# Sarcasm Detection Using Deep Learning-Based Techniques

**Niladri Chatterjee, Tanya Aggarwal and Rishabh Maheshwari**

**Abstract** Sarcasm is a figure of speech in which the speaker says something that is outwardly unpleasant with an intention of insulting or deriding the hearer and/or a third person. Designing a model for successfully detecting sarcasm has been one of the most challenging task in the field of natural language processing (NLP) because sarcasm detection is heavily dependent on the context of the utterance/statement and sometimes, even human beings are not able to detect the underlying sarcasm in the utterance. In this chapter, we design features for detecting sarcasm using pragmatic features that take into account the context of the utterance. The approach is based on a linguistic model that describes how humans distinguish between different types of untruths. We then train various machine-learning-based classifiers and compare their accuracies.

**Keywords** Sarcasm detection · Natural language processing · Classifiers

## 1 Introduction

Sentiment analysis forms a crucial part for various natural language processing tasks such as movie reviews, product recommendations. Sentiment analysis is very intricately intertwined with detection of sarcasm. While doing sentiment analysis, it is of immense importance that the model is able to detect sarcastic sentences as they carry a sentiment which is opposite to the surface sentiment. For illustration, there are some sentences like "I love solving math problems all day", which might be sarcastic for one person while non-sarcastic for other. Thus, sarcasm detection is greatly influenced by the context in which an utterance is made and the difficulty

N. Chatterjee · T. Aggarwal · R. Maheshwari (✉)
Indian Institute of Technology, New Delhi, India
e-mail: Rishabh.Maheshwari.mt614@maths.iitd.ac.in

N. Chatterjee
e-mail: Niladri@maths.iitd.ac.in

T. Aggarwal
e-mail: Tanya.mt614@maths.iitd.ac.in

B. Agarwal et al. (eds.), *Deep Learning-Based Approaches for Sentiment Analysis*, Algorithms for Intelligent Systems,
https://doi.org/10.1007/978-981-15-1216-2_9

associated in capturing the context of the utterance is something that adds to the challenge associated with sarcasm detection.

In terms of definition, sarcasm is defined as a form of verbal irony that is intended to express contempt or ridicule, i.e. Sarcasm has an implied negative (generally) sentiment but may not have a negative surface sentiment. For example, the sentence "I love being ignored" has a positive surface sentiment but has a negative implied sentiment (thereby creating an incongruity) and hence is sarcastic. There are three important parts to the definition of sarcasm:

1. Sarcasm is a form of irony
2. It is mostly intended by the speaker and hence is not just an interpretation of the listener
3. It is used to express contempt or ridicule.

Any model that is designed to solve the problem of sentiment analysis must have some mechanism to differentiate between the sarcastic and non-sarcastic sentences, because if present and not identified correctly, sarcasm can completely change the underlying meaning of the sentence and the way in which it is comprehended. Sarcasm detection is greatly influenced by the context in which an utterance is made and the difficulty associated in capturing the context of the utterance is something that adds to the challenge associated with sarcasm detection. For illustration, there are some sentences like "I love cooking", there are different scenarios possible (assuming the conversation is taking place between two people):

1. If the listener knows the actual liking of the speaker towards cooking:
    a. If the speaker actually likes cooking, it will be perceived as non-sarcastic by the listener
    b. If the speaker does not actually like cooking, it will be sarcastic for the listener
2. If the listener has no knowledge of the liking of the speaker towards cooking, he/she might perceive as either sarcastic or non-sarcastic.

Thus, a same sentence can be perceived as sarcastic by one person and non-sarcastic by some other person, thereby making the context of the utterance extremely crucial to detect sarcasm.

Camp [1] classifies sarcasm into four categories:

1. ***Propositional Sarcasm***: The sarcastic sentences that fall in this category would appear as ordinary propositions on the surface but they have a negative implied sentiment associated with them. For example, if you do not like a plan made by your friends and you say "*This plan sounds fantastic*". Again, it must be noted that if we just look at this sentence, we would perceive that the sentence has a positive sentiment associated with it, we need to know the context and the manner in which the person saying this sentence says it to know that it is actually a sarcastic remark.
2. ***Embedded Sarcasm***: In this types of sarcastic sentences, there is an incongruity in the sentence, that is, there would be positive phrases (or words) that are immediately followed by phrases (or words) that carry a negative sentiment and vice

versa. This type of sarcasm can generally be identified by checking if there is any incongruity present in the sentence or not. For example, "*I love being ignored*". Here, the word "love" carries a positive sentiment and it is immediately followed by the phrase "being ignored" which has a negative sentiment associated with it and hence,an incongruity is generated.

3. ***Like-prefixed Sarcasm***: As the name of this category of sarcasms suggest, these are preceded by the "*Like*", which provides an implied denial of the argument being made. For example, "*Like you care*" is a common sarcastic retort.
4. ***Illocutionary Sarcasm***: The sarcastic sentences that fall in this category would appear as non-sarcastic if we look only for the textual clues. Their sarcastic nature is attributed to some non-textual clues, like the body language, tone, gestures, etc., of the speaker that indicate an attitude opposite to that of a sincere utterance. For example, rolling one's eyes while saying "*Yeah right!*" The "*rolling of eyes*" is a gesture that indicates that the speaker does not literally means the statement he/she is saying and is being sarcastic.

The importance of detecting sarcasm correctly can be illustrated through the following examples:

1. Twitter is one of the places where the use of sarcasm is quite prevalent. When we try to do any sentiment analysis task on twitter data, our first task should be to segregate sarcastic and non-sarcastic tweets and then detect the sentiment. Some of the applications require very accurate sentiment analysis, predicting stock market behaviour using twitter sentiment analysis, being one of them. An inaccurate prediction by the model can lead to huge losses.
2. When we are dealing with product reviews on Amazon to find a rating of the product, many times the consumer writes sarcastic remarks. For example, a product on Flipkart has the following review "*Supercool..i just sold my 2nd kidney to buy this after i bought iphone6 s..now, i m in ventilation. Feels satisfied having this*". On the surface this review seems to be a positive one because of the presence of the words like "supercool", "satisfied", etc., but a human being on reading the entire sentence clearly gets to know that it is a sarcastic remark and hence should be considered as a negative review instead of positive one. An algorithm for review mining should be able to do the same kind of interpretation which is possible only if it is correctly able to identify sarcastic remarks.
3. Similar to the previous example, suppose that we are trying to summarize multiple reviews about a hotel or a movie. We must be able to identify the sarcastic ones as they can change the entire sentiment of the summary we generate. Consider the following two reviews about a same hotel:

    a. "*Very friendly service, continental breakfast was excellent (JUICEBOXES!) and the room was great. Very clean and the haunted sink and screaming toilet gave the bathroom personality!*"
    b. "*We spent 2 nights but received no hskg service. Carpet was filthy. Drapes were torn and dirty. Faucet was broken. Plumbing was noisy, especially at night when we were trying to sleep*".

Now, when a summarizer would try to give a combined review (summary) of the two reviews, it needs to correctly interpret the sarcasm in the first review to give an overall negative sentiment (in this case) to the summary, otherwise, it might take the first review as positive one and second as negative and get confused.

There are several other real-life natural language processing applications that require sarcasm to be detected correctly. We design some features of our model based on violations of Grice's Maxims which will be discussed in detail in the sections to follow. In our work, along with the lexical-, pragmatic- and polarity-based features, we devise four new features based on violations of Grice's Maxim of quality. Grice suggested that any conversation is based on shared principle of cooperation which describes how effective communication can be achieved in any conversation. He fleshed out the principle in a series of maxims. There are eight violations to the maxim of quality: Lies, White Lies, Hyperbole, Meiosis, Sarcasm, Euphemism, Metaphor and Paradox.

Based on these violations, we constructed four new features to detect sarcastic sentences as follows:

1. Overtness—How overt or obviously untrue a sentence is
2. Acceptability—Social acceptability of the sentence with help of number of unacceptable words
3. Exaggeration—Exaggeration in the sentence by evaluating intensity of words
4. Comparison—Similarity between the compared objects (if any) in the sentence using Wu–Palmer similarity [2] on Word-net.

The former two try to capture the semantic sense of a sentence while the latter two capture the implicit incongruity which is between the surface sentiment and the implied sentiment as in the example mentioned above. Mathematical formulas have been used to compute the above features from given text. Thus, the above-mentioned features have continuous values in their ranges. This allows us to use them in a machine-learning-based framework that we developed.

We train different machine-learning classifiers (random forest classifier, gradient boosted trees and SVM) as they are better than rules-based classifications. We worked on the Twitter dataset. For all semantic scoring purposes (positive intensity and negative intensity), we use Valence Aware Dictionary and sEntiment Reasoner (VADER) lexicon. The results show the effectiveness of ML algorithms in differentiating sarcastic and non-sarcastic statements.

## 2 Related Work

Sarcasm is one of the interesting subjects of language and has proven same for natural language processing as well as a perpetual challenge. There have been several approaches to this problem, from primitive rule-based, which needs close study of pattern of sarcastic sentences and its components to modern deep learning techniques, which need close study to create features to get a good detection model.

Tsur et al. [3] used semi-supervised learning for sarcasm detection. They used syntactic as well as pattern-based features. They gave labels to a sentence from 1 to 5, 5 being clear presence of sarcasm and 1 being absence.

They defined content words (CW) and high-frequency words (HFW). Words below some threshold were called content words, while words above some different threshold were called high-frequency words. Proper nouns were also considered high-frequency words. The patterns they used were which looked at a fixed-sized window of words in a sentence and then look at content words and high-frequency words, ordered sequence of high-frequency words and slots for content words.

For e.g. "I love waking up early in the morning" if considered a window of 2 CW or HFW, then the patterns will be as follows: "I CW CW up", "up CW in CW", "in the CW" and all other of such types.

Therefore, each sentence can have more than one pattern. Each observation (seed) was converted to feature vector, with each pattern having an entry in the feature vector. And this entry would be between 0 and 1 according to match (exact, sparse or incomplete) found in the sentence with the corresponding pattern. Punctuation-based features were also present in this vector, to represent length and frequency of different punctuation marks in the sentence.

For labelling test set, they used a k-nearest neighbour like algorithm. Euclidean distance to $k$ matching vectors to a test observation $t$ was calculated, where matching vectors are those observations which share at least one pattern feature with $t$. Then, the label is weighted average of the labels of the $k$ vectors, weights being the frequency of label same as that of the vector.

Although the method has interestingly incorporated the use of patterns for sarcasm detection, it does not use sentiment analysis, which intuitively plays a larger part in sarcasm detection.

Riloff et al. [4] proposed a very interesting method to use sentiment analysis and syntactic analysis. They defined sarcastic sentence to have a positive sentiment followed by negative situation, which is intuitively true (although they do not consider the negative sentiment with positive situation). Then, they developed a bootstrap algorithm to learn negative phrases.

For e.g. if "I love exams" is sarcastic, then "love" is a positive verb phrase, and "exams" will then become negative phrase (trigram).

First, they decided a "seed" word or initial positive sentiment verb phrase, then in each sarcastic sentence that contains this word, they looked at immediate following n-grams (unigrams, bigrams or trigrams), because due to brevity of sarcastic tweets (which comprised their dataset), they assumed simple sentence structure, and considered these as negative situation phrases candidates. They further pruned these by parts-of-speech (POS) tagging the phrases, and considered only those which matched their manually developed structure for the n-grams. Further, negative phrase candidates were only added to the final list, if the conditional probability of the sentence being sarcastic, given the negative situation phrase comes after a positive sentiment verb phrase, is more than a threshold.

For e.g. If seed word is “love” and next we come across “We eagerly wait for exams”, then as “exams” is a negative phrase, “eagerly wait” becomes positive verb phrase.

This generated a list of negative phrase candidates. Then, learning was done in the reverse direction, to learn positive verb phrases or positive predicate phrases using analogous conditional probability and corresponding threshold.

Therefore, if a sentence will contain one of the positive sentiment phrase and negative situation phrase, it will be predicted as sarcasm. This method looks not only at syntactic, but also sentimental behaviour of sarcasm, and defines a way to find negative situation phrases. But due to this, it depends a lot upon the versatility of training set in lexical sense also, rather than just syntactical.

Joshi et al. [5] use the same algorithm to look for implicit incongruity in a sentence, where implicit refers to the fact that negative situation is implied and not so apparent.

Knowledge of phrases implying negative sentiment was needed.

For e.g. “He loves this pant so much that he rarely wears it”, here “rarely wears it” is a phrase with negative implicit implication.

Instead of using a rule-based approach, they created feature vector for each sentence (tweets) and fed them to machine-learning model. The presence of implicit incongruity was then used as a feature for learning model.

Presence of explicit incongruity was also used as a feature, where the incongruity can easily be detected by sentiment analysis.

For e.g. “I love bitter food”, “bitter” is not known to be preferred taste.

Lexical features like bigram, unigram were used to contain properties of semantics.

As tweets were involved, pragmatic features like capitalized letters, punctuation marks, emojis’ frequency and type were also used as they signify sentiment of the user too.

They specified in the paper that this method was based on world knowledge and may overlook individual-specific sarcasm.

For e.g. “I love solving maths problems” may not be sarcastic for some individuals.

Zhang et al. [6] suggested that neural networks would be better at performing, because of automatic feature induction, rather than manual feature feeding. That is, using embedding to represent words in a sentence and developing feature vector using simple feature templates, like, representing word as concatenation of word embeddings of a word before it, the word itself and a word after it, can then be used by the neural model to automatically gain contextual as well as other sentimental insights.

For modelling, they used gated recurrent neural network, which does not forget the context as well as not carry all of the historical data. For e.g. long-short-term-memory (LSTM) is a GRNN.

They developed feature of current tweet and also historical 80 tweets, using word embedding and feature templates (as mentioned before) and fed it to an LSTM.

Poria et al. [2017] use convolutional neural network to extract the features from a sarcastic sentence rather than handmade features because they thought convolution network will capture context better and may also learn the hierarchical structure if any would be present.

Their model can then be divided into 4 parts,

1. **Word embedding model**: They used word2vec embedding to represent a word and therefore a sentence by concatenating *n* words present in a sentence, *n* being the length of longest sentence. This vector is then fed to a convolution network till a fully connected layer (explained further after description of models) for feature extraction.
2. **Sentiment feature extraction**: They used a pre-trained CNN model for feature extraction of 100 dimensional vectors. The training was done on a benchmark dataset for sentiments, classifying sentences into positive, negative or neutral sentiments in the final layer.
3. **Emotion feature extraction**: This too was done using a CNN model trained on dataset to classify emotion of a sentence into six categories, namely anger, disgust, surprise, sadness, joy and fear. The feature vector obtained was 150 dimensional from fully connected layer of the model.
4. **Personality Feature extraction**: CNN models are used to extract features or traits for each personality, which are openness, conscientiousness, extraversion, agreeableness and neuroticism. So, for each personality, there is a CNN model, with each giving a feature vector or trait vector of 150 dimensions, which then are concatenated to form a 750 dimensional personality feature vector.

All these feature vectors are then concatenated, word embedding model's feature vector till the fully connected layer, and the features extracted from the other three models, and then fed into a CNN with softmax output layer or SVM classifier for sarcastic/non-sarcastic classification.

Hazarika et al. [7] also used Stylometric features which contain the information about author's writing style based on gender, age, diction, syntactic influence, etc., along with the features mentioned above.

## 3 Grice's Maxims

In simple words, Grice Maxims are a set of properties which when present in any kind of conversation can make it more meaningful and logical. Whenever we engage in any kind of vocal conversation, the things that we speak are often progressive remarks of related things. We usually do not make comments that are disconnected from what the conversation was about. To put it differently, one can say that any kind of ***effective conversation*** between people is a result of cooperative efforts of each participant who also is well aware of the purpose of the conversation. This can be labelled as the "***Cooperative Principle***". This was introduced by Paul Grice [8] in his pragmatic theory as:

*'Make your contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged'*

Grice further suggested that this principal can be divided into four Maxims, which are popularly known as ***Grice's Maxims*** and these are:

1. Maxim of quantity
2. Maxim of quality
3. Maxim of relation
4. Maxim of manner

***Maxim of Quantity***: This maxim specifies the amount of information that a participant involved in the conversation should convey. So, according to this maxim:

1. Make your contribution as informative as is required (for the current purposes of the exchange).
2. Do not make your contribution more informative than is required.

***Maxim of Quality***: This maxim states how one can make a good quality contribution to the conversation. According to this maxim

1. Be truthful
2. Do not say what you believe is false
3. Do not say what you lack adequate evidence for.

***Maxim of Relation***: This maxim states that the contribution that one makes must ***be relevant*** to the topic of discussion going on

***Maxim of Manner***: The above three maxims focus only on the contribution made by a participant in terms of content. However, in any vocal conversation between people, it becomes of immense importance that how an utterance is made. This is precisely what this maxim lists, a set of points that dictate how we should make our contribution. We should

1. Be clear
2. Avoid difficult expressions
3. Avoid ambiguity
4. Be brief
5. Be orderly.

There can be different types of violations of the Grice's Maxims. For the purpose of this chapter, we would focus in particular on the violations of maxim of quality. Grice suggested that different violations of these maxims give rise to various figures of speech in discourse. Nair [7] proposes a model that describes how conversationalists across cultures differentiate systematically between different types of violations of the maxim of quality. There are eight different types of violations of the maxim of quality:

1. ***Lie***: A lie is an outright untruth which is made in self interest. They tend to be harmful, malicious and to betray a person. For example, when you have stolen your friend's favourite watch and upon being asked you say that you have not seen it or do not know where it is.

2. ***White Lie***: These are the lies that are spoken out of kindness and with the intention of not hurting anyone's feelings. They are selfless and harmless. For example, when you do not like a particular dish made by your mother but you do not want to hurt her feelings and say that the dish is tasty.
3. ***Paradox***: A paradox is a statement that appears to be self-contradictory but has a hidden meaning attached to it. For example, Truth is honey, which is bitter.
4. ***Metaphor***: A metaphor is an implicit comparison between two things that are not related to each other. To put it in other words, a metaphor draws a resemblance between two contradictory objects based on some shared characteristics. For example, Her voice is music to his ears. This statement is a metaphor because her voice is not literally music but it makes him feel happy and hence is compared to music.
5. ***Meiosis***: Meiosis is an understatement. Thus, this figure of speech implies that something is less significant or small than it actually is. For example, "Don't worry, I'm fine. It's only a scratch" when you are actually experiencing pain from the injury.
6. ***Hyperbole***: Hyperbole is an overstatement. It is basically an exaggerated or extravagant statement which is not meant to be taken literally. For example, "My grandmother is as old as the hills".
7. ***Sarcasm***: In sarcasm, the underlying meaning is completely opposite to the surface meaning. For example, "I love being ignored".
8. ***Euphemism***: Euphemism is a figure of speech, in which the speaker uses polite expressions instead of words or phrases that might otherwise be considered harsh or unpleasant. For example, saying "passed away" instead of "died".

Since all the above figures of speech involve some kind of misinterpretation of the reality (truth), they are violations to the maxim of quality. Having discussed the different types of violations to the maxim of quality, we need to find some way so that we can differentiate between them.

The above different types of violations can be distinguished from one another based on four features of ***overtness***, ***comparison***, ***exaggeration*** and ***acceptability***. These features can be defined as:

1. ***Overtness***: It is a measure of how obvious the untruth is, that is, how promptly can we identify the semantic or pragmatic violations of the literal truth
2. ***Comparison***: As the name suggests, this features measures if there are two objects being compared in the statement being made. In case of violations of the maxim of quality, mostly we will find instances where two very different lexical items (words or phrases) are compared with each other (This in turn leads to the violation)
3. ***Exaggeration***: This feature measures if the sentence that we are making, contains words that represent the entire situation or topic of conversation as better or worse than it actually is, hence leading to a violation of the literal truth.
4. ***Acceptability***: It is a measure of how socially acceptable a statement is irrespective of the fact that whether it has been recognized by the participants in the conversation as a violation of the maxim of quality or not.

To get a better understanding of the above features, it is important to have an idea about how the hearer would know that a violation of the maxim of quality by the speaker. In other words, what triggers the hearer's beliefs that some kind of untruth has been uttered by the speaker. Again consider the sentence "I love cooking", a hearer can interpret this sentence as truth or untruth depending on the knowledge that he has about the speaker. Nair [7] explains that the mutual knowledge shared between the speaker and the hearer can be of two types:

1. ***Pragma-linguistic knowledge***: This is related to the content of the sentence spoken by the speaker. It includes the entailments, presuppositions, etc., of the utterance along with the linguistic and lexical rules and the pragmatic norms.
2. ***Encyclopedic knowledge***: This is related to the context of the utterance and background information of the utterance.

We term a shared knowledge between the speaker and hearer as 'mutual' if there is a match (assumed or created) between them. Since, any conversation have the possibility of becoming a part of the mutual knowledge, they have an equal probability of creating a mismatch between the knowledge of speaker and hearer, thereby creating a conflict and hence leading to an interpretation of the statement as untruth.

On the basis of the type mismatch with respect to the two types of mutual information mentioned above, one can think of two types of violations as explained below:

1. If there is a violation of the pragma-linguistic knowledge, then an obvious (overt) violation is made by the speaker. Also, it must be noted that when such kind of violation is made, the speaker has no intention of misleading the hearer as he believes that he has made an obvious violation which would straight away be identified by the hearer as an obvious untruth. Generally, the literary violations (hyperbole, metaphor, paradox, meiosis, sarcasm and euphemism) fall in this kind of violation.
2. In case of the violations of encyclopedic knowledge, it must be noted that the speaker has an intention of misleading the hearer. Such kind of violations is not obvious to identify (and hence are not overt). Lies and white lies fall in this category.

Now, coming to the feature of exaggeration, consider a set of sentences:

a. John is the worst cook in the world
b. John is a bad cook
c. John is not the best cook in the world.

The above examples, show a clear distinction in fact that how exaggerated is the fact that 'John is a bad cook'. Sentence a) is an overstatement (+ exaggeration) whereas sentence b) is an understatement (− exaggeration). Now again, if the hearer knows that the speaker has a limited experience and he makes a statement with phrases like "worst/best in the world", the hearer would immediately (thereby overt) know that this is an untruth.

The feature of comparison mainly just checks if there are two things that are being compared, irrespective of the fact that the two things being compared are actually similar, + comparison (metaphor) or not, − comparison (paradox).

The feature of acceptability, as the name suggests, checks if the statement would appear to be offensive to the hearer (− acceptability) or not (+ acceptability). That is, is the statement socially acceptable or not.

To exert the fact that these four features are enough to distinguish the eight kinds of violations from each other, we present a table with example sentences and corresponding symbols:

1. +: necessarily a criteria requirement of the violation
2. o: not necessarily a criteria requirement of the violation
3. −: necessarily not a criteria requirement of the violation.

Table 1 also helps to compare how similar and different the eight kinds of violations of the maxim of quality are to each other.

## 4 Challenges in Sarcasm Detection

The main reason why it is difficult to design a model for detecting sarcasm is that at times even human beings are not able to detect sarcastic sentences.

The main challenge faced while doing sarcasm detection is capturing the context of the utterance. As an example, let's say we are detecting sarcasm for twitter data (tweets), our model should be able to find the appropriate context in which the utterance is made so that it can go about identify if it is sarcastic or not. Another thing that can be done is we can look at the kind of tweets that person has tweeted already, this would give us an idea about the manner in which the person writes tweets, that is whether he usually makes sarcastic remarks or not. If he/she usually makes sarcastic remarks then there are high chances that the tweet in consideration would also be sarcastic and vice versa. But again, our model should be able to identify the number of past tweets it will consider to get this detail. Also, some sarcastic sentences are sarcastic not because of their textual content but because of the manner in which they are spoken, which again is something which is difficult to deal with. In a nutshell, the challenges associated with sarcasm detection are:

1. Capturing the appropriate context of the utterance. Context in itself is a very vague term. There is no exact definition of what exactly is the context of the utterance. Some utterances would require the model to look at a very small context to say that it is sarcastic, whereas for others we might have to look at a very large context.
2. Capturing the tone and body language of the person who makes the statement, because at times the sarcastic nature of the sentence is attributed not to its content but to the manner in which it is spoken.

**Table 1** Violations of Grice's Maxims

| Violation | Example | Overtness | Exaggeration | Comparison | Acceptability |
|---|---|---|---|---|---|
| Lie | When you have stolen your friend's favourite watch and upon being asked you say you have not seen it or don't know where it is | – | o | o | – |
| White Lie | When you do not like a particular dish made by your mother but you do not want to hurt her feelings and say that the dish is tasty | – | o | o | + |
| Paradox | Truth is honey, which is bitter | + | o | – | O |
| Metaphor | Her voice is music to his ears | + | o | + | O |
| Meiosis | "Don't worry, I'm fine. It's only a scratch" when you are actually experiencing pain from the injury | + | – | o | O |
| Hyperbole | My grandmother is as old as the hills | + | + | o | O |
| Sarcasm | "I love being ignored" | + | o | o | – |
| Euphemism | "His father 'passed away'". instead of "His father is 'dead'" | + | o | o | + |

## 5 Dataset Description

We used tweets to test our model. As tweets typically are not more than 140 characters mostly they consist of one or two sentences. Moreover, for tweets, as they are devoid of external factors such as body language, voice modulation, facial expression, text is the only means for conveying sentiment. We used the twitter API provided for obtaining tweets containing certain strings, we used *#sarcasm* to collect sarcastic tweets, and collected around 6K sarcastic tweets, out of which 3K were non-sarcastic (general tweets). This set did not include re-tweets for so that set is as versatile as possible. We employed tweepy library to collect tweets, although it only gave tweets from 2 weeks back and would not let tweets be collected after a certain amount, as a security measure.

For historical tweet extraction:

1. We generated a list of users whose tweets were present in the corpus. There were around 4K unique users.
2. Then, we extracted the historical tweets using username as search key (maximum five for each user) and saved their sentiment (using VADER sentiment analysis), calculated the average and stored the average and number of tweets in a dictionary.
3. When came across a tweet, we first checked if historical tweets were present, then overtness was calculated using formula mentioned in next section, the new average was then calculated using following formula:

$$S_{\text{new}} = S_{\text{old}} * n_{\text{old}} + S_{\text{new}}$$
$$n_{\text{new}} = n_{\text{old}} + 1$$

4. Storing the new values in the dictionary, the whole algorithm is then repeated for all tweets in the 6K tweets.

After collecting the tweets, preprocessing was done as following:

1. Other hashtags, URLs (links) and mentions were removed, (mentions were replaced by nouns).
2. Added space before and after punctuation marks for better word and sentence detection.
3. Replaced contractions (e.g. don't) with their expansion using dictionary available, this was to improve tokenization and get better sentiment analysis.
4. Replaced social media slangs and abbreviations with their full forms (e.g. lol, tbh, btw, etc.)
5. Spelling check was then done on all of the words after performing the above steps.

Following were the shortcomings of the dataset:

1. Tweepy: Gave error 429 (frequency of twitter extraction is too high) Tweepy does not give data more than two weeks old.
2. With time, people used more and more hashtags and less words. Sometimes Mentions are used as nouns (twitter handles) or adjectives (e.g. #awesome).
3. Sometimes, people may not add '#sarcasm' and sometimes, may add unnecessarily
4. Also, our model requires tweets to be extracted in one go (that is, we cannot combine the tweets extracted from weeks that are not consecutive using tweepy

(which allows a maximum of two weeks tweets to be extracted only)) as the score of overtness cannot be calculated if there are no previous tweets from some person

The corpus still may contain a lot of noise, like incorrect tagging of a sentence as sarcastic or absence of the hashtag while the sentence might be sarcastic, absence of historical data of a user, or all of the historical tweets by the user being sarcastic, therefore misleading the overtness feature.

## 6 Feature Description

Our model makes use of the following features:

The features are inspired from Detection of sarcasm in tweets: a rough set based approach by Bajpai et al. [9]

a. ***Overtness***: As described earlier, overtness is a measure of how obvious the lie (untruth) is. So, this feature is a measure of how much overt or covert the statement is. If we are dealing with twitter data, we quantify this score by comparing the average sentiment of the tweets the same user has made in the past with the tweet that we are considering. The reason that we quantify this feature like this is because let us say a user is habitual of making positive (or negative) sentiment tweets and he suddenly makes a negative (or positive) tweet, then there are high chances that he is being sarcastic. To calculate the sentiment score of the tweet, we have made use of the NLTK library package and used the VADER implementation in it. Mathematically, the score is given by:

$$d = s - s_o$$

Here,

$d$ is the overtness score
$s$ is the sentiment of the tweet in consideration
$s_o$ is the average sentiment of the previous tweets by the same user

A nice observation that can be made here is that both $s$ and $s_o$ lie between the range $[-1, 1]$, where $-1$ signifies totally negative sentiment and $+1$ indicates totally positive sentiment. Hence, $d$ lies in the range $[-2, 2]$

b. ***Exaggeration***: As described earlier, exaggeration is a measure of how exaggerated (understated or overstated) the given statement or tweet is as compared to reality. To quantify this feature, we first observed and studied various exaggerated sentences and we came to a conclusion that adverbs, adjectives and verbs are the only figures of speech that would make a contribution to the exaggeration of a sentence or tweet. So, the next logical step that followed was to identify these figures of speech from the entire sentence or tweet. This was done with the help of POS-tagging using NLTK's library implementation of Penn Treebank tagging.

(POS-tagging assigns parts-of-speech tags to the constituent words of a sentence or tweet). Now, once we have identified the figures of speech of the constituent words of a sentence, we identified another problem, there are words in English like *"bass"* which can have different meaning depending upon the context of the sentence in which they are used. Following are the different meanings of the word "*bass*":

1. a type of fish
2. tones of low frequency
3. A type of instrument.

It is quite evident that depending upon the meaning of the word in the actual sentence can have an impact on the sentiment score of the sentence. There are many algorithms that help in word-sense disambiguation (capturing the actual meaning of the word out of its different meanings). One such algorithm is the Lesk's algorithm [10] for word-sense disambiguation. It was introduced by Michael E. Lesk in 1986. The basic idea of the Lesk's algorithm is that we can find the actual sense of a word by looking at the overlap between the Dictionary definitions of the word and the context (neighbourhood) of the word in the sentence. The Lesk algorithm is:

(i) for every meaning (sense) of the word, count the number of words in its context (neighbourhood) that are present both in the dictionary meaning as well as the neighbourhood of the word
(ii) the algorithm returns that meaning (sense) of the word which gets the maximum overlap of words with the neighbourhood of the word

So, for word-sense disambiguation, we use an implementation of Lesk's algorithm in NLTK library. Once we are done with this, we go about finding the sentiment scores of the constituent words of the sentence using Sentiwordnet [11]. Mathematically,

$$S_c = p_s + n_s$$

where

| | |
|---|---|
| $S_c$ | is the sentiment score (S-score) of the word |
| $p_s$ | is the positive sentiment score associated with the word |
| $n_s$ | is the negative sentiment score associated with the word. |

When we provide a word as an input to the SentiWordNet, we get three scores associated with the word, namely, positivity score (the positive sentiment associated with the word), negativity score (the negative sentiment associated with the word) and an objectivity score (the neutral or objective sentiment associated with the word). All the three scores lie in the range [0, 1] and sum to 1. Hence, a straightforward observation that can be made here is that $S_c \leq 1$.

Now, we also observed that many a times, sentences or tweets contain degree modifiers, which are words like "*very*", "*quite*", etc., which basically increase or decrease the intensity of the word that follows it. Thus, there are two types of degree modifiers:

(a) ***Degree Intensifying Modifiers***: These include words like "*very*", "*greatly*", etc., which increase the intensity of the words that follow them. When we provide degree intensifier words as an input to the SentiWordNet, we get $p_s > n_s$

(b) ***Degree De-intensifying Modifiers***: These include words like "*rarely*", "*barely*", etc. which decrease the intensity of the words that follow them. When we provide degree de-intensifier words as an input to the SentiWordNet, we get $p_s < n_s$

Now, we had earlier observed that the sentiment score of the word $S_c$ is less than 1. Therefore, we include the effect of degree modifiers in our model as follows: (Here, assume that $w$ is the word that follows the degree modifier $d$)

$$S_c(\omega) = \begin{array}{ll} \sqrt{S_c(\omega)} & \text{if } a \text{ is a degree intensifier} \\ S_c(\omega) & \text{if } p_s(d) = n_s(d) \\ S_c(\omega)^2 & \text{if } d \text{ is a degree de - intensifier} \end{array}$$

The exaggeration score of the sentence or tweet is the average of the scores of all the constituent words in the tweet, that is

$$e = \left(\sum_{i=1}^{n} S_c(w_i)\right)/n$$

where, $n$ is the total number of words in the sentence

(c) ***Acceptability***: Acceptability is a measure of socially acceptable a statement or tweet is. The most logical way of quantifying this feature is measuring the number of acceptable or unacceptable words. To calculate the number of unacceptable words, we make use of a slang dictionary and perform string matching with the words of the dictionary to get the count. Then, mathematically the acceptability score is given by:

$$a = 1 - \frac{n_a}{n}$$

Here, $n_a$ is the number of unacceptable words and $n$ is the total number of words in the sentence or tweet. The lexicon of negative words was available online [12]

d. ***Comparison***: This score measures if there is a comparison being made in the sentence or tweet (either similar or dissimilar objects are being compared). To mathematically quantify this score, we find the similarity score between the words being compared in the tweet. Now, comparison is a very frequent in sarcastic tweets or sentences; hence, comparison is an important score. The first step to find this score is to find the words being compared (these words can be adjectives, nouns, verbs, etc.). This is done using context-free grammar (CFG) rules to parse phrases of a sentence or tweet as follows:

S ⇒ NP “like” NP | ADJ “as” “NP” | ADJ “as” ADJ | ADJ “as” V | NP “like” V
NP ⇒ N | ADJ N | N N | “NNS” N
N ⇒ “NNP” | “NN”
V ⇒ “VBD” | “VB” | “VBG”
ADJ ⇒ “JJ” | “RB” | “RBR” | “VBG”

An important point to be noted here is that the above CFG is the one that we used in our work. It is not necessarily exhaustive. All the POS tags not present in the CFG will not be used in any further step to calculate this score and hence will be removed. An interesting observation from the above CFG is that a focus phrase cannot be of a length greater than 5. Now, once we have extracted the two words, next step is found out their similarity. This is calculated using Wu–Palmer similarity which returns the similarity score of two word senses based on the depth of the two senses and their least common subsumer (or least common ancestor).

Mathematically, the Wu–Palmer similarity is given by:

$$\text{sim}_{wup} = \frac{2 * \text{depth}(\text{lcs}(w_1, w_2))}{\text{depth}(w_1) + \text{depth}(w_{2)}}$$

Here, $w_1$ and $w_2$ are the words being compared and *lcs(*$w_1$*,* $w_2$*)* is the least common subsumer of them.

The similarity score is calculated using the Wu–Palmer Similarity measure implementation available in NKTK package of python after applying Lesk’s algorithm for word-sense disambiguation.

Two cases arise here,

**Case 1** If an adjective or adverb is found before and after *“as”* or “*like”*, then similarity score between the words in target is calculated.

Eg. “He is as active as snoring kid”

**Case 2** If an adjective or adverb is found before “*as*” or “*like*”, then that adjective is compared to all the words (maximum 2) following “*as*” or “*like*” and the similarity score is calculated of all words on the one side to the adjective or adverb on the other side.

Eg. “Alice is fast like a snail”

**Case 3** If the phrase that we extract using context-free grammar has no adjective, then all the words coming before “*as*” or *“like*” are considered being compared to all that come after “*as*” or “*like*” and

Eg. “My boss is as human as a Neanderthal”

The final comparison score of the sentence or tweet is given by:

$$w = \left(\sum_p WP\text{sim}(p)\right)/n$$

Here, $p$ is the number of word pairs found, $n$ is the total number of words, $WP\text{sim}(p)$ is the Wu–Palmer similarity score for the pair $p$.

Apart from the above-mentioned four features, as discussed by Joshi et al., we also take consideration some other features which are based on incongruity and some lexical features, which are as follows:

1. ***Explicit Incongruity-Based Features***: As has been discussed in the previous sections, incongruity arises when two words or phrases of opposite sentiment occur together in a sentence or tweet. This feature is a measure of inherent incongruity in a sentence or tweet. We quantified this feature with a help of a number of sub features which individually capture things observed when an incongruity occurs. These are:

    a. ***Positive Word Count***: This is the overall count of words carrying a positive sentiment in the sentence or tweet in consideration. The sentiment of the words is identified using SentiWordNet.
    b. ***Negative Word Count***: This is the overall count of words carrying a negative sentiment in the sentence or tweet in consideration. The sentiment of the words is identified using SentiWordNet.
    c. ***Number of contextual incongruities***: This is perhaps the most important sub feature to capture explicit incongruity. As the definition of explicit incongruity suggests that incongruity arises when words (or phrases) of contrasting sentiments appear together in the sentence. This feature measures exactly the same thing. It is a count of how many times a positive sentiment word is followed by a negative sentiment word and vice versa. In short, it is count of sentiment switches occurring in the sentence or tweet in consideration.
    d. ***Longest sequence of positive or negative sentiment of words***: This feature measures the length of the longest sequence of words in the sentence that carry the same sentiment (the sentiment can be positive or negative).
    e. ***Overall sentiment of the sentence***: This is the overall sentiment of the sentence. It is calculated using VADER.

2. ***Lexical Features***: Theses are a set of features that capture important information about the structure of the focus sentence. It includes:

    a. ***Tf-Idf Values***: We also provide as input to our model, the Tf-Idf matrix of unigrams, bigrams and trigrams. The Tf-Idf value is a score that assigns importance to words based on how frequently they appear. For the purpose of our work, we used only top 3000 most frequent unigrams, bigrams and trigrams. The number of frequent words considered can be changed depending upon individual requirements.
    b. ***Capitalized Word Count***: As the name suggests, this is the count of words that are capitalized. We take this feature into consideration because it was observed that people tend to use capitalized words in tweets when they are being sarcastic to lay extra emphasis on the word that is capitalized.
    c. ***Number of smileys and emojis***: This is the count of number of smileys or emojis being used in the tweet.

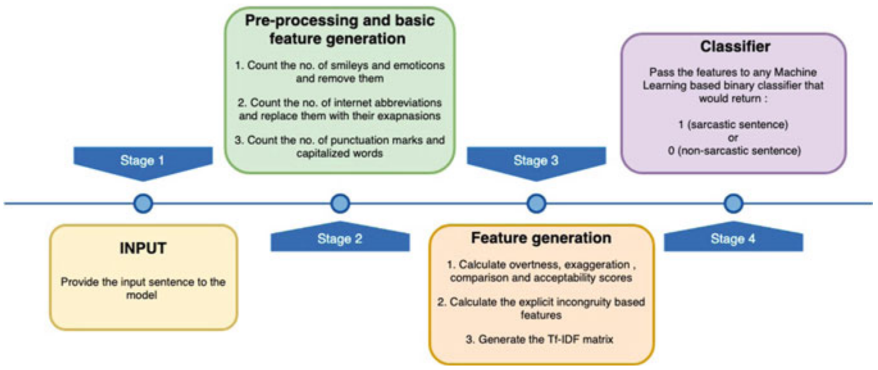
Pre-processing and basic feature generation
1. Count the no. of smileys and emoticons and remove them
2. Count the no. of internet abbreviations and replace them with their exapnasions
3. Count the no. of punctuation marks and capitalized words
Classifier
Pass the features to any Machine Learning based binary classifier that would return :
1 (sarcastic sentence)
or
0 (non-sarcastic sentence)
Stage 1
Stage 2
Stage 3
Stage 4
INPUT
Provide the input sentence to the model
Feature generation
1. Calculate overtness, exaggeration , comparison and acceptability scores
2. Calculate the explicit incongruity based features
3. Generate the Tf-IDF matrix

**Fig. 1** Process outline

d. ***Number of Internet abbreviations***: Nowadays, people tend to use a lot of abbreviations instead of writing the entire text on social media. These abbreviations usually add to the nature of the sentence being considered. We used a dictionary of common Internet abbreviations and their full forms to get this count.
e. ***Number of punctuation marks***: This is the count of punctuation marks (‘,’, ‘!’, ‘?’*)* in the focus tweet.

## 7 Process Outline

Figure 1 shows the process outline.

## 8 Models Used

We trained a number of machine-learning models so that we could compare them and draw further inferences. The models that we used are:

1. ***Decision Trees***: Decision trees are flow-chart like structure where at each node we make a test on some particular feature and depending upon the result, we go to the child node corresponding to the result.
2. ***Random Forest Classifier***: This fits a number of decision trees of given height on various subsets of input dataset and uses averaging to decrease over-fitting and increase accuracy.
3. ***Support Vector Machine***: Classifier which tries to learn the separating hyperplane between the instances by using support vectors instead of whole dataset, and may also mimic higher dimension by using kernel-based inner products.

4. ***Gradient Boosted Trees:*** Gradient boosting algorithms are those which generate ensemble of weak classifiers in such a way that each model generated next moves towards decreasing the error, i.e. moves towards opposite direction of gradient.

## 9 Experiments and Results

We obtained the following ROC curves for the different machine-learning models that we experimented with (using all the features mentioned previously as input to the model):

Some insights from the graph are:

1. Avg. AUC in 5 split cross validation = 0.90833
2. Avg. AUC in 10 split cross validation = 0.9316

This tells, if we have more data, the machine-learning model can perform even better (Fig. 2).

Table 2 compares the accuracies of various models having a different combination of features. (both F-score and AUC are used for comparison purposes):

Table 2 shows the comparison of results using different combinations of features.

While Model 3 has better Precision and F-score, Model 5 has better Recall and AUC. This tells that Model 3 was better at telling which is sarcastic, Model 5 is better at telling which is not.

Note: All the results in the above table are of random forest classifier as it has the highest score among the trained models.

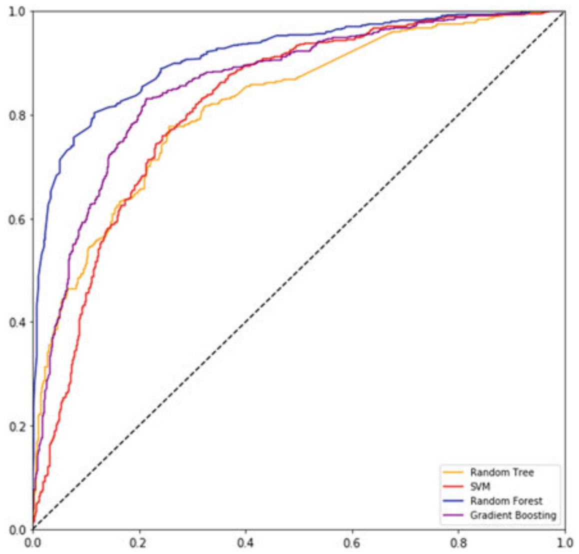

**Fig. 2** AUC curve of different methods, x-axis is false positive and y-axis is true positive

**Table 2** Here, *L* lexical features, *EI* explicit incongruity-based features, *A* acceptability, *E* exaggeration, *C* comparison, *O* overtness

| # | Features | Precision | Recall | F-score | AUC |
|---|---|---|---|---|---|
| 1 | L + EI | 0.815154 | 0.883084 | 0.847761 | 0.902518 |
| 2 | L + EI + A | 0.836104 | 0.875621 | 0.855407 | 0.905609 |
| 3 | L + EI + A + E | **0.883963** | 0.843283 | **0.863144** | 0.9117731 |
| 4 | L + EI + A + E + C | 0.855361 | 0.853233 | 0.854296 | 0.912210 |
| 5 | L + EI + A + E + C + O | 0.820895 | **0.889303** | 0.853731 | **0.914726** |

## 10 Future Scope

1. Joshi et al. also mentions use of another feature, namely, implicit incongruity which can be added to the set of features used in the model.
2. A more rule-based algorithm can be used and its results can be compared with the regression-based models used.
3. The sentiment of the emojis and smileys used in the text can also be considered in the set of features used.
4. More historical data can be collected to calculate value of overtness.
5. More grammatical rules can be used to calculate the value of comparison and exaggeration.

## References

1. Camp, Elisabeth. 2012. Sarcasm, pretense, and the semantics/pragmatics distinction. *Noûs* 46 (4): 587–634.
2. Wu, Z., and M. Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguist*ics, 133–138.
3. Davidov, Dmitry, Oren Tsur and Ari Rappoport. 2010. Semi-supervised recognition of sarcasm in twitter and amazon. In *CoNLL.*
4. Riloff, Ellen, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert and Ruihong Huang. Sarcasm as contrast between a positive sentiment and negative situation.
5. Joshi, Aditya, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (Volume 2: Short Papers), Vol. 2.
6. Zhang, Meishan, Yue Zhang and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *COLING.*
7. Nair, R.B. 1986. Telling lies: some literary and other violations of Grice's Maxim of quality. *Nottingham Linguistic Circular (Special Issue on Pragmatics)* 14: 53–71.
8. Grice, H.Paul. 1957. Meaning. *Philosophical Review* 66 (3): 377–388.
9. Bajpai, Anurag, Vaibhav Khandelwal and Niladri Chatterjee. Detection of sarcasm in tweets: a rough set based approach.

10. Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC'86: Proceedings of the 5th Annual International Conference on SYSTEMS Documentation, ACM*, 24–26, New York, USA.
11. Esuli, Andrea, and Fabrizio Sebastiani. 2006. Sentiwordnet: a publicly available lexical resource for opinion mining. In *LREC*, vol. 6.
12. https://github.com/shekhargulati/sentiment-analysis-python/blob/master/opinion-lexicon-English/negative-words.txt.