Approaches for Computational Sarcasm Detection: A Survey

Lakshya Kumar, Arpan Somani and **Pushpak Bhattacharyya**
Dept. of Computer Science and Engineering
Indian Institute of Technology, Powai
Mumbai, Maharashtra, India
{lakshyakri09,as16somani,pushpakbh}@gmail.com

Abstract

Sentiment Analysis deals not only with the positive and negative sentiment detection in the text but it also considers the prevalence and challenges of sarcasm in sentiment-bearing text. Automatic Sarcasm detection deals with the detection of sarcasm in text. In the recent years, work in sarcasm detection gains popularity and has wide applicability in sentiment analysis. This paper complies the various approaches that are developed to tackle the problem of sarcasm detection. In this paper, we describe Rule-based, Machine Learning and Deep Learning approaches for detecting sarcasm and also describes various datasets. We also give details of different features used by various sarcasm detection approaches from past upto the present.

1 Introduction

In this paper, we examine some representative work in the area of sarcasm detection. Various approaches to detect sarcasm in tweets and other textual data are described in the following sections along their respective limitations. This gives a sense of the work that has been published in the field. It describes about the various past works by illustrating the different types of techniques that has been used by these approaches. This chapter further divides the past works on the basis of: rule-based, machine-learning based and deep-learning based approaches. We look at these approaches in the following sections.

The organization of the paper is as follows. The paper starts with Section 2, which examines some of the initial work in sarcasm detection which used rule-based method to classify sarcastic text. Then Section 3 follows, which gives a detailed description of the various statistical approaches for sarcasm detection and also provides a comparative study of different types of features that have been used by these approaches. After that, finally Section 4 touches upon a few deep learning approaches for sarcasm classification and also describes the architecture used by these approaches. Finally, we conclude the paper in Section 5

2 Rule-Based Approaches

Rule-based approaches attempt to identify sarcasm through specific evidences. These evidences are captured in terms of rules that rely on indicators of sarcasm.

(Veale and Hao, 2010) focus on identifying whether a given simile (of the form '* as a *) is intended to be sarcastic. They use Google search in order to determine how likely a simile is. They present a 9-step approach where at each step/rule, a simile is validated using the number of search results. A strength of this approach is that they present an error analysis corresponding to multiple rules. They considered ironic similes with explicit grounds, of the form "as Ground as a Vehicle". Using the wildcarded query "as * as a *" on a search engine like Google reveals that the internet is awash with instances of this basic simile pattern, such as "as strong as an ox", "as cool as a cucumber" and "as dead as a doornail". To classify as Ground as a Vehicle , they follow following 9-step sequence:

- A simile is classified as non-ironic if there is lexical/morphological similarity between vehicle and ground word.
- If the web frequency of "about as Ground as a Vehicle" is more than half that of "as Ground as a Vehicle " then the simile is classified as ironic and noted as an ironic precedent.

- If this simile is recognizable as a direct variation of an ironic precedent (in bullet 2 above), then this simile is also classified as ironic.

- If this simile is recognizable as an inverse variation of an ironic precedent (in bullet 2 above), then this simile is inversely classified as non-ironic.

- If the ad-hoc category pattern "Ground * such as Vehicle " is found on the web, then the simile is considered non-ironic and is noted as a non-ironic precedent.

- If the simile is a direct variation of a non-ironic precedent, it is deemed non-ironic.

- If the simile is an inverse variation of a non-ironic precedent, it is deemed ironic.

- If the simile has a web-frequency of 10 or more, it is classified as non-ironic and is also noted as a non-ironic precedent.

- If the simile has a web-frequency less than 10, it is classified as ironic.

(Maynard and Greenwood, 2014) propose that hashtag sentiment is a key indicator of sarcasm. Hashtags are often used by tweet authors to highlight sarcasm, and hence, if the sentiment expressed by a hashtag does not agree with rest of the tweet, the tweet is predicted as sarcastic. They use a hashtag tokenizer to split hashtags made of concatenated words.

They have developed a set of rules which attempt to detect sarcasm information from hashtags:

- If there is a single hashtag denoting sarcasm, and the original sentiment is positive or neutral, we flip the polarity to negative.

- If there is more than one hashtag, we look at any sentiment contained in those hashtags.

- If two hashtags both contain sarcasm indicators, we treat them as one single sarcasm indicator, e.g. "#lying #notreally"

- If a positive hashtag is followed by a sarcasm indicator, and the polarity of the tweet is positive or neutral, we flip the polarity of the sentiment of the positive hashtag to negative, and then apply this sentiment to the text (flipping the polarity of the tweet from positive or neutral to negative), e.g. "Heading to the dentist. #great #notreally"

- If a negative hashtag is followed by a sarcasm indicator, and the polarity of the tweet is positive or neutral, we treat both hashtags as negative and flip the polarity of the tweet to negative.

(Bharti et al., 2015) present two rule-based classifiers. The first uses a parsebased lexicon generation algorithm that creates parse trees of sentences and identifies situation phrases that bear sentiment. If a negative phrase occurs in a positive sentence, it is predicted as sarcastic. The second algorithm aims to capture hyperboles by using interjection and intensifiers occur together. The system model proposed in their paper is given as:

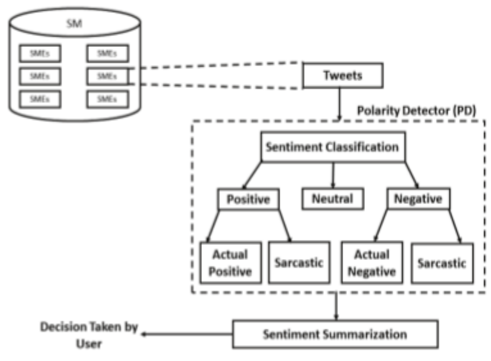

Figure 1: System model for decision making from users opinions. (Bharti et al., 2015).

In the Figure 1, there are three model entities described as :

- **Social media (SM)**: It is a social networking website (Facebook, Twitter, Amazon, etc.), where people use to write an opinion, reviews, and post some blogs, or micro-blogs about any social media entities.

- **Social media entities (SME)**: An entity about which, users want to post and retrieve comments, tweets and give reviews on social media.

- **Polarity detector (PD)**: An automated system which is capable to identify the actual sentiment or sarcasm sentiment from text.

(Riloff et al., 2013) present rule-based classifiers that look for a positive verb and a negative situation phrase in a sentence. The set of negative situation phrases are extracted using a well-structured, iterative algorithm that begins with a bootstrapped set of positive verbs and iteratively expands both the sets (positive verbs and negative situation phrases) as shown in the Figure 2. They experiment with different configurations of rules such as restricting the order of the verb and situation phrase. Their approach learns rich phrasal lexicons of positive sentiments and negative situations using only the seed word love and a collection of sarcastic tweets as input. They assume that the sarcasm probably arises from positive/negative contrast and exploit syntactic structure to extract phrases that are likely to have contrasting polarity. Another key factor is that they focus specifically on tweets. The short nature of tweets limits the search space for the source of the sarcasm. The brevity of tweets also probably contributes to the prevalence of this relatively compact form of sarcasm. The learning process relies on an assumption that a positive sentiment verb phrase usually appears to the left of a negative situation phrase and in close proximity (usually, but not always, adjacent). Pictorially, we assume that many sarcastic tweets contain this structure:
$[+VERBPHRASE][SITUATIONPHRASE]$

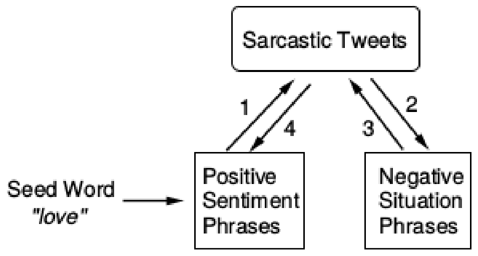

Figure 2: Bootstrapped Learning of Positive Sentiment and Negative Situation Phrases (Riloff et al., 2013).

The above structural assumption drives their bootstrapping algorithm, which is illustrated in Figure 2. The bootstrapping process begins with a single seed word, love, which seems to be the most common positive sentiment term in sarcastic tweets. Given a sarcastic tweet containing the word love, our structural assumption infers that love is probably followed by an expression that refers to a negative situation. So we harvest the n-grams that follow the word love as negative situation candidates. We select the best candidates using a scoring metric, and add them to a list of negative situation phrases. Next, we exploit the structural assumption in the opposite direction. Given a sarcastic tweet that contains a negative situation phrase, we infer that the negative situation phrase is preceded by a positive sentiment. We harvest the n-grams that precede the negative situation phrases as positive sentiment candidates, score and select the best candidates, and add them to a list of positive sentiment phrases. The bootstrapping process then iterates, alternately learning more positive sentiment phrases and more negative situation phrases.

3 Statistical Approaches

Statistical approaches to sarcasm detection vary in terms of capturing different features and learning procedures. We describe various statistical approaches for sarcasm detection in this subsection.

(Tsur et al., 2010) design pattern-based features that indicate presence of discriminative patterns as extracted from a large sarcasm-labeled corpus. To allow generalized patterns to be spotted by the classifiers, these pattern-based features take real values based on three situations: exact match, partial overlap and no match. Pattern-based features serve as their main contribution, for pattern extraction they followed the algorithm given by (Davidov and Rappoport, 2006). They have classified words into high-frequency words (HFWs) and content words (CWs). A word whose corpus frequency is more (less) than F_H (F_C) is considered to be a HFW (CW).

For example, given a sentence "Garmin apparently does not care much about product quality or customer support", we have generated several patterns including "[company] CW does not CW much", "does not CW much about CW CW or", "not CW much and about CW CW or CW CW.". Note that "[company]" and "." were treated as high frequency words.

Pattern matching: Once patterns were selected, they have used each pattern to construct a single entry in the feature vectors. For each sentence they have calculated feature value for each pattern as below:

- **1**: Exact match all the pattern components appear in the sentence in correct order without any additional words.

- α: Sparse match same as exact match but additional non-matching words can be inserted between pattern components.

- $\gamma * n/N$: Incomplete match only $n > 1$ of N pattern components that appear in the sentence, while some non-matching words can be inserted in-between. At least one of the appearing components should be a HFW.

- **0**: No match nothing or only a single pattern component appears in the sentence.

After extracting various pattern-based and punctuation based features they have used k-Nearest Neighbors *(k-NN)* like strategy to classify test data.

(Liebrecht et al., 2013) tackled the problem of sarcasm detection in twitter dataset by introducing bi-gram and tri-gram based features. They have designed and implemented a sarcasm detector that marks unseen tweets as being sarcastic or not. They analyze the predictive performance of the classifier by testing its capacity on test tweets that are explicitly marked with the hashtag #sarcasme (Dutch for sarcasm), left out during testing, and its capacity to rank likely sarcastic tweets that do not have the #sarcasme mark. They also provide a qualitative linguistic analysis of the features that the classifier thinks are the most discriminative. In a further qualitative analysis of sarcastic tweets in the test set they find that the use of an explicit hashtag marking sarcasm occurs relatively often without other indicators of sarcasm such as intensifiers or exclamations.

While their classifier performance gave an impression of its ability to distinguish sarcastic tweets, the strong indicators of sarcasm as discovered by the classifier may provide additional insight into the usage of sarcasm by Twitter users: in particular, the typical targets of sarcasm, and the different linguistic markers that were used. They further analyzed the feature weights assigned by the Balanced Winnow classifier ranked by the strength of their connection to the sarcasm label, taking into account the 500 words and n-grams with the highest positive weight towards the sarcasm class. These words and n-grams provide insight into the topics Twitter users are talking about: their targets.

(Buschmeier et al., 2014) incorporate ellipsis, hyperbole and imbalance in their set of features. They model the task of irony detection as a supervised classification problem in which a review is categorized as being ironic or non-ironic. They further investigate different classifiers and focus on the impact analysis of different features by investigating what effect their elimination has on the performance of the approach.

(Ptácek et al., 2014) use word-shape and pointedness features given in the form of 24 classes. They have presented the first attempt at sarcasm detection in the Czech language, in which the main focus was on supervised machine learning approaches and to evaluate their performance. They have selected various n-grams, including unigrams, bigrams, trigrams with frequency greater than three ((Liebrecht et al., 2013)), and a set of language-independent features, including punctuation marks, emoticons, quotes, capitalized words, character n-grams and skip-grams ((Reyes et al., 2012)) as their baseline.

For evaluation they have used the most promising language-independent features from the related work and POS-related features.

Group	**Features**
N-gram	Character n-gram
	N-gram
	Skip-bigram
Pattern	Pattern
	Word-shape pattern
POS	POS characteristics
	POS Word-shape
	POS n-gram
Others	Emoticons
	Punctuation-based
	Pointedness
	Extended Pointedness
	Word-case

Table 1: Various features used by (Ptácek et al., 2014) for Sarcasm Detection

The novel contributions of their work include the extensive evaluation of two classifiers with various combinations of feature sets shown in Table 1 on both the Czech and English datasets as well as a comparison of different preprocessing techniques for the Czech dataset.

(Liu et al., 2014) introduce POS sequences and semantic imbalance as features. Since they also experiment with Chinese datasets, they use language-typical features like use of homophony, use of honorifics, etc. Sarcasm is a pervasive linguistic phenomenon in online documents that express subjective and deeply-felt opinions.

They do not use explicit features to detect sarcasm and ignore the imbalance between sarcastic and non-sarcastic samples in real applications. They explore the characteristics of both English and Chinese sarcastic sentences and introduce a set of features specifically for detecting sarcasm in social media. Then, they propose a novel multi-strategy ensemble learning approach **(MSELA)** to handle the imbalance problem. They have shown the evaluation of their proposed model on English and Chinese data sets. Experimental results show that their ensemble approach outperforms the state-of-the-art sarcasm detection approaches and popular imbalanced classification methods.

3.1 Sarcasm Detection using author's history

For sarcasm detection sometimes we need to go beyond the text itself, as in case of *I absolutely love this restaurant!* which may be sarcastic, depending on the contextual situation or as per the user's perception. (Khattri et al., 2015) presented a novel approach for sarcasm detection that uses two components:

- **Contrast-based predictor** That identifies if there is a sentiment contrast within a target tweet.

- **Historical tweet-based predictor** That identifies if the sentiment expressed towards an entity in the target tweet agrees with sentiment expressed by the author towards that entity in the past.

Their whole architecture has three components described as follows:

1. **Contrast-based predictor** uses only the target tweet and identifies its sarcastic nature.

    - Using context incongruity based on (Joshi et al., 2015a).

2. **Historical tweet-based predictor** uses target tweet, historic tweet and name of author. The goal here is to identify whether sentiment of historic and target tweets are opposite. Using following steps:

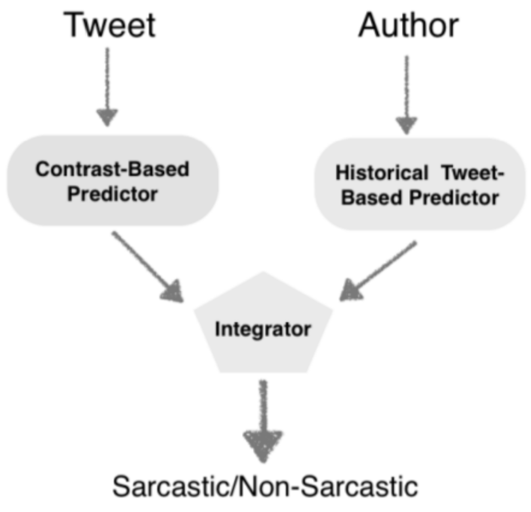

Figure 3: Architecture of our sarcasm detection approach.(Khattri et al., 2015).

    (a) Sentiment of target tweet is analyzed using rule-based approach
    (b) POS-tag the target tweet and mark NNP sequences as *target phrases*.
    (c) Download historic tweets that containing target phrases.
    (d) Mark sentiment of all historic tweets and use MV for authors opinion towards target phrase.
    (e) Finally, predict target tweet as sarcastic if historical sentiment is different from target tweets sentiment.

3. **Integrator Module** This module integrates the prediction of historical tweet-based & contrast-based predictor. Following are various variants of it:

    - Using **only historical tweet-based prediction** i.e. mark tweet as sarcastic only on previous information, if no information is available then non-sarcastic.
    - **OR** Mark tweet as sarcastic even if one of two module says it is.
    - **AND** Mark tweet as sarcastic if predictions of both the module matches.
    - **Relaxed-AND**
        – Mark sarcastic if both matches.
        – If no historical information, then consider output of contrast-based predictor.

3.2 Other learning based methods for Sarcasm Detection

A variety of classifiers have been experimented for sarcasm detection. Most work in sarcasm detection relies on SVM [(Joshi et al., 2015a); (Tepperman et al., 2006); (Kreuz and Caucci, 2007); (Tsur et al., 2010); (Davidov et al., 2010) (or SVM-Perf as in the case of (Joshi et al., 2016)). (González-Ibánez et al., 2011) use SVM with SMO and logistic regression. Chi-squared test is used to identify discriminating features. (Reyes et al., 2012) use Naive Bayes and SVM. They also show Jaccard similarity between labels and the features. (Riloff et al., 2013) compare rule-based techniques with a SVM-based classifier. (Liebrecht et al., 2013) use balanced winnow algorithm in order to determine high-ranking features. (Reyes et al., 2012) use Naive Bayes and decision trees for multiple pairs of labels among irony, humor, politics and education. (Bamman and Smith, 2015) use binary logistic regression. (Wang et al., 2015) use SVM-HMM in order to incorporate sequence nature of output labels in a conversation. (Liu et al., 2014) compare several classification approaches including bagging, boosting, etc. and show results on five datasets. On the contrary, (Joshi et al., 2016) experimentally validate that for conversational data, sequence labeling algorithms perform better than classification algorithms. They use SVM-HMM and SEARN as the sequence labeling algorithms.

3.3 Features Used

(González-Ibánez et al., 2011) use sentiment lexicon-based features. In addition, pragmatic features like emoticons and user mentions are also used. According to this paper, Sarcasm transforms the polarity of an apparently positive or negative utterance into its opposite. They proposed a method for constructing a corpus of sarcastic Twitter messages in which determination of the sarcasm of each message has been made by its author. They use this reliable corpus to compare sarcastic utterances in Twitter to utterances that express positive or negative attitudes without sarcasm. They investigated the impact of lexical and pragmatic factors on machine learning effectiveness for identifying sarcastic utterances and then compared the performance of machine learning techniques and human judges on this task. Their paper also claimed that neither the human judges nor the machine learning techniques perform very well. The Lexical features that they had used are described as :

- **Unigrams**
- **Dictionary-based** : These features were derived from i) Pennebaker et al.s LIWC (2007) dictionary, which consists of a set of 64 word categories grouped into four general classes: Linguistic Processes (LP)(e.g., adverbs, pronouns), Psychological Processes (PP) (e.g., positive and negative emotions), Personal Concerns (PC) (e.g, work, achievement), and Spoken Categories (SC) (e.g., assent, nonfluencies).
- WordNet Affect (WNA) (Strapparava and Valitutti, 2004).
- list of interjections (e.g., ah, oh, yeah), and punctuations (e.g., !, ?).

The Pragmatic Features that they had used are described as

- Positive emoticons such as smileys.
- Negative emoticons such as frowning faces.
- ToUser, which marks if a tweets is a reply to another tweet (signaled by $< @user >$)

(Reyes et al., 2012) introduce features related to ambiguity, unexpectedness, emotional scenario, etc. Ambiguity features cover structural, morphosyntactic, semantic ambiguity, while unexpectedness features measure semantic relatedness. According to this paper, these features intended to symbolize low and high level properties of figurative language based on formal linguistic elements. According to their results, it is important to highlight that the set of features work together as a system; i.e. no single feature is distinctly humorous or ironic, but all of them together provide a useful linguistic inventory for detecting these types of figurative devices at textual level.

(Joshi et al., 2015a) use features corresponding to the linguistic theory of incongruity. There paper is described as :

3.3.1 Sarcasm Detection using context Incongruity

Sarcasm is defined as a cutting, often ironic remark intended to express contempt or ridicule. In linguistic theory, Context incongruity forms the basis of sarcasm. Contextual Incongruity is the disparity between what the speaker/text wants to convey and what he actually meant.

- For eg: Imagine a situation where:
- Jay has agreed to give John a ride to school.
- Jay is 1hr late to pick up john.
- John says: Wow! You are so punctual. (Sarcastic)

Thus, inter-sentential incongruity is also explored for sarcasm detection. On the basis of degree/time to detect the contextual incongruity there are two types of incongruity defined by (Joshi et al., 2015b):

- **Explicit Incongruity**: Easily observable case through the words of different polarities.
  - I absolutely like it when people backstab me.

  Here we have words of opposite polarity and sarcasm is explicit.
- **Implicit Incongruity**: Here, sarcasm is covertly expressed through phrases of implied sentiment.
  - I love this paper so much that I made a doggy bag out of it.

Different types of features that are used by Joshi et.al., (Joshi et al., 2015b) :-

- **Lexical features**: Unigram based features obtained via χ^2 test.
- **Pragmatics** : Emoticons, laughter expressions, punctuation marks & Capitals.
- **Explicit Incongruity features(Numeric feature)** used by them are as follows:-
  1. No. of times positive word is followed by negative word and vice versa.
  2. Largest positive negative subsequence.
  3. Number of positive and negative words.
  4. Lexical polarity - Highly positive on surface tends to be more sarcastic.
- **Implicit incongruity features(Boolean feature)** are as follows:-
  1. Extracted implied sentiment phrases.
     - Eg. I absolutely adore it when my bus is late.

(Rajadesingan et al., 2015) use extensions of words, number of flips, readability features in addition to others. Their paper aims to address the difficult task of sarcasm detection on Twitter by leveraging behavioral traits intrinsic to users expressing sarcasm. They identify such traits using the users past tweets. They also employed theories from behavioral and psychological studies to construct a behavioral modeling framework tuned for detecting sarcasm. They build a framework called **SCUBA framework** Sarcasm Classification Using a Behavioral modeling Approach. According to this paper, tweets are not always created in isolation. When posting sarcastic tweets, users make conscious efforts to express their thoughts through sarcasm. They may decide to use sarcasm as a behavioral response to a certain situation, observation, or emotion. These situations, observations, or emotions may be observed and analyzed on Twitter. They had observed that some individuals have more difficulty in creating or recognizing sarcasm than others due to cultural differences, language barriers. In contrast, some individuals have a higher propensity to use sarcasm than others. Their framewok system called SCUBA also considers users likelihood of being a sarcastic person. This can be achieved on Twitter by analyzing the users past tweets. Using their observations, sarcasm generation can be characterized as one (or a combination) of the following:

- **Sarcasm as a contrast of sentiments**: A popular perception of sarcasm among researchers is that sarcasm is a contrast of sentiments. A classical view of sarcasm, based on the traditional pragmatic model, argues that sarcastic utterances are first processed in the literal sense and if the literal sense is found incompatible with the present context, only then is the sentence processed in its opposite (ironic) form. This perceived contrast may be expressed with respect to mood, affect or sentiment.
- **Sarcasm as a complex form of expression**: (Rockwell, 2007) showed that there is a small but signifi- cant correlation between cognitive complexity and the ability to produce sarcasm. A high cognitive complexity involves

understanding and taking into account, multiple perspectives to make cogent decisions. Furthermore, expressing sarcasm requires determining if the environment is suitable for sarcasm, creating an appropriate sarcastic phrase and assessing if the receiver would be capable of recognizing sarcasm. Therefore, sarcasm is a complex form of expression needing more effort than usual from the user.

- **Sarcasm as a means of conveying emotion**: Sarcasm is primarily a form of conveying ones emo- tions. While sarcasm is sometime interpreted as aggressive humor(Basavanna, 2000) or verbal aggression(Toplak and Katz, 2000), it also functions as a tool for self expression. Past studies(Grice, 1978), recognize that sarcasm is usually expressed in situations with negative emotions and attitudes.

- **Sarcasm as a possible function of familiarity**: Friends and relatives are found to be better at recog- nizing sarcasm than strangers (**?**). Further, it has been demonstrated that the knowledge of language (Cheang and Pell, 2011) and culture(Rockwell and Theriot, 2001) also play an important role in the recognition and usage of sarcasm.

- **Sarcasm as a form of written expression**: Sarcasm in psychology has been studied primarily as a spoken form of expression. However, sarcasm is quite prevalent in written form as well, especially with the advent of online social networking sites. Through time, users have become more adept at conveying sarcasm in writing by including subtle markers that indicate to the unassuming reader, that the phrase might be sarcastic. For example, while youre so smart does not hint at sarcasm, “Woowwww you are SOOOO cool” elicits some doubts about the statements sincerity.

Abhijit Mishra and Bhattacharyya(Mishra et al., 2017) conduct additional experiments with human annotators where they record their eye movements. Based on these eye movements, they design a set of gaze based features such as average fixation duration, regression count, skip count, etc. In addition, they also use complex gaze-based features based on saliency graphs which connect words in a sentence with edges representing saccade between the words. According to this paper, Sarcasm can often be traced to incongruity that becomes apparent as the full sentence unfolds. This presence of incongruity- implicit or explicit- affects the way readers eyes move through the text. They observe the difference in the behaviour of the eye, while reading sarcastic and non sarcastic sentences. Motivated by this observation, they augment traditional linguistic and stylistic features for sarcasm detection with the cognitive features obtained from readers eye movement data. They performed statistical classification using the enhanced feature set that is obtained. The augmented cognitive features improve sarcasm detection by 3.7% (in terms of F-score), over the performance of the best reported system.

4 Deep Learning Approaches

As the deep learning based models gain popularity for NLP tasks and other classification tasks, few such different deep learning based approaches have been reported for automatic sarcasm detection as well.

Silvio Amir et.al. (Amir et al., 2016) presented a novel convolutional network-based that learns user embeddings in addition to utterance-based embeddings. They introduced a first of its kind deep neural network architecture for automated sarcasm detection from user embeddings. Recent work in sarcasm detection has emphasized the need for models to capitalize on contextual features, beyond lexical and syntactic cues present in utterances. For example, different speakers will tend to employ sarcasm regarding different subjects and, thus, sarcasm detection models ought to encode such speaker information. Available methods achieves this by way of laborious feature engineering. By contrast, they propose to automatically learn and then exploit user embeddings, to be used in concert with lexical signals to recognize sarcasm. Their approach does not require elaborate feature engineering (and concomitant data scraping); fitting user embeddings requires only the text from their previous posts.

They proposed a novel approach to sarcasm detection on social media that does not require extensive manual feature engineering. Instead, they develop a neural model that learns to represent and exploit embeddings of both content and context. For the former, they induce vector lexical representations via a convolutional layer; for the latter, their model learns user embeddings. Inference concerning whether an utterance (tweet) was intended ironically (or not) was then modeled as a joint function of lexical representations and corresponding author embeddings.

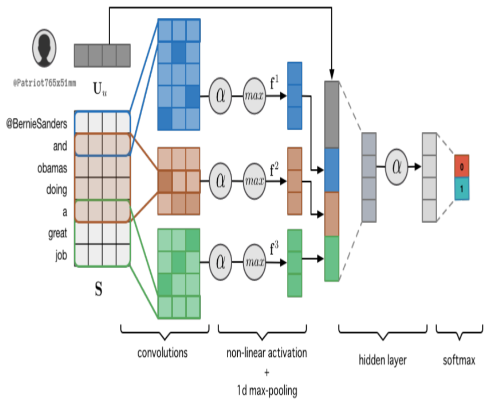

Figure 4: Illustration of the CNN model for sarcasm detection. The model learns to represent and exploit embeddings of both content and users in social media.(Amir et al., 2016).

- **Learning User Embedding:** Their goal is to learn representations (vectors) that encode latent aspects of users and capture homophily, by projecting similar users into nearby regions of the embedding space. They hypothesize that such representations will naturally capture some of the signals that have been described in the literature as important indicators of sarcasm, such as contrasts between what someone believes and what they have ostensibly expressed ((Campbell and Katz, 2012)) or (Kreuz and Caucci, 2007) principle of inferability, stating that sarcasm requires a common ground between parties to be understood.

  To induce the user embeddings, they adopt an approach similar to that described in the preliminary work of Li et.al. (Li et al., 2015). In particular, they capture relations between users and the content they produce by optimizing the conditional probability of texts, given their authors (or, more precisely, given the vector representations of their authors). This method is akin to Le and Mikolov (Mikolov et al., 2013) *Paragraph Vector model*, which jointly estimates embeddings for words and paragraphs by learning to predict the occurrence of a word w within a paragraph p conditioned on the (learned) representation for p.

  Given a sentence $S = w_1, ..., w_N$ where w_i denotes a word drawn from a vocabulary V, their aim to maximize the following probability:

  $$P(S|User_j) = \sum_{w_i \in S} \log P(w_i|u_j) + \sum_{w_i \in S} \sum_{w_i \in C(w_i)} \log P(w_i|e_k) \tag{1}$$

  Where, $C(w_i)$ denotes the set of words in a pre-specified window around word w_i, $e_k \in R^d$ and $u_j \in R^d$ denote the embeddings of word k and user j, respectively. This objective function encodes the notion that the occurrence of a word w, depends both on the author of S and its neighboring words.

  The conditional probabilities in Equation **??** can be estimated with log-linear models of the form:

  $$P(w_i|x) = \frac{exp(W_i.x + b_i)}{\sum_{k=1}^{Y} exp(W_k.x + b_i)} \tag{2}$$

  Where x denotes a feature vector, W_k and b_k are the weight vectors and bias for class k.

  To learn meaningful user embeddings, they seek representations that are predictive of individual word-usage patterns. In light of this motivation, they approximate $P(w_i|u_j)$ via the following hinge-loss objective which we aim to minimize:

  $$L(w_i, user_j) = \sum_{w_l \in V, w_l \notin S} max(0, 1 - e_i.u_j + e_l.u_j) \tag{3}$$

  where e_l is the embedding.

- **Proposed Model**: Given a message S authored by user u, they try to capture both the relevant aspects of the content and the relevant contextual information about the author. To represent the content, they use pre-trained word embeddings as the input to a convolutional layer that extracts high-level features. More formally, let $E \in R^d \times |V|$ be a pretrained word embedding matrix, where each column represents a word from the vocabulary V as a d-dimensional vector.

Further they run convolutional filters of different sizes on these sentence embedding matrix to learn different feature maps, henceforth after max-pooling the final feature vector is passed through a fully-connected layer.

The proposed model of (Amir et al., 2016) outperforms (by over 2in absolute accuracy) a recently proposed state-of-the-art model that exploits an extensive, hand-crafted set of features encoding user attributes and other contextual information. Unlike other approaches that explicitly exploit the structure of particular social media services, such as the forum where a message was posted or metadata about the users, learning user embeddings only requires their preceding messages. Yet, the obtained vectors are able to capture relevant user attributes and a soft notion of homophily, and hence is easier to deploy over different social media environments.

(Zhang et al., 2016) investigated the use of **neural network** for **Tweet Sarcasm Detection**, and compared the effects of the continuous automatic features with discrete manual features. In particular, they used a bi-directional gated recurrent neural network to capture syntactic and semantic information over tweets locally, and a pooling neural network to extract contextual features automatically from history tweets. Their results showed that neural features gave improved accuracies for sarcasm detection, with different error distributions when compared with discrete manual features.

They have shown results with the baseline discrete model and compared it with their proposed neural model. We elaborate their neural model below:

- **Proposed Neural Model:** In contrast to the discrete model, their neural model explores low-dimensional dense vectors as input.

- Figure 5(a) shows the discrete model. In particular, the local component (the left Figure 5(a)) is used to extract features f from the target tweet content, and the contextual component (the right Figure 5(a)) is used to extract contextual features $f^{'}$ from the history tweets of the author.

- Figure 5(b) shows the overall structure of their proposed neural model, which has two components, corresponding to the local and the contextual components of the discrete baseline model, respectively. The two components use neural network structures to extract dense real-valued features h and $h^{'}$ from the local and history tweets, respectively, and they add a non-linear hidden layer to combine the neural features from the two components for classification.

In conclusion, (Zhang et al., 2016) have showed that neural network model gave improved results over a state-of-the-art discrete model. In addition, they found out that under the neural setting, contextual tweet features are as effective for sarcasm detection as with discrete models.

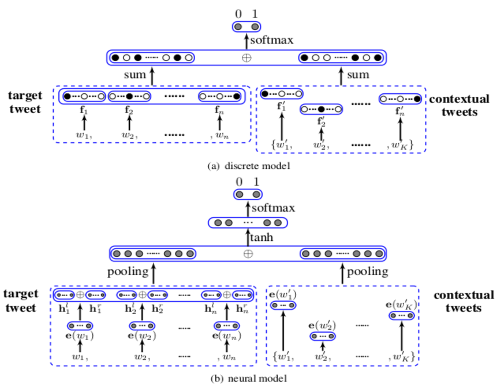

Figure 5: Discrete and neural models for tweet sarcasm detection (Zhang et al., 2016).

(Ghosh and Veale, 2016) use a combination of convolutional neural network, LSTM followed by a DNN. They compare their approach against recursive SVM, and show an improvement in case of deep learning architecture. Their architecture is shown in Figure 6

They have different layers in their architecture which are briefly described as :

- **Input Layer**: A tweet as input containing n words. The tweet is converted into a vector by replacing each word with its dictionary index $s \in \Re^{1 \times n}$. To resolve different lengths of input, the tweet vector is padded and the tweet is converted into matrix $s \in \Re^{1 \times l}$,

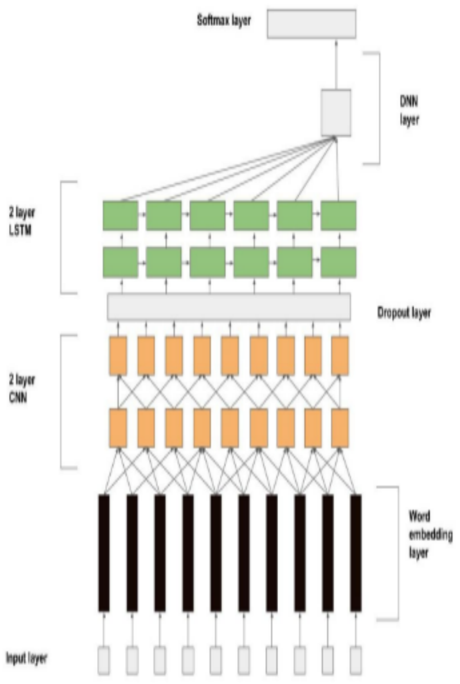

Figure 6: Architecture.

where l is the maximum length of tweets in the input corpus. The input vector is fed to the embedding layer which converts each word into a distributional vector of dimension D. Thus the input tweet matrix is converted to $s \in \Re^{l \times D}$.

- **Convolutional network Layer**: Convolution network is used to reduce frequency variation through convolutional filters and extracting discriminating word sequences as a composite feature map for the LSTM layer. The convolution operation maps the input matrix $s \in \Re^{l \times D}$ into $c \in \Re^{s+m-1}$ using a convolutional filter $k \in \Re^{D \times m}$. Initially, pass the output of the convolutional network through a pooling layer and max-pooling is used with size 2 and 3. Later, discard the max-pooling layer and fed the LSTM network. Each component is computed as follows:

  $$c_i = (s * k)_i \sum_{k,j} (S_{:,i-m+1:i} \otimes F)_{kj}$$

  Convolution filter, which has the same dimension D of the input matrix, which slides along the column dimension of the input matrix, performing an element wise product between a column slice s and a filter matrix k producing a vector component c_i and summed to create a feature map $c \in \Re^{1(|s|m+1)}$. f filters create a feature map $C \in \Re^{f(|s|m+1)}$. They have chose Sigmoid for non-linearity. Initially they passed the output of the convolutional network through a pooling layer and max-pooling is used with size 2 and 3. Later, they discarded the max-pooling layer and fed the LSTM network with all of the composite features to judge sarcasm, which improved the performance of their model.

- **LSTM Layer**: RNN has demonstrated the power of semantic modelling quite efficiently by incorporating feedback cycles in the network architecture. A variant of RNN, i.e., LSTM, which is able to plot long term dependencies by defining each memory cell with a set of gates $\Re^d$, where d is the memory dimension of hidden state of LSTM, and it does not suffer from vanishing or exploding gradient while performing back propagation through time. LSTM contains three gates, which are functions of x_t and h_{t1} : input gate i_t , forget gate f_t , and output gate o_t . The gates jointly decide on the memory update mechanism. Equation (3.2) and (3.1) denote the amount of information to be discarded or to be stored from and to store in memory. Equation (3.4) denotes the output of the cell c_t.

  $$i_t = \sigma(W_i[h_{t1}, x_t] + b_i) \tag{4}$$

  $$f_t = \sigma(W_f[h_{t1}, x_t] + b_f) \tag{5}$$

  $$q_t = \tanh(W_q[h_{t1}, x_t] + b_q) \tag{6}$$

  $$o_t = \sigma(W_o[h_{t1}, x_t] + b_o) \tag{7}$$

  $$c_t = f_t \odot c_{t1} + i_t \odot q_t \tag{8}$$

  $$ht = o_t \odot \tanh(c_t) \tag{9}$$

- **Deep Neural Network Layer**: The output of LSTM layer is passed to a fully connected DNN layer, which produces a higher order feature set based on the LSTM output, which is easily separable for the desired number of classes. Finally, a softmax layer is added on top of the DNN layer.

(Vosoughi et al., 2016) present Tweet2Vec, a novel method for generating general purpose vector representation of tweets. Their model learns tweet embeddings using character-level CNN-LSTM encoder-decoder. They trained their model on 3 million, randomly selected English-language tweets. The vector representations of tweets generated by their model are generic, and hence can be applied to a variety of tasks. Though the model presented in their paper is trained on English-language tweets, the method presented can be used to learn tweet embeddings for different languages. These learned tweet embeddings can be used for various classification task like sarcasm detection, sentiment detection etc. The architecture present in their paper is shown in Figure 7. Their CNN-

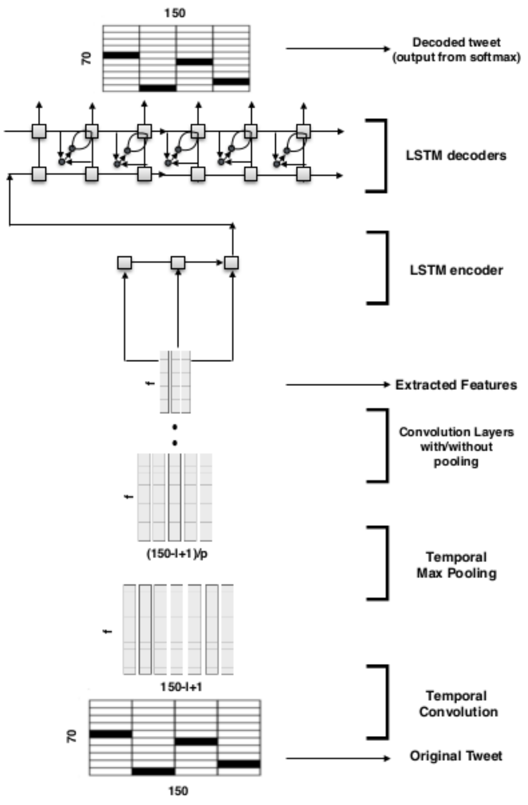

Figure 7: Illustration of the CNN-LSTM Encoder-Decoder Model(Vosoughi et al., 2016).

LSTM encoder-decoder model draws on the intuition that the sequence of features (e.g. character and word n-grams) extracted from CNN can be encoded into a vector representation using LSTM that can embed the meaning of the whole tweet. The input and output to the model are the tweet represented as a matrix where each row is the one-hot vector representation of the characters. Their encoder-decoder model is described in the below section :

- **Encoder**: Given a tweet in the matrix form T $(size\ :\ 150 \times 70)$, where 70 denotes the total possible characters that can come in the tweet and 150 denotes the maximum length of the tweet along with the padding, CNN extracts the features from the character representation. The one-dimensional convolution involves a filter vector sliding over a sequence and detecting features at different positions. The new successive higher-order window representations then are fed into LSTM. Since LSTM extracts representation from sequence input, they have not applied pooling after convolution at the higher layers of Character-level CNN model. Their encoding procedure can be summarized as:

$$H_{conv} = CharCNN(T) \tag{10}$$

$$h_t = LSTM(g_t, h_{t1}) \tag{11}$$

where $g = H^{conv}$ is an extracted feature matrix where each row can be considered as a time-step for the LSTM and h_t is the hidden representation at time-step t. LSTM operates on each row of the H_{conv} along with the hidden vectors from previous time-step to produce embedding for the subsequent time-steps. The vector output at the final time-step, enc_N , is used to represent the entire tweet. The size of the enc_N taken by them is 256.

- **Decoder**: The decoder operates on the encoded representation with two layers of LSTMs. In the initial time-step, the end-to-end output from the encoding procedure is used as the original input into first LSTM layer. The last LSTM decoder generates each character, C, sequentially and combines it with previously generated hidden vectors of size 128, h_{t1} , for the next time-step prediction. The prediction of character at each time step is given by:

$$P(C_t|.) = softmax(T_t, h_{t1}) \tag{12}$$

where C_t refers to the character at time-step t, T_t represents the one-hot vector of the character at time-step t. The result from the softmax is a decoded tweet matrix T^{dec}, which is eventually compared with the actual tweet or a synonym-replaced version (you can refer the paper for this(Vosoughi et al., 2016)) of the tweet for learning the parameters of the model.

5 Conclusion

In this paper, we have covered various literature that has been presented in the field of sarcasm detection of tweets and other textual data. We have described these past works on the basis of rule-based, statistical-based and deep-learning based approaches. In summary a lot of work has been done in the field of sarcasm detection, in particular past approaches have used different types of features to train their sarcasm classifier. Sarcasm occurs typically due to incongruity in text, but sometimes a system may need to go beyond the information in the text to detect sarcasm like author details and tweet history. Such cues have helped improve performance of sarcasm detection systems. Finally, we have also touched upon recent trends in sarcasm detection, where researcher have also tried with deep learning architectures and have shown improvement in performance as compared to statistical baseline approaches.

References

Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*.

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

M Basavanna. 2000. *Dictionary of psychology*. Allied Publishers.

Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. 2015. Parsing-based sarcasm sentiment recognition in twitter data. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 1373–1380. IEEE.

Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. 2014. An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49.

John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, 49(6):459–480.

Henry S Cheang and Marc D Pell. 2011. Recognizing sarcasm without language: A cross-linguistic study of english and cantonese. *Pragmatics & Cognition*, 19(2):203–223.

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 297–304, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of NAACL-HLT*, pages 161–169.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

H Paul Grice. 1978. Further notes on logic and conversation. *1978*, 1:13–128.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015a. Harnessing context incongruity for sarcasm detection. In *ACL (2)*, pages 757–762.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015b. Harnessing context incongruity for sarcasm detection. *Volume 2: Short Papers*, page 757.

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

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2015. Your sentiment precedes you: Using an authors historical tweets to predict sarcasm. In *6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis WASSA*, page 25.

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

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. *CoRR*, abs/1510.05198.

CC Liebrecht, FA Kunneman, and APJ van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not.

Peng Liu, Wei Chen, Gaoyan Ou, Tengjiao Wang, Dongqing Yang, and Kai Lei, 2014. *Sarcasm Detection in Social Media Based on Imbalanced Classification*, pages 459–471. Springer International Publishing, Cham.

Diana Maynard and Mark A. Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC*, pages 4238–4243. European Language Resources Association (ELRA).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Harnessing cognitive features for sarcasm detection. *arXiv preprint arXiv:1701.05574*.

Tomás Ptácek, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *COLING*, pages 213–223.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM.

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

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, volume 13, pages 704–714.

Patricia Rockwell and Evelyn M Theriot. 2001. Culture, gender, and gender mix in encoders of sarcasm: A self-assessment analysis. *Communication Research Reports*, 18(1):44–52.

Patricia Rockwell. 2007. The effects of cognitive complexity and communication apprehension on the expression and recognition of sarcasm. *Hauppauge, NY: Nova Science Publishers*.

Joseph Tepperman, David R Traum, and Shrikanth Narayanan. 2006. " yeah right": sarcasm recognition for spoken dialogue systems. In *INTERSPEECH*.

Maggie Toplak and Albert N Katz. 2000. On the uses of sarcastic irony. *Journal of pragmatics*, 32(10):1467–1488.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*.

Tony Veale and Yanfen Hao. 2010. Detecting ironic intent in creative comparisons. In *ECAI*.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1041–1044. ACM.

Zelin Wang, Zhijian Wu, Ruimin Wang, and Yafeng Ren. 2015. Twitter sarcasm detection exploiting a context-based model. In *Proceedings, Part I, of the 16th International Conference on Web Information Systems Engineering — WISE 2015 - Volume 9418*, pages 77–91, New York, NY, USA. Springer-Verlag New York, Inc.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network.