high reputation in NLP. The biggest advantage of using deep learning is its ability to automatically gather optimum features for a given task [8, 9].

In this work, a vanilla Convolutional Neural Network architecture is proposed to extract ten deep features. The overall architecture of the sarcasm detection part from Figure 3 is shown in Figure 4.

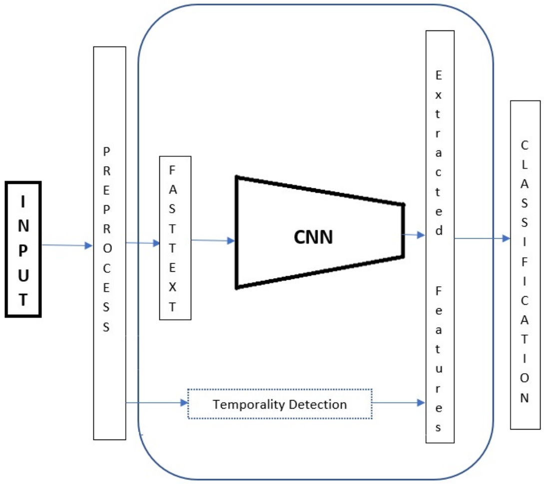

Fig. 4 Overall Architecture of Sarcasm Detection Part

As shown in Figure 4, the deep features extractor (CNN) is a large part of the overall architecture of the Sarcasm Detector. It is also assisted by a word-embedding technique as a way to convert tweet sentences into feature vectors as input. The details are described below.

4.3.2 FASTTEXT

We have decided to use FastText [29] as our word embedding technique instead of the commonly used Word2Vec [30]. This is because FastText breaks every word into N-grams instead of using individual words as with Word2Vec. Then the words are fed it into a neural network.

FastText feeds the words into a neural network in the form of unigram, bigram and trigram. For example, the word “human” is broken into “hum”, “uma” and “man” as a trigram for the word ”human”. The vectors for ”human” will be the total of all the broken N-grams. Artificial Neural Network (ANN) is used as the training method.

The output is a word-embedding vector for all the broken N-grams in the training dataset. Hence, FastText gives a better representation even for the rare and misspelled words. This makes it very effective for social networks analysis.

4.3.3 CONVOLUTIONAL NEURAL NETWORK(CNN)

For the purpose of extracting the deep features, a vanilla CNN architecture as shown in Figure 5 is used. This is the detail of the CNN part from the overall architecture of the sarcasm detection part shown in Figure 4.

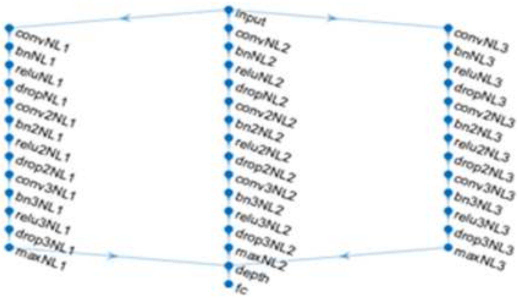

Fig. 5 CNN architecture used in the Sarcasm Detection Part

The CNN architecture in Figure 5 is shown in a top-down manner starting from the start (top) to the finish (bottom) node. "NL" stands for N-gram Length. The breakdown is:

1. An input layer of size 1 X 100 X N where N is the number of instances from the dataset. Vectors of embedded-words are used as the initial input.
2. Then the layers between the input and the concatenation is introduced:
   - One convolutional layer with 200 neurons to receive and filter size 1 X 100 X N where N is the number of instances from the dataset. The stride is [1 1].
   - Two convolutional layer with 200 neurons to receive and filter size 1 X 100 X 200. The stride is [1 1].
   - Three batch normalization with 200 channels.
   - Three ReLU activation layers.
   - Three dropout layers with 20 percent dropout.
   - A max pooling layer with stride [1 1].
3. A depth concatenation layer to concatenate all the last max pooling layers.
4. A fully connected layer with ten neurons.

In this architecture, the focus is on the convolutional layers which are used to produce the feature maps. This is followed by the batch normalization layers to improve the speed, performance, and stability by re-centring and re-scaling the input data. The activation functions are used for the scanning of input data. Finally, the minimum 0.2 dropout layers are used to avoid overfitting and increase the validation accuracy, as well as to increase the generalizing power. Then the max pooling layer is used to do the final vote. The initial inputs are the word vectors created using the Fasttext. The vector size is set to [1 100]. These vectors are then split into three groups- first group (N-Gram Length-1 or normally known as unigram), second group (N-Gram Length-2 or normally known as bigram), and third group (N-Gram Length-3 or normally known as trigram).

After the word vectors has been split into their respective N-Gram Length groups, they are fed into three graph architecture which begins after the input layer. The three graph architecture is running concurrently before combined at the concatenation layer. Then the network goes to a fully connected layer with 10 neurons, which are extracted to be our deep features. This feature set is then combined with the manual features before the classification.

4.4 TEMPORALITY DETECTION

Another quality of sarcasm is temporality [24]. For example, the sentence "You think just like Donald Trump now" in 2016 after he won the US election would have a different connotation then in 2020 when he lost the US election.

The existence of a temporal word in each tweets is taken as one feature. The checking of whether there is more than one temporal word in each tweets is another feature.Three more features are added by counting the cases where the word that comes in the vicinity of three words before or after the temporal word is a noun, depending on the distance. All of these features are in the form of Boolean. In total, 5 features are extracted using manual handcrafting. For the purpose of extracting them, the lexicons of temporal words and nouns are used. These lexicons are explained in the subsection below.

4.5 LEXICONS

All the handcrafted features are extracted using two lexicons. The lexicons are:

Temporal Words

Temporal or transition words are words that deals with time. We downloaded a list of these words from a prominent website providing it [31]. It consists of 52 instances. This lexicon is used in the process of extracting the temporal-related features.

Nouns

This lexicon is downloaded from a website that has the most comprehensive list of nouns [32]. This list contains 1500 instances. This lexicon is used in the process of extracting the features for temporal-related features.

4.6 MACHINE LEARNING CLASSIFIERS

For the purpose of comparison, five classification algorithms are used. This include: i. Support Vector Machine ii. K-Nearest Neighbor iii. Logistic Regression iv. Decision Tree v. Discriminant Analysis. All of these are well-known methods in statistics, pattern recognition and machine learning.

4.6.1 SUPPORT VECTOR MACHINE

The support vector machine (SVM) binary classification is a searching algorithm that searches for an optimal hyperplane that can partition a set of data into two classes, negative and positive [33]. In this work, we have decided to use the Radial Basis Function (RBF) as the kernel. An RBF is a function that depends only on the distance between the input and another fixed point. This has yielded the best result amongst all the other kernels for binary classification. This CNN-SVM scheme has also been proven to be quite useful for text classification with deep learning from a previous study [8].

4.6.2 K-NEAREST NEIGHBOR

K-nearest neighbor (KNN) is a classification algorithm that puts the outputs into classes by looking at the neighboring nodes[34]. This algorithm depends on a threshold "k". The default value is used for the threshold in this work; "k=10".

4.6.3 LOGISTIC REGRESSION

Same as most keyword-based models, logistic regression (LR) has two varieties: regression and classification [35]. In a binary classification model, logistic regression simply models probability of output in respond to the input given. As a binary classifier, a cutoff value is chosen and the classification is based on whether the probability of the inputs are greater than the cutoff which is going to be put in one class or below the cutoff which is going to be put in the other class For this study, the default cutoff value of 10 is used.

4.6.4 DECISION TREE

A decision tree (DT) is an algorithm that uses nodes to represent tests which are ran on an attribute. It also uses branches to represent the result of the tests and leaf nodes to represent labels or conclusions made after calculating all attributes in the experiments. Rules of the experiments are represented by the paths from the root of the tree to the leaf nodes [36].

4.6.5 DISCRIMINANT ANALYSIS

Discriminant analysis (DISCR) or also known as Linear Discriminant Analysis (LDA) models the differences between the classes of data given. It can only work when the measurements on the variables used are continuous. It can also be used when groups are known in advance. Each case must have a score on one or more quantitative measures and a score on group measures [37]. Basically, DISCR is the algorithm to group or classify instances of the same type into one group or class.

4.7 EVALUATION

Once the features are extracted, we proceed to our experiments. The metrics used to evaluate the approach are F1-measure, precision, recall and accuracy. The formulas used for F1-measure, precision, recall and accuracy are given in the equations (1), (2), (3) and (4) below respectively.

F1-measure:

$$F1 = 2.\,\frac{precision\,.\,recall}{precision + recall} \tag{1}$$

Precision:

$$precision = \frac{TP}{TP + FN} \tag{2}$$

Recall:

$$recall = \frac{TP}{TP + FN} \tag{3}$$

Accuracy:

$$accuracy = \frac{Correct\ predictions}{Total\ predictions} \tag{4}$$

A classification algorithm's effectiveness is usually measured in accuracy, which is denoted in equation (4) [38]. However, the accuracy could be very high while still not bringing any significant value to a detection algorithm. For example, a president of a nation could write one normal sentence and it could be predicted as sarcastic. This mistake might lead to unwanted or serious ramifications even though it's only one sentence. The accuracy of the system could still be truly high even though the significance or implication of the mistake would be very serious. There is no clear indication of whether the mistake is caused by the algorithm or not.

This is the reason why precision, recall, and F1-measure are used in this work. Precision is denoted in equation (2). It is basically the count of instances that are true positive from the total of true positive and false positive instances. This way, the percentage of how many of the real sarcastic instances over everything that is predicted as sarcastic by the system would be known. If the precision is low, the meaning is the system is not doing a good job in predicting real sarcastic sentences.

Recall is denoted in equation (3). It is basically the count of the instances which are true positive over the total of the instances that are true positive and false negative. In this case, the percentage of the real sarcastic instances over everything that is actually sarcastic would be known. If the recall is low, the meaning is some sarcastic sentences are labeled as normal by the system.

The final equation F1-measure is denoted in equation (1). It is used to put balance between precision and recall, especially when the dataset is imbalanced. Naturally, a sarcasm dataset would have more normal instances than sarcastic ones. This is a direct relation to the real world where human seldom use sarcasm in their conversations in comparison to the normal sentences.

The publicly shared dataset [15] used for the experiments in this work is initially split into two sets, Train and Test in the ratio of 80:20. Then the Train set is split again into Train and Validation in the ratio of 80:20. In short, the whole dataset is split into Train, Test and Validation sets in the ratio of 64:16:20. This dataset has 780,000 English tweets with 83 percent normal instances and the rest are sarcastic instances. Hence, the number of instances in the training set would be 499,200 (414,336 normal and 84,864 sarcastic), the validation set would be 124,800 (103,504 normal and 21,296 sarcastic) and the testing set would be 150,000 (124,500 normal and 25,500 sarcastic).

5 Setup

The environment used to carry out the experiments is a computer running on 64-bit Windows 10 with an Intel(R) Core(TM) i7 8th Gen with NVIDIA(R) GeForce(R) GTX. The software used is Mathworks Matlab 2019a.

6 Results and Discussion

In this section, the results produced by all the experiments is given. This is followed by the explanation of how each component in the experiments are used to fulfill the objective of the the study.

6.1 Classification Results

This subsection consist of performance results for every classification algorithm in the experiment using all the feature sets combined. It is shown in terms of F1-measure, Precision, Recall and Accuracy.

Table 1 Performance comparison for Classification Algorithms using all the Feature Sets combined

Classification Algorithm	F1	Precision	Recall	Accuracy
SVM	0.87	0.87	0.87	87%
KNN	0.86	0.86	0.86	86%
LR	**0.89**	**0.90**	**0.89**	**89%**
DT	0.87	0.87	0.87	87%
DISCR	0.87	0.87	0.86	86%

SVM and Decision Tree are both good classifiers for this task, with high accuracies and F1-scores. However, the performance of Logistic Regression are the highest. For the rest of the analysis in this paper, the results used are those from the classifier Logistic Regression.

6.2 Performance comparison with Existing Works

For comparison, two recent works that used the same dataset as in our experiment are chosen. Both of the chosen works are in the domain of sarcasm detection.The results are compiled and shown in Table 2.

Table 2 Performance comparison with Existing Works

Method	F1	Precision	Recall	Accuracy
Ilić et al.[13]	0.87	0.87	0.87	88%
Shmueli et al.[18]	0.86	0.87	**0.91**	87%
Proposed Method	**0.89**	**0.90**	0.89	**89%**

The proposed method showed significantly better performance in comparison to the others almost across all metrics used. This supports our claim that manually extracted contextual feature sets are useful for this task. The performance for each of the feature sets is also experimented.

6.3 Performance comparison among Feature Sets

This subsection consist of performance results for each of the feature set. It is shown in terms of Accuracy.

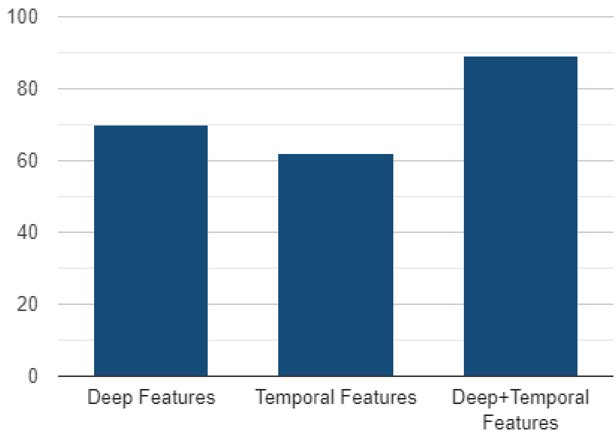

Fig. 6 Performance of Individual Feature Sets

Compared to deep feature, temporal feature shows a lower performance. According to observation, the first reason for this to happen is this type of feature has a low presence in the data set. Secondly, due to the informal language used on Twitter, temporal words are becoming hard to detect. However, an accuracy of more than 60 percent shows its importance. It can be seen clearly from the overall results that the temporal features add value to the whole framework.

7 Conclusion and Future Work

The results of our experiments provide some valuable insights into the usage of different features of tweets. The features are then exploited to build a framework that's proven useful to detect sarcasm. The method also significantly improves the F1-measure from the existing study using the same dataset. This work also demonstrates the generality of a deep learning architecture. For future work, a few datasets will be used to further generalize the comparisons. The process of extracting and gathering meaningful features could be expanded further.

Acknowledgments

This research is funded by Universiti Putra Malaysia under the grant: Geran Putra Inisiatif Siswazah (Research Title: Sarcasm Detection Using Machine Learning to Enhance Sentiment Analysis.

8 Conflict of Interest

There is no conflict of interest.

9 Data Availability

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

[1] Ghosh, A., Veale, T.: Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 482–491 (2017)

[2] Aquino, J.: Transforming social media data into predictive analytics. CRM Magazine **16**(11), 38–42 (2012)

[3] Liu, B.: Sentiment analysis and opinion mining. Synthesis lectures on human language technologies **5**(1), 1–167 (2012)

[4] Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., Huang, R.: Sarcasm as contrast between a positive sentiment and negative situation. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 704–714 (2013)

[5] Liu, B., Zhang, L.: A survey of opinion mining and sentiment analysis. In: Mining Text Data, pp. 415–463. Springer, ??? (2012)

[6] Joshi, A., Agrawal, S., Bhattacharyya, P., Carman, M.J.: Expect the unexpected: Harnessing sentence completion for sarcasm detection. In: International Conference of the Pacific Association for Computational Linguistics, pp. 275–287 (2017). Springer

[7] Davidov, D., Tsur, O., Rappoport, A.: Semi-supervised recognition of sarcasm in twitter and amazon. In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pp. 107–116 (2010)

[8] Poria, S., Cambria, E., Hazarika, D., Vij, P.: A deeper look into sarcastic tweets using deep convolutional neural networks. arXiv preprint arXiv:1610.08815 (2016)

[9] Majumder, N., Poria, S., Peng, H., Chhaya, N., Cambria, E., Gelbukh, A.: Sentiment and sarcasm classification with multitask learning. IEEE Intelligent Systems **34**(3), 38–43 (2019)

[10] Abulaish, M., Kamal, A., Zaki, M.J.: A survey of figurative language and its computational detection in online social networks. ACM Transactions on the Web (TWEB) **14**(1), 1–52 (2020)

[11] Bouazizi, M., Ohtsuki, T.: Sarcasm detection in twitter:" all your products are incredibly amazing!!!"-are they really? In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2015). IEEE

[12] Bamman, D., Smith, N.A.: Contextualized sarcasm detection on twitter. In: Ninth International AAAI Conference on Web and Social Media (2015). Citeseer

[13] Ilić, S., Marrese-Taylor, E., Balazs, J.A., Matsuo, Y.: Deep contextualized word representations for detecting sarcasm and irony. arXiv preprint arXiv:1809.09795 (2018)

[14] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)

[15] Ptáček, T., Habernal, I., Hong, J.: Sarcasm detection on czech and english

twitter. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 213–223 (2014)

[16] Barbieri, F., Saggion, H., Ronzano, F.: Modelling sarcasm in twitter, a novel approach. In: Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 50–58 (2014)

[17] Bouazizi, M., Ohtsuki, T.O.: A pattern-based approach for sarcasm detection on twitter. IEEE Access **4**, 5477–5488 (2016)

[18] Shmueli, B., Ku, L.-W., Ray, S.: Reactive supervision: A new method for collecting sarcasm data. arXiv preprint arXiv:2009.13080 (2020)

[19] Joshi, A., Bhattacharyya, P., Carman, M.J.: Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR) **50**(5), 1–22 (2017)

[20] Kreuz, R., Caucci, G.: Lexical influences on the perception of sarcasm. In: Proceedings of the Workshop on Computational Approaches to Figurative Language, pp. 1–4 (2007)

[21] Amir, S., Wallace, B.C., Lyu, H., Silva, P.C.M.J.: Modelling context with user embeddings for sarcasm detection in social media. arXiv preprint arXiv:1607.00976 (2016)

[22] Ghosh, A., Veale, T.: Fracking sarcasm using neural network. In: Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 161–169 (2016)

[23] Rajadesingan, A., Zafarani, R., Liu, H.: Sarcasm detection on twitter: A behavioral modeling approach. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, pp. 97–106 (2015)

[24] Ebrahimi, M., Yazdavar, A.H., Sheth, A.: Challenges of sentiment analysis for dynamic events. IEEE Intelligent Systems **32**(5), 70–75 (2017)

[25] Attardo, S., Eisterhold, J., Hay, J., Poggi, I.: Multimodal markers of irony and sarcasm. Humor **16**(2), 243–260 (2003)

[26] Rockwell, P.: Vocal features of conversational sarcasm: A comparison of methods. Journal of psycholinguistic research **36**(5), 361–369 (2007)

[27] Kunneman, F., Liebrecht, C., Van Mulken, M., Van den Bosch, A.: Signaling sarcasm: From hyperbole to hashtag. Information Processing & Management **51**(4), 500–509 (2015)

[28] Zhang, M., Zhang, Y., Fu, G.: Tweet sarcasm detection using deep neural network. In: Proceedings of COLING 2016, The 26th International Conference on Computational Linguistics: Technical Papers, pp. 2449–2460 (2016)

[29] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)

[30] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[31] Temporal Words. https://grammar.yourdictionary.com/style-and-usage/\list-transition-words.html. Accessed: 2021-02-21

[32] Noun. https://www.talkenglish.com/vocabulary/top-1500-nouns.aspx. Accessed: 2021-02-21

[33] Vapnik, V., Izmailov, R.: Knowledge transfer in svm and neural networks. Annals of Mathematics and Artificial Intelligence **81**(1), 3–19 (2017)

[34] Peterson, L.E.: K-nearest neighbor. Scholarpedia **4**(2), 1883 (2009)

[35] Christmann, A.: Least median of weighted squares in logistic regression with large strata. Biometrika **81**(2), 413–417 (1994)

[36] Kamiński, B., Jakubczyk, M., Szufel, P.: A framework for sensitivity analysis of decision trees. Central European journal of operations research **26**(1), 135–159 (2018)

[37] Garson, G.D.: Discriminant function analysis. statnotes: topics in multivariate analysis. Retrieved March **29**, 2010 (2008)

[38] Huang, J., Ling, C.X.: Using auc and accuracy in evaluating learning algorithms. IEEE Transactions on knowledge and Data Engineering **17**(3), 299–310 (2005)