Unparalleled Sarcasm: A Framework of Parallel Deep LSTMs with Cross Activation Functions Towards Detection and Generation of Sarcastic Statements

Sourav Das (sourav.das.research@gmail.com)
Future Institute of Technology https://orcid.org/0000-0003-4864-1635
Soumitra Ghosh
Indian Institute of Technology Patna
Anup Kumar Kolya
RCC Institute of Information Technology
Asif Ekbal
Indian Institute of Technology Patna

Research Article

Keywords: Sarcasm Detection, Natural Language Generation, Sarcasm Generation, Feature Optimization, Long Short Term Memory Networks

Posted Date: November 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-593322/v1

*Un*paralleled Sarcasm: A Framework of Parallel Deep LSTMs with Cross Activation Functions Towards Detection and Generation of Sarcastic Statements

Sourav Das · Soumitra Ghosh · Anup Kumar Kolya · Asif Ekbal

Received: date / Accepted: date

Abstract Sarcasm is a modest kind of mockingly expressing one's own thoughts. With the advent of social networking communication, new routes of sociability have proliferated. It may also be stated that the four chariots of being socially hilarious nowadays are humour, irony, sarcasm, and wit. Sarcasm is a clever means of encapsulating any intrinsic truth, message, or even satire in a humorous way. In this paper, we manually extract the features of a benchmark pop culture sarcasm corpus encompassing sarcastic conversations and monologues in order to build padding sequences from the vector representations' matrices. We also suggest a hybrid of four Parallel Long Short Term Networks (pLSTM), each with its own activation classifier. Consecutively it achieves 98.31% accuracy among the test cases on open-source English literature. Our approach transcends several previous state-of-the-art works and results in sophisticated sarcastic statement generation. We also culture the probable prospects for producing even better refined automated sarcasm generation.

Keywords Sarcasm Detection · Natural Language Generation · Sarcasm Generation · Feature Optimization · Long Short Term Memory Networks

1 Introduction

Sarcasm in any language is strongly dependent on the context of the topic under discussion. That is why it may be delivered without revealing any evidence of emotion, such as a straight face, a smile, or even laughing. The delivery of a sarcastic statement also depends on the speaker's personality. When the sarcasm is a part of an ongoing conversation, the previous and following statements, questions, or topics are just as crucial as the sarcasm itself, as those contexts lay the groundwork for something funny to come up at that given point in time. It is more difficult to determine the underlying meaning of a sarcastic statement than it is to determine the straight targeted sentiment polarity of any speech. However, while dealing with an isolated, sarcastic text corpus, we may not have to deal with the full surrounds and context, but simply with the sarcastic messages. This can also occur in the situations like a television comedy, or web series is being telecasted. If we suddenly start watching it at any point, we might get to understand the next joke or sarcasm. This recognition of sarcasm is generated by judging the contextual aspects using multiple traits of human cognition. For instance, a sarcastic dialogue like "Are you sure a dummy like me can handle something as complicated as a blanket?" from the popular situational comedy show The Big Bang Theory[1] needs little to no prequel aspect to understand. It can

Sourav Das
Department of Computer Science and Engineering (With Specialization in AIML & IOT CS Including BCT), Future Institute of Technology, Kolkata 700154, India
E-mail: sourav.das.research@gmail.com

Soumitra Ghosh
Department of Computer Science and Engineering, Indian Institute of Technology Patna, India
E-mail: ghosh.soumitra2@gmail.com

Anup Kolya
Department of Computer Science and Engineering, RCC Institute of Information Technology, Kolkata 700015, India
E-mail: anup.kolya@gmail.com

Asif Ekbal
Department of Computer Science and Engineering, Indian Institute of Technology Patna, India
E-mail: asif@iitp.ac.in

[1] https://en.wikipedia.org/wiki/TheBigBangTheory

be inferred that the speaker is upset against another person for some reason, and he or she is expressing that topic of discontent in another conversation with the help of sarcasm.

The majority of research in this field is done with binary classification tasks and empirical information to determine whether a text or phrase is sarcastic or not [1]. Furthermore, the accuracy of neural networks used for sarcasm detection tasks is dependent on context, or sentence level attention encompassing the full sarcasm corpus [2], [3]. Because neural networks express computational combinations of weighted vectors via input neurons, training can enhance their capacity to reproduce the weightage combinations over time, improving the detection power of sarcasm but failing to accrue subjectivity. This leads to another modality of computational linguistics research; the grey area of sentiments with sarcasm. It is not always the linear sentiments instead a far more mixed plate of complex emotions. Some people tend to express anger, disgust, joy-like sentiments within a sarcastic opinion style. This sarcastic way of speaking can also be shifted for carrying positivity or negativity towards any brand, campaign, events, commercial products, etc. We show such a tweet in Figure 1.

Fig. 1: A sarcastic tweet on the negative popularity of Instagram.

It is determinable from the tweet that the tweeting person is exhibiting a negative view through a mockery of modern trends by taking pictures of any activities and uploading them to the social networks, here Instagram. Using similar scenarios to obtain the sarcasm within a sentence, the performance of the sarcastic sentiment analysis could be enhanced when the sarcasm can itself be identified within a sentence [4].

The main aim of this work is to accurately identify the complex sarcasm within spoken sentences, which can lead to the successful auto-generation of grammatically and semantically correct sarcastic dialogues. First, the model takes up a sarcasm text corpus for reading and tokenizes the file for lexical occurrence indexing distributed within the file. Second, the padded distribution sequence for such tokens is determined and provided with categorical labels. We also print a set of such words from the corpus to check the validity of this step. Next, we print the vector-matrix gained from the file and combine our parallel LSTM networks by feeding the input with the distribution weighted vectors from the token indices. We deploy a robust and straightforward set of four parallel extended short-term networks (pLSTM), each with a different activation function. Thus, a cross-activation function incorporated the network for achieving a better validation accuracy of 98.31% compared with several other diverse baseline models. Finally, we initiate a text seeding with a provided range for generating a sarcastic dialogue, which takes the token index sequence and occurrence weightage into consideration and predicts the following word in the sentence with the best suitable fit.

The novel contributions of this paper are as follows:

- We present a detailed and manual feature study of already established sarcasm data to determine how conversational sarcasm occurs in sequence and how we can further analyze it.
- We detect the sarcasm successfully from conversational dialogue-based texts.
- We exploit a heterogeneous activation function setup with four parallel deep LSTM models (pLSTM) to depict the pattern of which activation function can be best suited for weighted sum combination and bias summarization from the sophisticated text corpus features.
- Finally, the main contribution of this work is to generate meaningful and natural sarcastic dialogues as text, just like a statement delivery from any human.

The rest of this paper is organized as: we discuss some foundation works, recent studies, and popular public datasets on sarcasm from social networks or English texts in Section 2. Section 3 deals with our choice of sarcasm corpus and the detailed manual feature extractions distinguished in subsections accordingly to the same. Section 4 presents the elaborated details of our proposed method on the pLSTM model architecture and operational process. Section 5 highlights the experimental setup and selected model parameters. Section 6 categorizes the obtained results by incorporating subsections to describe detailed result-oriented analysis, automated sarcasm generation, and benchmark analysis with similar work on open-source data. We discuss the performance strengths and shortcomings of our approach in Section 6.4 and finally conclude the article and outline future extensions of the work in Section 7.

2 Sarcasm Analysis in NLP

Initially, sarcasm detection started as a binary classification problem. The objective was simple; whether a statement is sarcastic or not. Complex parameters like sentence context, surroundings, human traits for speech expression, and voice tonality gradually evolved to the new scope. Discussing some of the significant and recent contributions in the domain so far, we categorize the previous works in the following three categories.

2.1 Background works

One good source of natural sarcasm is product reviews directly given by purchasers on e-commerce websites. These review comments can either be funny or not, but they represent examples of natural human modeling of sarcasm and a sense of humor. These comments often define a positive or negative user experience towards any product associated with the reviewer's perspective. In one of the earlier works, [5] represented a semi-supervised algorithm for identifying sarcasm from online product reviews. They took a paired approach to identify the sarcastic patterns generated from highly frequent words within a sentence. In addition, they demonstrated a classification algorithm with five-fold cross-validation of their product review corpus. Native English speakers manually annotated the corpus. They achieved a summarised accuracy of 94% in detecting sarcastic reviews. In a similarly followed work, [6] extended the work for streaming sarcastic tweets from Twitter as well as reviews from Amazon. They classified the sarcastic and non-sarcastic texts based on a KNN classifier, which labeled the texts. They manually deployed both the positive and negative texts for seeding based on the annotations. It was followed by cross-validation for classifying new such tweets or reviews into the same category. Omitting the stopword punctuation, they achieved an accuracy of 90.6%. **The authors in [7] developed various machine learning-based classifiers to detect sarcasm from tweets exploiting lexical, pragmatic, dictionary-based, and part-of-speech information alongside term-frequency, term-presence, and TF-IDF-based representations.**

Since Twitter became one of the most popular microblogging sites, it opened multiple new avenues of natural linguistics-based research. In work from 2011, researchers took an approach to fetch the lexical and pragmatic features of tweet words from standard word repositories [8]. It helped them narrow down even the non-sarcastic tweets to their sentiment polarity. They classified the polarity feature ranking and used the baseline regression algorithm to measure the ranking performance of unigrams, dictionary, and frequency-based lexical and pragmatic features. Finally, they compared their performance with manually human-annotated sarcastic tweets or regular polar tweets. [9] has discussed the overall coverage of corpus generation from Amazon, rather than only sarcastic reviews. It was collected in both documents as well as word level. The author analyzed that regular reviews mostly carry higher ratings than sarcastic reviews. Hence, sarcastic reviews are pretty targeted towards negative sentiment. However, people can judge by reading a review whether it is negatively impacted and how many ratings the reviewer may have given to the product. Also, unlike sarcastic statements, general reviews do not contain any contextual word utterance, which impacts the sentence in a comical form.

Rakov and Rosenberg proposed automatic sarcasm detection from a speech corpus [10]. They develop a corpus based on the voice pitch, mean, and deviation of generally sarcastic categorized voices. In such scenarios, the amplitude of voice reach and the octave change of tone also appear. Considering these features, the researchers proposed a K-means clustering algorithm to classify the sarcastic patterns from the corpus. Here, the change of word choice also plays a vital role because we do not choose the exact words for non-sarcastic and sarcastic statements. By following the proposed approach, they observed that the pitch and intensity of sarcastic tonalities from their respective corpus are detected by their system successfully. [11] showcased that sarcastic tweets are more aggressive than ironic ones. Ironical statements carry idioms wrapped within, whereas sarcastic statements can even be targeted for bullying and insult. Nevertheless, the paper findings, none of the sarcastic or ironic statements or tweets affect the polarity score of the respective tweet. Since we have already discussed how sarcasm and irony are categorically different, it is also demonstrated that both do not share any symmetrical distribution. With that, the person's intention to tweet sarcasm leads to an opposite meaning.

Deep learning-based methods have also found applications in detecting sarcasm from user-generated social media content. In this context, topic-enriched word embedding methods have been compared to traditional word-embedding techniques for their predictive ability (such as word2vec, fastText, and GloVe) [12]. In addition to feature sets based on word embedding, traditional lexical, pragmatic, implicit incongruity, and explicit incongruity are considered. ToçoğluAytuğ and Onan [7] proposed a machine learning-based technique to satire identification in Turkish news items, where they considered unigrams, bigrams, and tri-grams to model lexical information along with term-frequency, term-presence, and TF-IDF-based methods. In a following work by the authors [13], an ensemble approach for identifying satirical news in Turkish news items was introduced. To extract the feature sets, the approach provided here used linguistic and psychological feature sets (i.e., linguistic, psychological,

personal, spoken categories, and punctuation). Among the various machine learning and deep learning schemes employed in this work, the recurrent neural-based model outperformed the machine learning approaches.

2.2 Recent Works

2.2.1 ***Sarcasm Detection***

Recent trends in sarcasm detection open up to multiple diversified parameters besides the text-only approach. Mediums like personality, audio, video, vision and even dialogue delivery are considered for a better overall social humor analysis. These combined approaches for research are mentioned as a "multimodal" approach. **The role of linguistic and pragmatic features in automatic identification of sarcastic tweets from both negative and positive tweets was studied in [8]. Low accuracy scores from the classification performances by the human annotators and machine learning methods demonstrated the challenging nature of the sarcasm classification task. Riloff et al. [14] developed a sarcasm recognizer to detect different types of sarcasm in tweets. The authors described a unique bootstrapping method that automatically learns separate lists of positive and negative situation phrases from sarcastic tweets. It is observed that associating contrasting contexts with the words learned through bootstrapping improves memory for sarcasm recognition. Previous works have also studied irony identification, which often goes hand in hand with sarcasm. One such study [15] focused on irony from the perspective of a multidimensional model of literary components. An automated model was proposed to distinguish between ironic and non-ironic tweets, and the model was evaluated on four publicly available Twitter datasets. Results provide light on the metaphorical difficulties that arise in tasks including sentiment analysis, reputation evaluation, and decision-making.**

A work from 2018 [16] proposes a supervised learning approach to detect sarcasm from Facebook posts and images. Also, they cultured the comments and reply-based threads among the users for a better understanding of it. For the textual polarity, they considered the subjectivity and polarity of the posts and comments. For the image semantics and sarcasm detection, they used a recognition model for image captions to filter out the information. Simultaneously, the authors engaged a convolutional network to detect sarcasm from non-captioned images. [17] used a bidirectional recurrent network with gated units to form a Bayesian model capable of detecting the tendency of a conversation starter person (or the author as they state). For an empirical approach, they better results than prior works for their full balanced and unbalanced political and Reddit datasets. Their sarcastic detection probability scored as high as 84% accurate in an open comparative analysis with similar manually predicted or statistically predicted experiments. Using the benchmark dataset [18], a group of researchers investigated attention-based bidirectional and convolutional networks for sarcasm recognition. They used an existing developed sarcastic tweet collection and randomly streamed tweets on sarcasm. The attention layer is implemented within an LSTM network, incorporating the softmax activation function into the backpropagation characteristic. Backpropagation-based feedback aids in the identification and differentiation of tweets. Researchers implemented the attention layer for lexical level impact on tweets. With that, they considered the additional features such as punctuations in a sarcastic tweet the impact of uppercase or lowercase letters within tweets in conveying either something funny or ironical meaning.

[19] introduced a data preprocessing setup for efficient data representation formats in order to improve the corresponding inputs to various deep learning models. Several features were extracted to capture components of the social network user's writing style. Based on the integration of multiple deep learning approaches, a robust Deep Ensemble Soft Classifier (DESC) was constructed and trained on the retrieved features for figurative language identification. The DESC approach demonstrated strong performances against state-of-the-art methods when evaluated on three benchmark datasets. [20] presented sentiment categorization and sarcasm detection as interrelated aspects of a shared language issue in recent work. They devised numerous task-based learning algorithms for their proposed technique, as well as sentiment and sarcasm categorization tags. For their experiment, they paired the gated network with a fully linked layer and softmax activation. Their sarcastic sentiment outperformed earlier state-of-the-art efforts. Furthermore, they discovered that their suggested classifier makes better use of sentiment shift to identify sarcasm. They further tested the same with an attention-based network specifically for sarcasm detection, which performed better than their previously implemented network. [21] classified sarcasm detection into genres, naming polite, rude, and raging sarcasm. Instead of identifying the linear polarity of a sarcastic phrase, they retrieved the conflicting emotions involved with the sarcasm itself. They fragmented and ensembled many feature factors from the input tweets to improve semantic comprehension. Also, they analyzed the relationship of the person commenting sarcasm with their present state of emotion. Finally, to remove the fuzziness of the inputs, they used a rule-based classifier.

[22] proposed an end-to-end technique that requires no feature engineering or lexical dictionaries and a preprocessing phase that only comprises de-capitalization. The main contribution of this work was an unsupervised pre-trained transformer approach to capture metaphorical language in many of its forms. The proposed approach was observed to achieve commendable performances of comprehensive evaluation against state of the

art methods on multiple benchmark datasets. The authors in [3] experimented with several deep learning-based methods to address the task of detection of sarcasm in Twitter. The best results were attained from an ensemble of various transformer-based pre-trained language models. One of the common ways of starting and continuing the conversation flow in social networking platforms is communicating with each other. **The authors in [23] investigated the connections and similarities across a wide range of tasks, such as humour identification, sarcasm detection, offensive content detection, motivating content detection, and sentiment analysis on memes. They proposed a multi-task, multi-modal deep learning framework for the simultaneous solution of several problems. Sarcasm, sentiment, and emotion are all intertwined in a multi-modal conversational situation thus the work in [24] developed a multi-task deep learning framework to address all three issues jointly.** In relevance, the nature of the communicating person can also be revealed. **In another study on sarcasm identification, the authors in [25] used trigrams to represent text documents using an inverse gravity moment-based term weighted word embedding model. Critical terms are given a boost in value due to preserving the word order. The three-layer stacked bidirectional long short-term memory architecture proposed in this study is used to recognize sarcastic text documents in real-time.**

2.2.2 ***Sarcasm Generation***

The authors in [26] demonstrated a sarcasm-generation module for chatbots. *SarcasmBot* creates a sarcastic answer in response to user input. *SarcasmBot* is a sarcasm generator module with eight rule-based sarcasm generators, each of which outputs a different form of sarcastic remark. At run-time, one of these sarcastic generators is chosen based on user input parameters such as question type, number of entities, and so on. Two manual experiments are used to test the sarcasm-generation module. For the three metrics of coherence, grammatical accuracy, and sarcastic nature, human evaluators awarded average scores of more than 0.69 (out of 1). In addition, a comparison assessment was performed in which the majority of our evaluators were able to properly identify the output of *SarcasmBot* from among two possible outputs in 70.97% of test instances. In the following work, [27] by the authors, a web-based interface, *Sarcasm Suite*, was introduced that integrated and presented four sarcasm detection modules and one sarcasm generation module. The four categories of incongruity employed by sarcasm detection modules are sentiment incongruity, semantic incongruity, historical context incongruity, and conversational context incongruity. The sarcasm generating module in *Sarcasm Suite* is a chatbot that responds to user input sarcastically. This is the first study believed to involve computational sarcasm.

The authors in [28] presented a unique framework for generating sarcasm in this research. The system takes a negative opinion as input and converts it to a sarcastic version. For training, the methodology does not require any paired data. Only unlabeled non-sarcastic and sarcastic opinions were used to train the framework, which used reinforced neural sequences to learn and retrieve sequences. Due to the lack of a suitable dataset for the task, the authors created a benchmark dataset that included opinions and sarcastic paraphrases to evaluate the proposed approach. Comprehensive evaluation against popular unsupervised statistical and neural machine translation and style transfer techniques suggested that the developed system significantly outperformed the considered baselines. [29] proposed an unsupervised approach if sarcasm generation from a non-sarcastic textual input. Their solution models commonsense knowledge in a retrieve-and-edit framework to simulate two fundamental features of sarcasm: valence reversal and semantic incongruity with the context. While previous research on sarcasm creation focused on context incongruity, this work showed that combining valence reversal with semantic incongruity based on commonsense knowledge produces higher-quality sarcastic messages across various parameters. The developed system-generated sarcasm better than human evaluators 34% of the time, and better than a reinforced hybrid baseline 90% of the time, according to human review.

2.3 Corpora Development

Open-source, public corpora have unveiled new possibilities for relatable platforms in benchmark analysis among works led by the same data. Many standard corpora are rich with indexed balanced and unbalanced data or a mixture of standard statements with sarcastic ones. Different experimentation frameworks have been conducted on such data, providing scalability to the works. [30] contributed a novel sarcasm dataset from their work in 2016. They represented a Twitter-based sarcasm corpus, mainly containing #sarcasm hashtags. However, since the ”sarcastic” tweets in Twitter cannot generate any expressional behavior, they approach a latent-based semantic analysis first. Their test set of 39 thousand tweets consists of 46.15% sarcastic tweets and 53.85% generic tweets with a positive label. The split words, general and offensive words from the Tweets, are also included within the corpus.

Print media are well known for their sarcastic and satirical articles, criticism writings, and caricatures. Focusing on that aspect, [31] collected sarcastic articles on the latest events from The Onion[2] and non-sarcastic news reporting from the HuffPost[3]. Unlike Twitter, their primary objective for constructing the corpus was to collect preprocessed and clean English texts from the source itself. They extracted 28 thousand texts based on the filtration of sarcastic and general headlines. It was then categorized as an extraction label and further semantically processed to be utilized. In 2018, a group of researchers proposed an ironical tweet dataset for English tweets [32]. While we need a sentence construction to build up the impactful meaning, the information often tends to be at the later end. For this reason, they fragmented the tweets into informative and non-informative parts and made a cluster. When merged arbitrarily later, the informative parts fitted in with the random general remaining text of the tweets. Also, they captured the polarity shift of tweets carrying the sarcasm. They distributed the discussed observations in two subtasks while converging to the results for non-ironic tweet coverage up to 50.15% of their developed corpus.

3 Dataset

To train our model, we use the text corpus [33] from MUStARD, or Multimodal Sarcasm Detection Dataset. The dataset, as the name implies, contains multimodal components of sarcastic expressions, mostly video and audio. The data includes almost 6,000 videos from a variety of prominent TV comedy shows. The researchers later annotated these videos for their sarcastic authenticity and audio video alignment. The manual annotation process progressed by the mutual decision-making of the research team whether to label a video as sarcastic or not. The authors then annotated the video's utterances, identifying the context in which the characters delivered caustic lines. Furthermore, they picked over 600 films from their complete video library as a combination of balanced sarcastic and non-sarcastic labelling. We selected only the transcription or textual modality from the multimodality parameters of dialogues' utterances. Their overall character-label ratio and distribution assist see which characters have the greatest utterance of words throughout the conversations gathered and comprehend what proportion of dialogue contributions the characters have in developing the corpus. The final suggested textual corpus consists of 690 lines of unique line-indexed dialogues. The largest conversation per character cumulation reaches a maximum of up to unbroken 65 words, and the shortest dialogue varies from 7 words. It represents the fact that the corpus contains both monologues and dialogues. The researchers used sentence-level utterance characteristics from BERT [34] to represent the contextual text utterance. Bert-input refers to the full text data set. Finally, they show the number of occurrences of sarcastic word utterances in the respective sentences, as well as the first token average.

3.1 Sarcastic Words Distribution

We do not divide balanced or unbalanced data allocations any further because the whole corpus is made up of sarcastic duplex or simplex remarks. We basically attempt to extract the frequency of the hundred most repeating terms in dialogue in our fair share of study. Because there is no sequentially documented material from the literature instead of sarcastic exchanges, it may be assumed that each time these words appear in a dialogue or monologue, they are meant to be sarcastic. These words carry an emphasis on being funny. Hence, it subjectifies the context of person-related sarcasm within a sentence. Furthermore, we build sarcastic sentences with these words in our manual annotation process. The concept behind it is maintaining a concise gold set of sarcastic comments generated from human cognition. We keep the file to compare whether these words occur in our automated dialogue generation or not, defining our proposed framework's sophisticated word choosing capability. It helps to compare the automated sarcastic dialogues' colloquialism in the later part of the work with the naturally funny comments. For example, consider the following naturally sarcastic sentences below with the seed words *'oh'* and *'know'*.

"oh, so you're a scientist now?"
"you know you could really get help from a dictionary."

Each sentence carries semantic value for adequately conveying a message. We further fragment the employability of such words in sarcastic comments. For instance, we labeled the sarcastic applicability of the word 'well' as 'maybe' since it can be in both a severe sentence and in a funny one, too.

Like in a serious sentence:

"well, we still have an alternative option."

However, in a sarcastic sentence:

[2] https://www.theonion.com/

[3] https://www.huffingtonpost.in/

"well, there goes our chance of winning the match."

Table 1 shows the 10 most frequently occurring terms from the whole corpus.

Table 1: A list of the most often used terms and their frequency in the corpus.

Words	Frequency	Distribution (%)	Sarcastic Applications	No. of Comments	
				Sarcastic	Non-Sarcastic
'oh'	74	8.21	yes	62	121
'know'	49	7.90	yes	42	7
'like'	46	7.78	yes	16	30
'yeah'	43	7.66	yes	25	18
'well'	39	7.52	maybe	36	3
'go'	37	7.50	maybe	5	32
'right'	34	7.47	yes	19	15
'think'	32	7.46	maybe	23	9
'really'	30	7.44	yes	26	4
'see'	27	7.41	maybe	3	24

3.2 Sentiment Features Extraction

Now that we have the sarcastic words within the corpus, we further extract the parts of speech forms that lead to sentiment impact for these words. Every sarcastic statement contains impactful words that pack a punch for being humorous. Now, if we extract their respective part of speech forms, it can be conveyed that some forms carry more sentiment impact than others. For instance, it has already been shown that verbs and adverbs have higher emotional impacts than nouns [35]. Since the sarcastic texts contain dialogues and monologues, the appearance of a similar form of words can impact differently in distinct places. An example would be the following statements:

"I like their rejection.", and,
"I like their rejection as it will provide me the opportunity to prepare better."

Here, 'like' is the verb for both statements. Although in two contexts, the verb here impacts two completely different expressions of sentiment. Whereas the first statement generates a sarcastic tone with a negative definition, the second statement generates a positive explanation expanding the same queue. Henceforth, we consider the inconsistency between sentiments of the same PoS tags appearing in different contexts of the corpus. Therefore, to identify these inconsistencies, we extract the sentiment features of such PoS tagged words for both the occurrence in dialogue and monologues. Here, we maintain a manual parameter to distinguish between both types of texts with different word ranges. If a statement contains twenty or a smaller number of words, it is a dialogue. In more words than that, it qualifies as a monologue. Following that, we formulate an impact scaling factor of such words appearing in different places, as mentioned earlier.

For the contrast of words positioned for positive or negative meanings within the text, we calculate the impact scale of PoS tagged words in eq. 1 as:

$$I_{\text{PoS}} = (n + p)\frac{a(t_d + t_m)}{f_r} \tag{1}$$

Where n and p are the cumulations of negative and positive statements, while a is the number of appearances for such words in dialogues and monologues, as t_d and t_m from the corpus, finally, f_r is the frequency of the words. From a list of a hundred words extracted from the corpus, here we denote only the top ten words in Table 2 with their respective parts of speech tags, alongside their impact scale when used in sarcastic and non-sarcastic monologues versus dialogues.

The table shows that most identical words throughout the texts with their separate PoS tags are used both in sarcastic and non-sarcastic contexts. That justifies our assumption of the inconsistencies of these words, dependent on the context they are being used. Also, such words are statement deliveries from their on-screen characters from particular comedy shows. Therefore these words co-exist within the corpus, being situationally deployed in both sarcastic and non-sarcastic themes of the discussions.

Table 2: PoS tagging of the frequent words and determining their ratings on sarcastic and non-sarcastic scales in the corpus.

Words	Part of Speech	Part of Speech Tags	Sarcastic Impact Scale		Non-sarcastic Impact Scale	
			Monologues	Dialogues	Monologues	Dialogues
'oh'	'Noun'	'NN'	0.19	0.81	0.02	0.98
'know'	'Verb'	'VB'	0.23	0.77	0.14	0.86
'like'	'Preposition'	'IN'	0.12	0.88	0.37	0.63
'yeah'	'Noun'	'NN'	0.06	0.94	0.03	0.97
'well'	'Adverb'	'RB'	0.25	0.75	0.06	0.94
'go'	'Verb'	'VB'	0.05	0.95	0.09	0.91
'right'	'Adverb'	'RB'	0.10	0.90	0.15	0.85
'think'	'Verb'	'VB'	0.33	0.67	0.03	0.97
'really'	'Adverb'	'RB'	0.21	0.79	0.06	0.94
'see'	'Verb'	'VB'	0.17	0.73	0.17	0.83

3.3 Punctuation Features

While the word impact features can undoubtedly provide insight into the most impactful words from the corpus, their significance escalates when coupled with certain punctuation marks. When human sarcasm is directly converted into textual statements, for most of the time, sarcastic statements conclude with an exclamation (!) mark or question mark (?) or both following consecutively (?!). However, we have already discussed earlier that sarcastic dialogues can be delivered without showing any facial expressions whatsoever, i.e., keeping a straight face. These dialogues or statements are conveyed seriously, implying the full stop (.) used at the end of the statement when converted to text. For every application of a punctuation mark embodied with a sentence, we focus on manually extracting the number of such marks. Additionally, we intend to match these marks with the appearance of PoS tagged impactful words from our previous findings. This approach thoroughly helps identify the impact of words containing sentences coupled with their respective punctuation marks for a further detailed understanding of the statements they used (profound versus funny). For reference, from our extracted list of PoS tagged words, 'yeah' has been deployed in several distinct types of dialogue sentences within the corpus. Following are three examples of such sentences in which the use of the same exclamation mark has three different use cases for the exact words:

*"Oh, **yeah**, that can only be good for you."*
*"**Yeah**, I was the only person who could cheer her up."*
*"**Yeah**, I don't think so Joe."*

From the above set of statements, it is evident that each carries three different sentiments. While the first one is sarcasm, the second is a direct positive statement, finally, a direct negative statement. Nevertheless, each stamen is punctuated by a full stop. Hence, it is evident that statements with different expressions can conclude with the same punctuation mark. In a similar example, we take up another word 'really' to show three different exclamation marks, concluding three sentences, defining the contrasting usage of these words:

*"I'm **really** looking forward to it."*
*"You don't **really** believe in that superstition, do you?"*
*"Oh my God I would **really** love that!"*

In this set of examples, the appropriateness of the punctuation marks is synchronized with the expression of the statements. The first statement reflects a positive sentiment, hence concluding with a simple full-stop. The second one is a question, carrying a significant doubt, succeeded by a question mark. Finally, the third statement reflects overwhelm, a joy. For which an exclamation mark has followed it. It can be portrayed in the vivid contexts of statement constructions, which let us use identical punctuations for distinct statements or exploit several punctuations according to their proper use case scenarios. We manually annotate such features for each statement, consisting of the words discussed before. Furthermore, we cultivate the analyzed interpretation of the proposed initially corpus for subsequent phases of experimentation.

For the elaboration about how to select and tune heterogeneous activation functions for future corpora, we follow the lexical analysis steps done in the current work and feature selection from the data as per the dataset characteristics before deploying the proposed model.

4 Proposed Method

While detecting sarcasm is the foundation of our work, it is certainly not the least. Our prioritized objective is to generate automated sarcastic dialogue after successfully detecting the same. Hence, it can be depicted that we utilize a multitask objective to achieve both purposes. A similar type of approach has earlier been mentioned in [14]. One immediate enhancement through our proposed parallel neural network (here pLSTM) is that the successful identification of sequencing and semantics will carry to the dialogue generation, making it a similarly natural expressive sentence. We represent our proposed method for approach in Figure 2. For phase-by-phase elaboration, we further break down the critical standalone modules of our proposed approach.

4.1 Tokenization and Sequencing

It can be deducted from a previous work that, for unsupervised classification algorithms, the linear discourse task of segmentation is closely dependent on the cohesion of words [36]. It can be achieved by lexical tokenizing the sentences from a corpus. The lexical cohesion efficiency is responsible for attaching and binding the textual units together to form sentences. The cohesion efficiency also determines the usage of an identical set of words, synonyms, antonyms, and hypernyms among the words of a sentence. We have already achieved a similar positional parametric scale level from the previous section's analysis.

For the tokenization task, we consider a vector space in eq. 2 containing dialogue sentences V_s and a constrained number of parameters s_1, ... ,$s_k \, \epsilon \, V_s$. The number of parameters represents features like word frequency, distribution, PoS tags, and sarcastic impact. Hence, for every collection of sentence feature for V of the form:

$$\begin{aligned} v &= \omega_1 s_1 + \omega_2 s_2 + ... + \omega_k s_k \\ v &= \sum_{i=1}^{k} \omega_i s_i \epsilon V_s \end{aligned} \tag{2}$$

Here, ω_1, ... , ω_k in a uni-dimensional space is a linear combination of vectors s_1, ... ,s_k. We are considering the non-transitive features, i.e., the parameters which are impactful on their own. Once the linear parameters are cumulative and combined within the common space, the remaining task stands to arrange the ascending order of sequencing for these words, as they appear within the entire corpus range. This step combines two subphases. In the first sub-phase, the sequences are generated, while in the second sub-phase, the sequences are made padded in the ascending order of frequency. The logic behind producing padded sequences is to maintain a uniform input shape for the latter phases in the model. The padded matrix contains additional 0s for the short tokens, but a truncation for the tokens exceeds the maximum length. For the sequence segmentation S of sentence parameters ranging within the whole set of V_s, we hypothesize a length of k more maximum individual token sequence length for a random sentence N. For the occurrence of each token, we compare the boundary reference with that of k to check that the token limit is restricted to exact length. At the same time, k-1 indicates the shorter length, and k+1 determines the surpassing range. The k-1 set of tokens will be padded, and the k+1 set of tokens will be truncated to the margin of k itself. More formally, if k_{ij} defines the boundary length of a token, where i is the lower limit and j is the upper limit, and V_s contains the dialogue texts in a space, then equation 3 could be represented as:

$$V_s : S(k_{ij}) = \frac{1}{N-k} \sum_{i=0, j=n-1}^{N-k} (range(k) - (start_i + end_j)) \neq 0 \tag{3}$$

Following the sequence, the padded matrices are generated using correlational cells filled with 0s. Here we aim to maintain an exact matrix size for each token sequence to sustain the uniformity among the lexical analysis for vectorization. Hence, we add a total of r rows of padding for the row-wise filling up of empty cells and c columns of padding for the row-wise filling up of the same. The output shape will be generated as per Equation 4 and Equation 5 for rows and columns.

$$range(kij) = (end_i - start_i + r - 1) \tag{4}$$

$$range(k_{i_j}) = (end_i - start_i + c - 1) \tag{5}$$

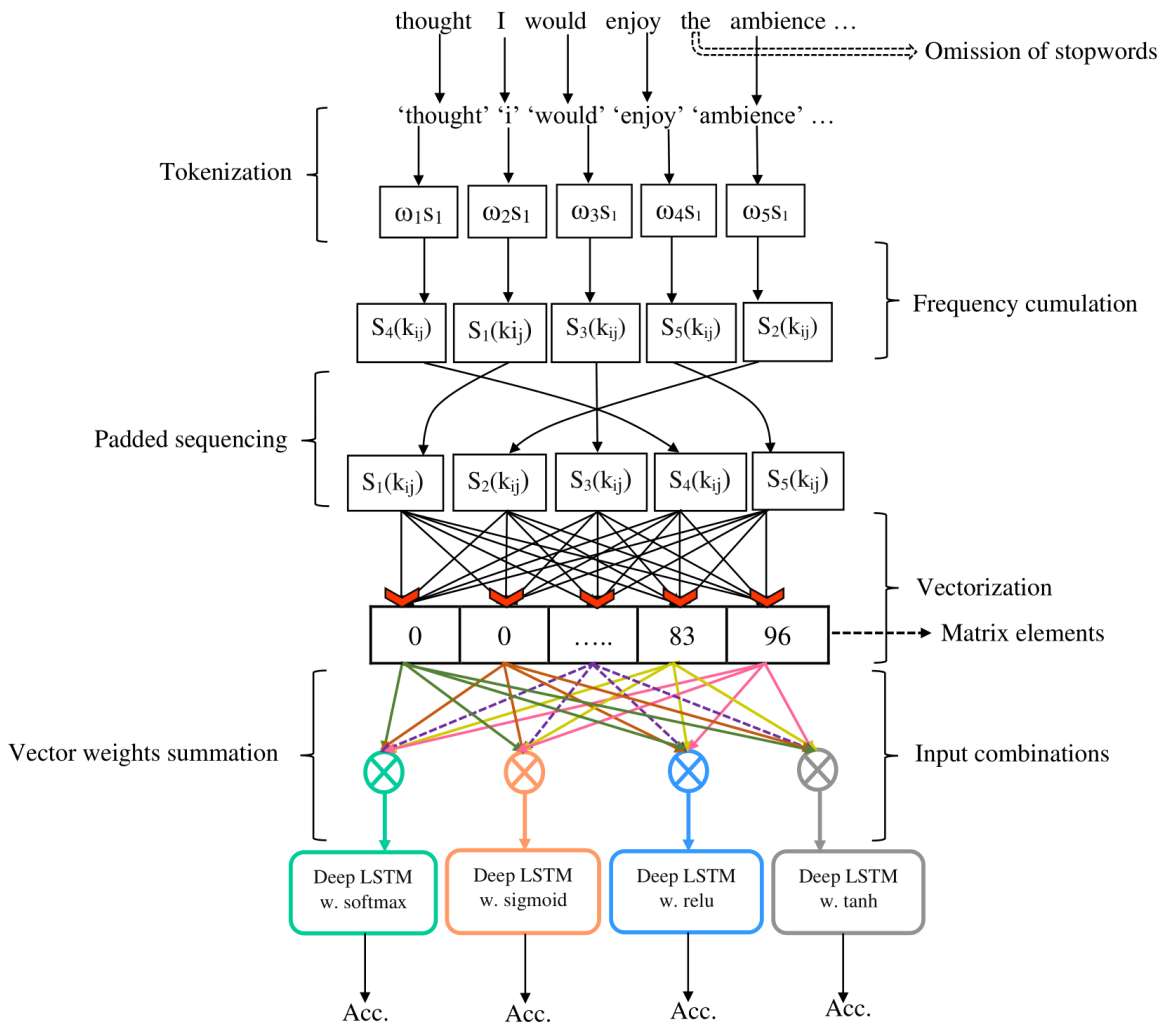

Fig. 2: Proposed approach for feature extraction and input production for pLSTM modules.

Where the optimal rows and columns padding matrices are of 1 capacity. We deduct the starting position of tokens $start_i$ from their respective ending positions end_i to evaluate the necessity of padding generation. The starting position also denotes the minimum length, whereas the ending indicates the maximum length. Followingly, these padding sequences are aligned to be vectorized.

4.2 Vectorization

At this point, it can be stated that we have successfully extracted the identically padded sequential units from the corpus itself. Now the vectorization process can utilize the segmentation process further. This method aids in determining the quality of the generated matrix-vector representations. This works well with comparable words that are near to terms with varying degrees of resemblance.

Instead of applying any pretrained vectorized model, we manually vectorize and arrange the vectors for respective tokens, maintaining the tokenizer_word_index listing. This manual vectorization process accumulates the frequency distribution of $range(k_{ij})$ with row-column padding within the evenly spaced Vs. Sentences set and generate a sparse matrix of vectors. We take row and column vectors for similar tokens and create a dot multiplication product for each

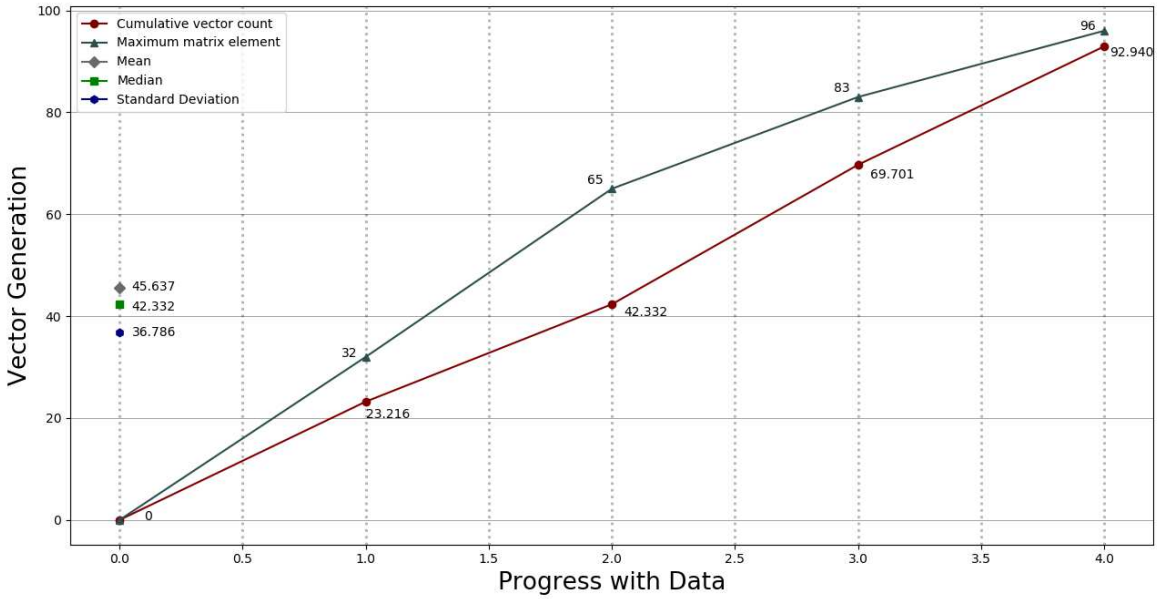

Fig. 3: Contrast between actual vector generation progress vs. maximum vector weight encountered in each quarter.

element summing up the results. Here, the dot product $token\_k_{ij}$ between two vectors t_{ixr} and t_{jxc} is defined as:

$$k(t_{ixr}.t_{jxc}) = \sum_{i=0,\ j=N-1}^{N} t_{ij} = i_1j_1 + i_2j_2 + ... + i_nj_n \ \epsilon \ V_s \tag{6}$$

From eq. (6), we can depict that our word-converted vector representations $k(t_{ixr}.t_{jxc})$ are distributed closely in a common plane V_s, with a close linear interrelation with each other by the form of summation. Now, to accumulate the distributed vector representations into a condensed matrix form, we closely follow a similar previous experimentation [37]. Since it has already been observed that, while in a document, paragraph vectors are unique within phrases or paragraphs, although the word vectors are shared, which we have already presented earlier in the form of frequency. For the convergence of the word vectors, we perform a similar regression-based analysis to work described above with a prediction probability of a scale of 10 supporting the occurrence of vectors. We initially distinguish our corpus in four quarters of word vectors, analyzing the total prediction as cumulative vector count, maximum matrix element vector, and finally, cumulative vector averaging the entire corpus as mean, median, and standard deviation for the inconsistency difference. We showcase our results in Table 3.

Table 3: Vectorization index with the countability of weighted vector generation with every quarter of data size.

Index	**Cumulative vector count**	**Max. matrix element**
25%	23.216	32
50%	42.332	65
75%	69.701	83
100%	92.940	96
Mean	45.637	
Median	42.332	
Standard deviation	32.786	

In the table, we normalize the total vector count for each quarter of the corpus in a range within 100. Also, we examine the maximum matrix element found in that respective quarter of the data. For a comparative analysis, we represent the ranging perspective in Figure 3.

4.3 Parallel LSTM (pLSTM) Architecture

We propose four equipollent long-short-term models with the same sizes as our baseline model. They are referred to as a combined pLSTM network, but each output is distinct. Each LSTM network has distinct end classifier functions at the output. These classifiers, also known as activation functions, assist us in understanding the side-by-side comparison of their behavioural characteristics while dealing with big textual produced vectors. A particular activation function can help make an LSTM model perform better than the rest when it is patched with its respective weighted sum combination and bias summarization in the outermost classification layer. The padded vector matrices from the preceding phase are channelled in a four-parallel fashion as the input feeding for each associated model. Each of these models is deep, with fully linked layers, yet they are architecturally segregated from one another. Each of the suggested models has bidirectional signalling inside its layer(s), with a backward LSTM pass and a forward LSTM pass for backpropagation, with the output gate acting as the input for the following layer. Here, the forward layer passes $F_{seq}(\rightarrow)$, the forward sequence is responsible for the iterative calculation of the inputs from the previous layer in time T-1. As the layers progress with the inputs, it is iterated for more cycles for each cell, while the time exhaustion gradually increases. Similarly, in the backward layer, pass $R_{seq}(\leftarrow)$, or reverse sequence is responsible for propagating the reverse inputs with the same amount of time. This process occurs in each of the separate models without intercepting each other's computation patterns. Within the LSTM layers, each cell contains the respective input as x_n and an inner-layer output as on. The state of the previous cell output is stored within the c_n gate. As mentioned before, there is also an additional gate within each unit of LSTM in a network, which is termed the forget gate or f_n. The forget gate provides dynamicity for sequential learning problems. It keeps the relevant information, passes it to that respective layer's output state, and discards the irrelevant, insignificant information. The insignificance is determined by how impactful the information could turn out for the successive layer or not.

A forward and backward pass within an LSTM layer can be more formally standardised by the equation: Incoming input into a layer is represented as follows in eq. 7:

$$i = \phi(W_i x_i + U_i(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_i) \tag{7}$$

Where ϕ signifies the corresponding network's activation function, W is the input layer's weight vector. The updated combination of sequence signals is indicated by U, and the bias vector for the output summarization specific layer is marked by b.

Similarly, we show the output generation from a layer in 8:

$$o = \phi(W_o x_o + U_o(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_o) \tag{8}$$

The forget gate of a layer could be expressed as in eq. 9:

$$f = \phi(W_f x_f + U_f(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_f) \tag{9}$$

Finally, the computation cell accepts the forget gate f's output quality and concurrently maintains or forgets the preceding layer's input qualities. Similarly, it receives the input from the input gate I and channels it as the new computational memory, denoted as c. It then adds the two results to get the total computation memory c. As a result, the cell operation of computation could be depicted from eq 10 as:

$$c = (f(W_n + U_n).c) + (i_n.c\,) \tag{10}$$

These are the sample models of each layer of a single long-short term network. As the layering density increases, these equational simulations will be aggregated for each layer. Consecutively, as already mentioned, we are using four sets of parallel LSTM models getting the identical inputs channelized from the padded vector matrices. From now on, we will add up the inputs for each LSTM channel in eq. 11 as follows:

$$\begin{aligned} i_{1\ldots n} &= [\phi(softmax)(W_i x_i + U_i(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_i)] + \\ i_{2\ldots n} &= [\phi(sigmoid)(W_i x_i + U_i(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_i)] + \\ i_{3\ldots n} &= [\phi(relu)(W_i x_i + U_i(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_i)] + \\ i_{4\ldots n} &= [\phi(tanh)(W_i x_i + Ui(F_{seq}(\rightarrow) + R_{seq}(\leftarrow)) + b_i)] \end{aligned} \tag{11}$$

Where, the number of intrinsic layers within each LSTM structure is denoted by n. Similarly, within the unified architecture, the output cumulation of fragments $o_1 \rightarrow o_4$, forget gates $f_1 \rightarrow f_4$, and $c_1 \rightarrow c_4$ are summarised for each layer.

We represent the architecture of our proposed model in Figure 4.

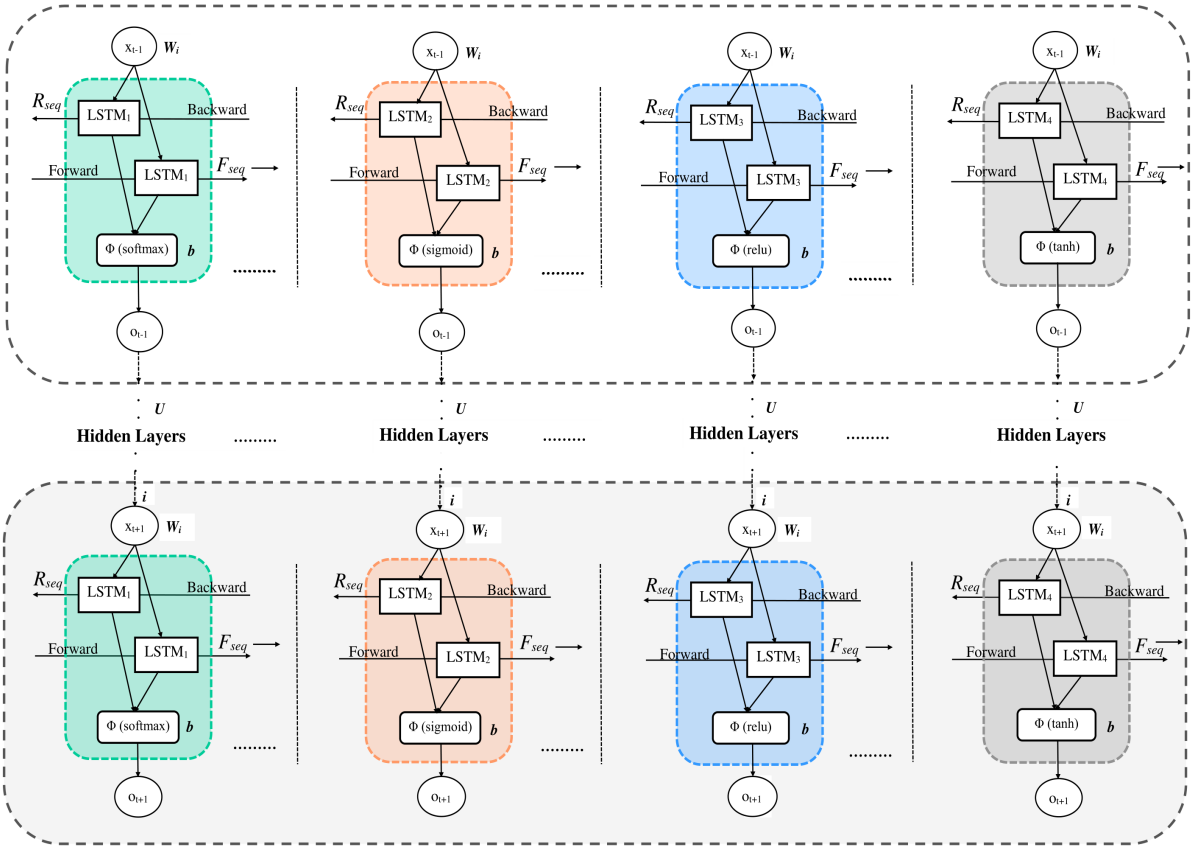

Fig. 4: Proposed architecture for parallel long-short term network.

5 Experimental Settings

5.1 Setup

We utilise Google Colab[4] for our experiments. It is a Python notebook framework that is cloud-based and supports Python versions higher than 3. It also contains a native feature known as a hardware accelerator, which allows for faster execution speed but comes with a resource limit. Colab gives users the option of using a dedicated graphics processor environment based on Tesla K80 GPUs or Google's tensor processing units, which are designed to perform parallel neural computations concurrently. Although customers have the option of adopting one of the previously mentioned options, we chose none because using the GPU or TPU accelerator creates a significant delay in resource allocation for changing usage limitations and hardware availability. We additionally link to a hosted runtime with enough of abstract RAM and disc space because Google does not supply accurate figurative details.

5.2 Hyperparameters Tuning

We use the training dataset without any partitions for broadening the scope of learning from the features. However, the evaluation metrics are testified on four more development corpora from sarcasm or satire genres. These metrics are labialized uniformly based on F1-measure and classification reports. The dense layering holds the activation classifiers, different for each substructure. We initialize the Adam optimizer with a loss function to evaluate the training loss. The dropout rate for preventing overfitting in the vectorization and bidirectional layers is 0.6 and 0.4, respectively. We sequentially fit the model inside the padded vector-matrix in the x and y-axes. Each LSTM module's epochs were set to 500. The verbose information is kept as 1 for word (vector) to the training logs. **The details of the various hyperparameters used in our experiments are shown in Table 4.**

[4] https://colab.research.google.com/

Table 4: Details of hyperparameters used in our experiments.

	pLSTM Training Hyperparameters	**pLSTM Validation Hyperparameters**
Word Embedding dimensions	400	NA
Hidden Layers	500	500
Activation Classifier	4 channels	4 channels
Optimizer	Adam	Adam
Learning Rate	0.01	0.01
State Size	128.96	128.96
Epochs	500	500
Batch Size	32	32
Data Partition	0.66	0.34

6 Results and Analysis

6.1 Cross Activation Performance Evaluation

We begin by contrasting the performance of four pLSTM modules with different activation classifiers. The comparison is performed after all of the modules have reached the designated peak training locus, i.e., on the 500th epoch. Figure 5 depicts a scalable comparison graph.

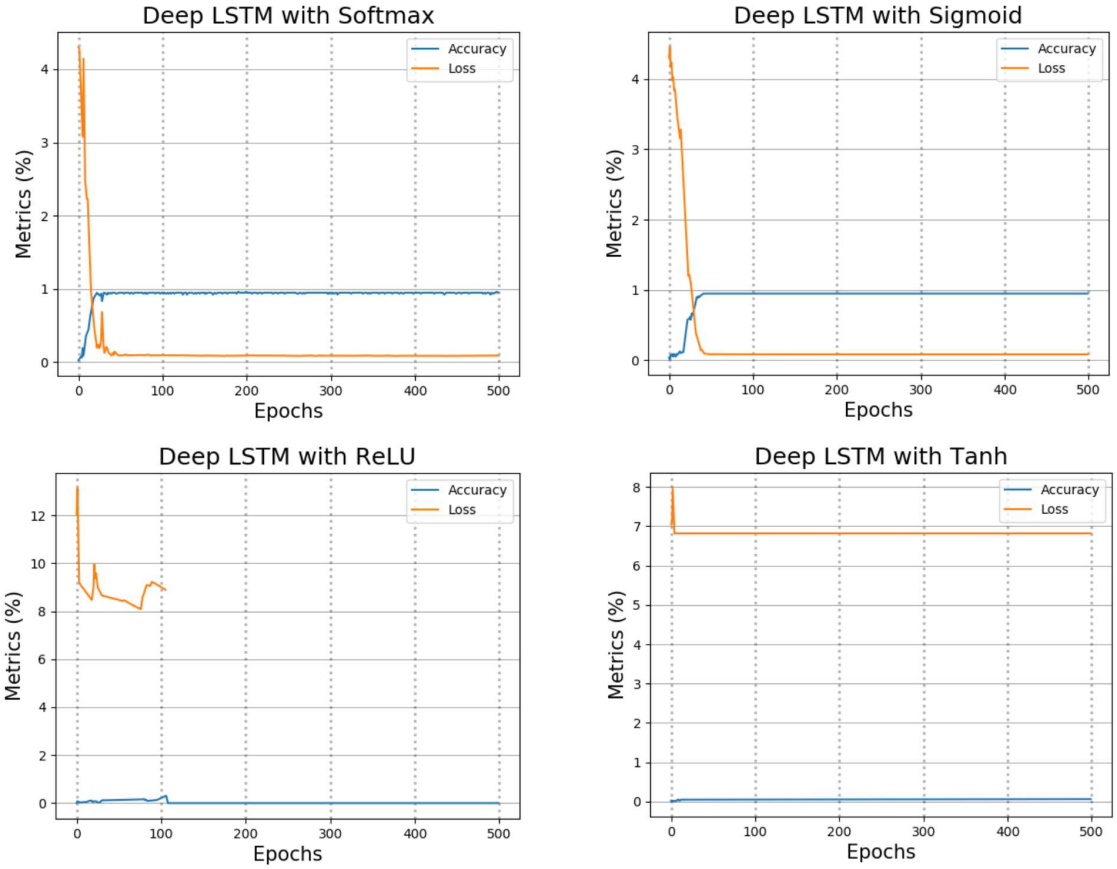

Fig. 5: Epoch level analysis of four LSTM modules with training loss and accuracy.

From Figure 5, Even before the first 100 epochs, we can see that our baseline LSTM module with softmax activation function reduces training error to a minimum. During this process, the training accuracy also reaches a high of over 90%. As a result, between 0 and 100 epochs, there is an overlap in which the error rate decreases while the precision increases. Meanwhile, the accuracy rises to nearly 98.95%, and for the remainder of the training process, it mainly stays inside the

high accuracy range of 96% -98%. Similarly, the LSTM combined with sigmoid performs well, but closer inspection reveals that it only achieves 96.88% accuracy, falling short of the first module. LSTM modules combined with ReLU and tanh, on the other hand, suffer from significant data loss during training and perform significantly worse. Because the loss values are always high, the training success for basic sarcasm classification does not proceed very far. Each module receives the same vectorization inputs that have been pretrained. As a result, in multiclass and aspect-based classification tasks, the softmax feature attached as the output classifier with an LSTM network performs better than the others [38]. At least for the sarcasm classification, this trend contradicts the previous finding that tanh had better training results than sigmoid or softmax [39].

Table 5: Summary data for training sets, where set denotes the number of epochs every 100 cycles. The effective identification of sarcastic or non-sarcastic conversations and monologues from data is referred to as *S/NS*.

Classifiers	**Set**	**S/NS**	**Set**	**S/NS**	**Set**	**S/NS**	**Set**	**S/NS**	**Set**	**S/NS**
pLSTM + softmax	100	93.07	200	92.15	300	95.33	400	97.79	500	**98.95**
pLSTM + sigmoid	100	91.12	200	90.03	300	92.50	400	94.52	500	96.88
pLSTM + relu	100	21.27	200	12.20	300	11.16	400	11.90	500	10.63
pLSTM + tanh	100	10.60	200	09.23	300	09.11	400	08.25	500	08.16

The softmax classifier outperforms the other modules in the comparative scale of performance momentum gained per 100 epochs training phase. The training analysis for epochs is reported in table 5. The visibility of results aids in narrowing down the F-measure study. Followingly, we discard the train sets of both LSTM coupled with ReLU and tanh. Table 6 compares the first two classifiers combined modules in more detail. The output represents the classification report produced during the training process.

Table 6: Comparative analysis of the best performing pLSTM modules in terms of classification report.

Module	**Sarcasm**			
	Precision	**Recall**	**F1-Score**	**Accuracy**
pLSTM + softmax	0.9900	0.9800	0.9851	0.9850
pLSTM + sigmoid	0.9600	0.9400	0.9505	0.9500

To better understand the output information, we further scrutinize the identical fine-grained training features from the data. These are accessed by both the better performing modules throughout the epoch cycle. The exact details of the training dependent variables are represented in Table 7.

Obviously, despite having similar layer configurations for both, the LSTM network with softmax classifier accumulated a significantly higher number of trainable parameters to train. It allowed for better feature learning and ultimately led to improved training accuracy.

Table 7: Layer-wise parametric details of the better performing pLSTM modules.

Classifier	**Layer**	**Output Shape**	**Parameter**
pLSTM w. softmax	Embedding	(None, 64, 100)	203200
	Bidirectional	(None, 400)	301200
	Dense	(None, 2032)	611632
	Total parameters	1,116,032	
	Trainable Parameters	1,116,032	
	Non-Trainable Parameters	0	
pLSTM w. sigmoid	Embedding	(None, 64, 100)	203200
	Bidirectional	(None, 400)	160800
	Dense	(None, 2032)	408432
	Total parameters	772,432	
	Trainable Parameters	772,432	
	Non-Trainable Parameters	0	

6.2 Automated Sarcasm Generation

After a successful training phase on the corpus, for validating the training, we attach a word sequence generation piece as the outcome classifier for the pLSTM subnetworks. As a result of this, statements are directly generated after completing training. Since two modules perform poorly in training, we discard them for the final statement generation. We explicitly input two seed words for initiating the sarcasm and the next word limit. Since no sentences contain a vast collection of n words, we restrict the limit to 20.

For the following word sequence generation, we start fetching text_to_sequences from the tokenizer list, which was created after tokenization. Now following these tokens' latter forms as vectors, we collect their even further processed padding sequences (pad_sequencing), maxlen or maximum length of sequencing, and the padding mode as pre, since the padding is done here before channeling the padded vectors as the training input for the pLSTM modules. Now we model the linear stream of word predictions into a class containing the sequence list and verbose. Once the list of a word sequence is generated, we match them with the original tokenizer list for understanding the impact of the words. We show the pseudocode for our approach in Algorithm 1.

This approach provides two benefits:

- The highly distributed and impactful words from the training corpus are used as sarcastic words and kept for the statement punchline. It certainly makes the statement more naturally sarcastic.
- The low-density words are used for generic sentence completion. Hence, the entire corpus is covered for producing more diverse statements.

We keep a manual index here to counter the words filtered within the word sequence list. Now, if this index matches the shape of our prediction model, as discussed before, we start printing the words till the sentence length is reached. We successfully generate natural spoken English statements on sarcasm following the approach. At first, we generate ten sets of statements for human evaluation, although as many as required can be generated. To better understand the statements, we manually score them on a semantic scale ranging from 1 to 5. Here 1 implies the least understandable and non-sarcastic, while 5 implies the most understandable and highly sarcastic. On the other hand, 3 is implied as to the average between two endpoints and serves as quite understandable or neutral.

Algorithm 1: Generating word sequences as natural statements.

```
Input: Trained sarcasm detection capable pLSTM combined with seed words.
Output: Sarcastic statements in natural English.
START
if seed_text (←) "seed_words " then
    next_words ← length;
    for _ in range(next_words) do
        token_list ←  tokenizer.texts_to_sequences([seed_text])[0];
        token_list ← pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
    end
    predicted ← model.predict_classes(token_list, verbose=0)
else
    error_message("Word limit exceeded!")
end
output_word ← " "
for word, index in tokenizer.word_index.items() do
    generate_word_index
end
if word_index == predicted then
    output_word ← word;
    break;
    seed_text += " " + output_word;
    output(seed_text)
else
    output(null_string)
end
STOP
```

These scores are to manually rate the statements in the scope of human convenience as being sarcastic or not. Here we symbolize five sentences representing each class of score. We represent five seed word patterns initiating these statement generations in no particular order: 'Don't worry', 'He means', 'I suppose', 'I'm really', 'You sure'.

As per our manual semantic scale, we organize the statements in Table 8 generated from these initial seed words.

Instantly, two key points are observable from the generated statements. The primary one is the sophisticated punctuation usage (inverted commas and full stops) within the sentences. In the earlier stage, the punctuation features were

Table 8: Semantic scale of sarcastic statements initiated from seed words.

Seed Word	Statement Generation	Semantic Score	Significance
I'm really	I'm really sure she likes coffee better than the dictionary.	5	Highly sarcastic
Don't worry	Don't worry he's probably not doing the physics right.	4	Sarcastic
I suppose	I suppose he is even going to buy the remains of the elephant he is insane.	3	Understandable
You sure	You sure, she is going to be able to crack that code happening in california.	2	Somewhat sarcastic
He means	He means to to say good the initiative organized in last week in in	1	Non-semantic: nonsarcastic

already extracted from the corpus and matched with their adjoining words in the respective PoS tagged forms. That results in appropriate positional punctuation placements. The second observation reveals that most of the generated statements are highly fathomable and sarcastic from human perceive (ranging from 3 to 5), one is not transparently understandable and non-sarcastic (2), and the final one is a poor output (1). The statement representing score 3 can be explained either as sarcastic or random depending on the context. Expressing presumption on attempting a hypothetically impossible task is a sarcastic statement; otherwise not. It is noticeable that the preposition constructions of the least semantically correct output scoring 1 are placed arbitrarily. Since all outputs from each pLSTM module are generated as a combined set, the poor statement productions certainly hint at the subpar performing modules within the architecture. It also indicates probable mispositioned training inputs of PoS tags to vector sequence generation or hyperparameter selection for these modules.

This outcome supports the mechanisms of successful manual feature extraction done in the initial phase, combining sarcastic word distribution and IPoS from eq. (1). When channeled in sequence and padded to matrix form, these features provide good learnability of the neural network, leading to primarily successful natural sarcasm output.

6.3 Benchmark Analysis

Following that, we conduct comparable rounds of trial without the text production phase. For traditional validation testing of our proposed architecture, we picked numerous open-source and sarcasm-based datasets. It demonstrates our suggested method's ability to identify sarcasm. First, we choose two English works of humour and comedy literature from the Project Gutenberg digital library[5], such as The Comedy of Errors^{dag} (TCE^{dag}) and Three Men in a Boat^{ddag} (TMB^{ddag}). Each of these expresses linear shapes using plain text phrases, with no preprocessing required. Following that, we test our suggested model with three more sarcastic data sets collected and produced from the internet. The first is Sarcasm SIGN, a sarcastic twitter corpus provided by [40]. It is a collection of 3000 tweets in English and contains the *#sarcasm* keywords. The tweets contain an average word length of 12.10. For the sarcasm evaluation, the authors approached traditional classification measures and human judgment. Also, they took a de-clustered approach for context filtering to refine the inner layered sarcasm.

[41] developed a dataset of a combination of sarcastic and non-sarcastic data employing supervised method. The primary texts in the corpus are questions and rhetorical statements in quote and response pairs. A new pair of quotes and replies are added to the data in the manual annotation process when 66% of the human annotators labeled it as sarcastic. The summary analysis of the corpus achieved an F measure of 74 while incorporating the n-grams and Word2Vec features for lexical property inspection.

The final public corpus we select is a collection of sarcasm data from Reddit posts (SARC), represented by [42]. It is a manually annotated corpus consisting of eight years of 1.3 million sarcastic Reddit posts, replies, and comments. Here only the original post could be sarcastic, or even the sequence of comments and replies can lead to a persuasive sarcasm thread. The researchers further evaluated the data by baseline classification parameters to determine between sarcastic and non-sarcastic contents. For a combination of both types of data, their balanced majority human evaluation of detecting sarcasm within the corpus reaches a score of 92. Further analysis using a rule-based bag of words and sentence embeddings represents the contrast of low dimensional sparse data representation.

We shuffle the data in a 3:2 ratio by doing 5-fold random cross-training and validation on each dataset. The testing is then repeated on the full corpus. Table 9 demonstrates that, among all pre-trained modules, our baseline pLSTM with softmax has the greatest validation accuracy of 98.31%, gaining a margin of 2.27% over the next best performing

[5] https://www.gutenberg.org/

Table 9: *L* denotes the number of lines present, *V* represents the vocabulary length, *Size* represents the training and test size in kb of the relevant data, and *SoA* reflects the corresponding state-of-the-art findings observed. *(tr + ts)* denotes the combination of training and test data. Where no results have yet been reported, dashed lines are used.

Data	L	V	Size	Model	Train	Test	SoA
TCE†	2882	18020	100	pLSTM + softmax	98.90	**98.31**	-
				pLSTM + sigmoid	98.16	96.04	
				pLSTM + relu	40.81	37.63	
				pLSTM + tanh	38.11	36.92	
TMB‡	7703	69849	382	pLSTM + softmax	97.70	**96.93**	-
				pLSTM + sigmoid	96.00	95.40	
				pLSTM + relu	42.51	40.10	
				pLSTM + tanh	41.72	39.83	
SIGN [40]	14970	410979	2076	pLSTM + softmax	97.05	**95.89**	-
				pLSTM + sigmoid	95.96	94.09	
				pLSTM + relu	46.67	44.32	
				pLSTM + tanh	46.29	45.90	
Sarcasm V2 [41]	4693	433270	2568	pLSTM + softmax	95.37	**94.06**	76.00 [43]
				pLSTM + sigmoid	93.20	91.54	
				pLSTM + relu	32.19	30.24	
				pLSTM + tanh	29.07	27.88	
SARC (tr + ts) [42]	160874	514762	4646	pLSTM + softmax	94.91	**93.00**	77.00 [44]
				pLSTM + sigmoid	91.02	89.43	
				pLSTM + relu	38.12	36.00	
				pLSTM + tanh	32.59	31.06	

substructure of pLSTM with sigmoid (96.04%). Not only that, but both the softmax and sigmoid connected modules get F scores of 97.03% and 94.58% for TCE-based validation, respectively, which is the highest among the benchmark study. For [41] and [42], our suggested technique beats the prior state-of-the-art findings [43] and [44] by a significant improvement leap.

As previously shown, we employ two of our best performing baseline modules, pLSTM + softmax and pLSTM + sigmoid, to illustrate the performance comparison with analogous tasks for sarcasm, satire, and/or irony detection.

Our observation maintains several linear scales of comparative parameters, such as the proposed approach by other established works, validation accuracy, and overall F measures. Table 10 represents the contrast between our work and previous works. We concisely discuss the works that we select here for benchmark comparisons. [45] introduced the concept of contextual inconsistency when the polarity determining words are absent in a sentence. To overcome this issue, they showed a combination of similarity features of sentences and weighted sentiments (WS). In [46], the researchers focused on two sequence labeling algorithms for sarcasm detection in text-based dialogue corpus from a popular comedy show. Among them, the support vector algorithm with a hidden Markov model for better sequence tagging (SVM-HMM) outperforms previous rule-based statistical classification algorithms. [47] proposed a convolutional network for learning content and context word embedding presentations. This model was proposed to eliminate the need for manual feature engineering in the utterance level of natural English texts. Their CUE-CNN model on sarcastic tweets improved accuracy over the previous baseline approach. [48] proposed a computational model built around the linguistic, pragmatic, and contextual features (L+P+C) of short phrases. They combined the features for covering every possible sarcasm utterance aspect within sarcastic tweets. Their analysis results in better results when compared to the baseline approach with similar but fewer features in consideration. A group of researchers came up with an emotion-based sarcasm detection system [49]. They primarily constructed the Japanese sarcasm model and tackle ambiguous expressions. Later they validated their model with English texts. Their manual sarcasm evaluation (S-E Detection) scores a significant improvement when the polarity of an expression is determined first, and then the sarcasm is apprehended. [50] experimented with the context of sarcasm, keeping the priority as a particular passage within a sarcasm document contains the primary sarcastic influence or punch. It can influence the other portion of the document, irrespective of being positive or normally negative in nature. The authors proposed a linear support vector algorithm (SVM linear) driven from the native Python machine learning library and observed a better performance on overall bipolar (positive and negative) user reviews from Amazon. Similarly to our approach, [51] inspected the roll of sarcasm context of conversations in social media discussions, implying the tweets and replies on Twitter. They worked with a support vector model with manual features and a combination of attention LSTMs compared to human performance in similar tasks. Among all the models they tested on sarcastic tweets, and discussion forum data from [41], the combination of context-attention LSTM and reply-attention LSTM models (LSTM cas + LSTM ras) tends to perform the best. [52] came up with the concept of synchronization of the sarcastic mood of a Twitter user who depends on the topic and context in which they are tweeting. They collected sarcastic tweets from a specific list of users using bot software. The authors combined a variety of alterations of neural networks with context modeling to test out their approach to their data and several public data.

Table 10: Comparative results on standard benchmarks for sarcasm or irony detection on similar public datasets. Here D/T_sv indicates the test vocabularies on which the respective models were tested. Either the works reported *F* measure as the performance scale or the overall accuracy. Dashed lines are introduced where the other counterparts for benchmark were not reported. Some works, including that of ours, reported both the scales.

Year		Model	D/Tsv	Acc.	F1
2016	1	S + WS [45]	GloVe embeddings	-	84.50
	2	SVM-HMM [46]	Pop culture sitcom	-	84.40
	3	CUE-CNN [47]	Sarcastic tweets	87.20	-
	4	L + P + C [48]	Sarcastic tweets	78.70	-
2017	1	S-E Detection [49]	Amazon reviews	-	63.00
	2	SVM (linear) [50]	Amazon reviews	-	78.90
	3	$\text{LSTM}^{\text{ca}_\text{s}}$ + $\text{LSTM}^{\text{ra}_\text{s}}$ [51]	Discussion forums + Sarcastic tweets	-	76.36
	4	CNN1 + CNN2 + LSTM1 + LSTM2 + EAW + CL + DNN [52]	Sarcastic tweets	-	90.00
2018	1	CASCADE [53]	SARC	79.00	86.00
	2	$\text{LSTM}^{\text{ct}_\text{as}}$ + $\text{LSTM}^{\text{pt}_\text{ps}}$ [54]	IAC + Sarcastic tweets + Reddit posts	-	76.36
	3	UCDCC [55]	Ironic tweets	79.70	72.40
	4	Dense-LSTM + ens [56]	Ironic tweets	-	70.54
2019	1	BiLSTM + slf [57]	Book snippets + Sarcastic tweets	88.16	-
	2	E-b S Detection [58]	Facebook pages + Facebook reactions	70.47	56.14
	3	IN-W-CASCADE [59]	Riloff + Ptacek data	-	93.40
	4	E-S Irony Detection [60]	Twitter emoji sentiments + Bipolar tweets	-	82.00
2020/21	1	ENS [61]	SARC + IAC + Sarcastic tweets + Reddit	-	74.00
	2	MHS-BiLSTM [18]	SARC	-	77.48
	3	MHA + GRU [62]	SARC + Sarcasm V2	81.00	81.00
	4	iSarcasm [63]	Intended Sarcasm Dataset	79.30	78.09
	5	BERT-Base + GRU [64]	SARC + Sarcasm V2	81.00	81.00
	6	Ours (pLSTM + softmax)	Project Gutenberg - TCE	**98.31**	**97.03**
	7	Ours (pLSTM + sigmoid)	Project Gutenberg - TCE	**96.04**	**94.58**

Since online discussion forums like Reddit hold rich sarcastic threads with posts, comments, and replies, the content of the main post and the context of the comments are both equally important. Hazarika et al. addressed this issue with their CASCADE model [53]. They manually introspect user profiles for their tendencies linked to sarcasm on such a platform. After extracting such information, with stylometry, they propose a convolutional model to analyze the syntactic features and evaluate the performance of the proposed approach. In the following work [54], researchers took a similar approach to conversations on Twitter and Reddit. They indicated a prior turn that initiates the sarcastic turn of comments. The LSTMs with prior turn and sarcastic replies with sentence-level attention achieve the best performance among their distinct proposed models. For detecting irony in tweets, [55] took a two-phase approach. At first, they made a bipolar classification of whether the tweet was ironic or not. Then they proceeded to the multiclass variation of the irony itself; the irony's situation, verb, and context. They made a simple classification approach for detecting the same manually annotating such ironic tweets. [56] proposed a dense LSTM for multitask learning from textual ironical data; Their method features the learnability of whether a tweet is ironic, identifying ironic hashtags within tweets, and the type of irony. Their performance evaluation on ground truth algorithms such as SVM, CNN, and the BiLSTM model revealed that their dense LSTM+ens for ensemble multiclass learning achieved a better F score for irony detection. Since sarcastic expressions are not the same in all languages but rather highly differ, determining the linguistics and social features of a topic also plays a role. [57] exploited this perspective. They made an empirical study and modeled their proposed deep learning framework around such features. They considered book snippets containing humorous phases and sarcastic tweets as their data.

They achieved significantly better performance with their sociolinguistic features combined with LSTM (BiLSTM + slf) than the previous statistical baseline methods. [58] utilized the Facebook reactions to understand people's tendencies of being sarcastic about any particular post. They made an empirical study on this and made a performance evaluation with the previous standard emotion-based sarcasm detection approaches. These researchers collected Facebook reactions from several popular pages. They made a study on people who react to a particular post, comment accordingly on the same, and often both the sentiment expressions match each other. [59] proposed user embedding features with the neural network model. They approached two previous sets of corpora with several fine-tuned established models. Summarizing the performances, they claimed to achieve state-of-the-art results on one corpus set, which we draw here

for relative comparison. Often in social media, sarcasm or irony, we use emojis and emoticons in our posts but carry opposite sentiment flavors. This is to express sarcasm via emojis. It also reflects the polarity contrast of a sentence. [60] analyzed such behaviors of emojis in ironic tweets. They manually matched the sentiment of emojis with tweets and then used a BiLSTM network with logistic regression classification for output prediction. [61] approached sarcasm detection in Reddit and Twitter, considering the sentiment, length, and source of posts and comments. They explored an alteration of various deep neural models while adding manual features like stylometry, emoji, hashtags of original posts, and emotional lexicon embeddings. In the final work for our comparative observations, [18] studied the manual features of sarcastic phrases and further processed those features with a multi-head self-attention combined BiLSTM (MHS + BiLSTM). They compared their work with several previous baseline models with traditional classifiers and achieved a better system for sarcasm detection. **Akula and Gabaray [62] discusses how to find the natural explainable sarcasm from social media textual conversations. They developed an ensemble of BERT [34] with Gated Recurrent Units and evaluated their model on the SARC and Sarcasm V2 datasets. Their proposed approach consists of five parts: Information Pre-handling, Multi-Head Self-Consideration, Gated Repetitive Units(GRU), characterization, and model interpretability. Information pre-handling includes changing information text over to word embeddings, needed for preparing a profound learning model. For this, they utilized the pre-prepared language model, BERT, to remove word embeddings. They implemented these word embeddings, which catch worldwide settings. These embeddings structure the contribution to the proposed multi-head self-consideration module, which distinguishes words in the information text that give vital signs to mockery. In the subsequent stage, the GRU layer supports learning significant distance connections among these featured words and, result, a solitary element vector encoding the whole grouping. Finally, a wholly associated layer with sigmoid actuation is utilized to get the last grouping score.**

In the following work, [64], the authors combined their earlier approach with the multi-headed attention mechanism to efficiently identify the contextual sarcasm. They have kept the fully-connected single-layered Gated Recurrent module for the classification task. They fostered an interpretable profound learning model utilizing multi-head self-consideration and gated repetitive units. The multi-head self-consideration module supports distinguishing key snide prompt words from the info. The repetitive units learn long-range conditions between these sign words to arrange the information text more readily. They further demonstrated the adequacy of their proposed approach by accomplishing cutting-edge outcomes on different datasets from long-range informal communication stages and online media. Models prepared to utilize their suggested approach are effectively interpretable and empower recognizing snide signals in the information text, which add to the last order score.

As the observation can be drawn from Table 10, for a similar group of tasks, our model outperforms several state-of-the-art approaches from the past and current years. All works for comparison have been examined on open-sourced public text corpora. Our approach for comparison here is a parallel and non-overlapping dense BiLSTM model for successful sarcasm detection, capable of taking inputs as vector sequences generated from both sarcastic and non-sarcastic words. We believe the intricate features extracted from the data, such as the Sarcastic Word Distribution, Sentiment Features, and Punctuation Features, provide quality auxiliary training inputs. These inputs, attached with padded sequences, help achieve high training accuracy. It has also allowed the proposed model to sequential learning, ultimately generating the sarcasm context in the text generation part. Although two of our substructures perform subpar and fall below average in comparison, the better performing modules perform excellent results in sarcasm detection and after that generation. Our model scores at least 11.03% better F measure score and 10.15% in overall accuracy than the previous best work within the comparison [53], [57]. We also obtain at most as 40.89% performance in a similar F measure scale, and 27.84% betterment in overall accuracy than the least performing models from the lot [58]. We also achieve uninterrupted peak stability between the better performing modules throughout the training phase; combining it to a slow learning rate helps avoid overfitting and enhances the learnability of the model. Figure 6 presents the accuracy and/or F1 scores from the comparative analysis with the benchmark models from each year. Here we show the best-reported scores across among the comparisons. However, we cannot depict the general accuracy score of any of the approaches from 2017, as it is not postulated in any work within our findings from that year.

6.4 Discussion

Our suggested architecture is capable of detecting and producing successful sarcasm generation based on training epochs. Our architecture's LSTM modules function independently, pulling from the same input sets. The functioning of one of these modules has no bearing on the others. This makes it possible to track the individual performance of each independent module. Furthermore, because modules are substructures of a cohesive architecture, the tuning of hyperparameters and the simulation environment are the same for all modules. Finally, the statements are formed as a single set by integrating the separate outputs of all the modules. Our strategy, in addition to maintaining overall constant performance, misses out on specific occurrences. From the first set of ten sarcastic statements generated for manual

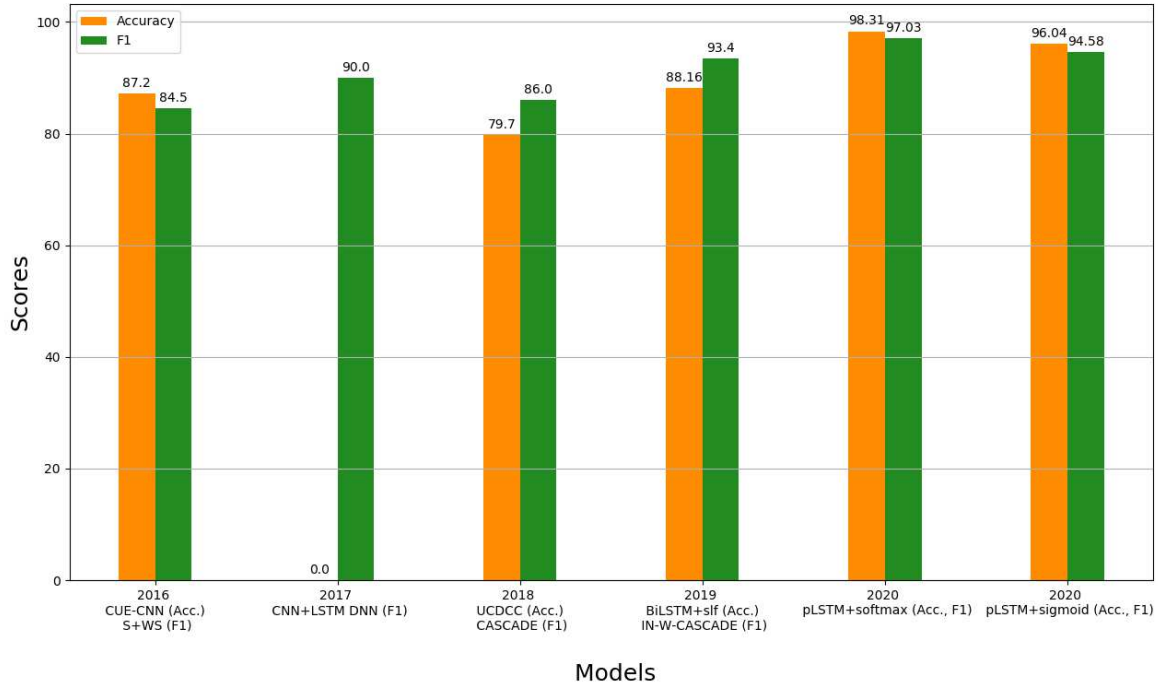

Fig. 6: Overall accuracy and F measure standings of individually better performing works from each year.

validation, two outputs are semantically and grammatically incorrect and non-sarcastic. We have already discussed one such statement in subsection 6.2. Even if we try to re-write ”He means to say a good initiative organized in the last week in in” or ”We know how all proud you with are your game collection” correctly, it would still semantically mean no sense. In these cases, our method, particularly an independent module(s) of the architecture, repetitively failed to learn the prepositions and adverbs.

7 Conclusion and Future Scope

Sarcasm leads to several emotional expressions. Even for us humans, it might be difficult to grasp at times. Although sarcasm can be shown without any revelation of expressions, in our daily conversations, sarcasm can be detected with the help of other characteristics. These are the voice tonality, facial expressions, and eyebrow movements. When dealing with linguistics data only, we do not have such features for consideration. That being said, we have additional language-specific differentiating traits in that language's syntax, semantics, and lexicon. We have concentrated on these features while making a sarcasm detection framework and extending further to generate natural human-alike sarcasm successfully.

We chose an open-sourced text corpus manually built to collect famous comedy show dialogues. We categorized sarcastic word distributions, their sentiment impacts, and their respective PoS tagged forms that can be extracted from such corpus, and we demonstrated the outcome of each such phase. We even examine the punctuation positions of the dialogues within the corpus. This rigorous data fragmentation is a good quality input for our parallel deep neural networks. We designed this architecture with the idea of how the homogeneity of one such network module would compare to the homogeneity of another when given the same inputs, equal tuning, but different classifiers at the outermost layers. Two of the independent modules outperformed our expectations on the training data, obtaining a maximum total accuracy of 98.95%. We then validated the training by providing seed words for initiating sarcastic statements. At the same time, the model chooses the words as per their impact on the corpus itself, learned from the frequency distributed and tokenized inputs. This approach mainly produces good-quality sarcasm, with an average of eight out of ten sarcastic statements. These statements are sophisticated, semantically correct, and successful in mimicking human uttered sarcastic sentences. We also used prominent public sarcasm corpora to test our framework, and two of our independent modules scored above 95% accuracy in recognising sarcasm, with the best accuracy being 98.31%.

However, we would like to address some limitations for our future extension of this work. Instead of deploying four deep neural models to achieve a homogeneous sarcasm detection task, we aim to develop a real-time dialogue-based system for a natural conversation. There are a few areas of impact for sentence-level sarcasm, which can be

identified with the help of sentence attention mechanisms. We look forward to attaching it to our existing network to even additionally fine-grain the sarcasm. We're also interested in establishing a sarcastic language entirely generated by neural networks using a compelling model of textual interaction that includes humour, wit, and irony.

Conflict of interest

The authors declare that they have no conflict of interest.

Data Availability Statement

The datasets analyzed during the current study in subsequent order are openly available as:

1. **MUStARD or Multimodal Sarcasm Detection** dataset in the **Github** repository, https://github.com/soujanyaporia/MUStARD
2. **The Comedy of Errors** in the **Project Gutenberg** repository, https://www.gutenberg.org/ebooks/2239
3. **Three Men in a Boat** in the **Project Gutenberg** repository, https://www.gutenberg.org/ebooks/308
4. **Sarcasm SIGN** dataset in the **Github** repository, https://github.com/lotemp/SarcasmSIGN
5. **Sarcasm V2** dataset in the **JB School of Engineering** NLDS Corpora repository, https://nlds.soe.ucsc.edu/sarcasm2
6. **SARC** dataset in the **Princeton CS** repository, https://nlp.cs.princeton.edu/SARC/2.0/main/
7. **The source-code of the project will be made available at:** https://github.com/SouravD-Me

References

1. S. Mukherjee, P.K. Bala, Technology in Society **48**, 19 (2017)
2. A. Avvaru, S. Vobilisetty, R. Mamidi, in *Proceedings of the Second Workshop on Figurative Language Processing, Fig-Lang@ACL 2020, Online, July 9, 2020*, ed. by B.B. Klebanov, E. Shutova, P. Lichtenstein, S. Muresan, C.W. Leong, A. Feldman, D. Ghosh (Association for Computational Linguistics, 2020), pp. 98–103. DOI 10.18653/v1/2020.figlang-1.15. URL https://doi.org/10.18653/v1/2020.figlang-1.15
3. H. Gregory, S. Li, P. Mohammadi, N. Tarn, R.L. Draelos, C. Rudin, in *Proceedings of the Second Workshop on Figurative Language Processing, Fig-Lang@ACL 2020, Online, July 9, 2020*, ed. by B.B. Klebanov, E. Shutova, P. Lichtenstein, S. Muresan, C.W. Leong, A. Feldman, D. Ghosh (Association for Computational Linguistics, 2020), pp. 270–275. DOI 10.18653/v1/2020.figlang-1.37. URL https://doi.org/10.18653/v1/2020.figlang-1.37
4. D. Maynard, M.A. Greenwood, in *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, ed. by N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, S. Piperidis (European Language Resources Association (ELRA), 2014), pp. 4238–4243. URL http://www.lrec-conf.org/proceedings/lrec2014/summaries/67.html
5. O. Tsur, D. Davidov, A. Rappoport, in *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*, ed. by W.W. Cohen, S. Gosling (The AAAI Press, 2010). URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1495
6. D. Davidov, O. Tsur, A. Rappoport, in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning, CoNLL 2010, Uppsala, Sweden, July 15-16, 2010*, ed. by M. Lapata, A. Sarkar (ACL, 2010), pp. 107–116. URL https://aclanthology.org/W10-2914/
7. A. Onan, in *Artificial Intelligence Trends in Intelligent Systems - Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 1*, *Advances in Intelligent Systems and Computing*, vol. 573, ed. by R. Silhavy, R. Senkerik, Z.K. Oplatková, Z. Prokopova, P. Silhavy (2017), *Advances in Intelligent Systems and Computing*, vol. 573, pp. 374–383. DOI 10.1007/978-3-319-57261-1_37. URL https://doi.org/10.1007/978-3-319-57261-1_37
8. R.I. González-Ibáñez, S. Muresan, N. Wacholder, in *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers* (The Association for Computer Linguistics, 2011), pp. 581–586. URL https://aclanthology.org/P11-2102/
9. E. Filatova, in *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, ed. by N. Calzolari, K. Choukri, T. Declerck, M.U. Dogan, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis (European Language Resources Association (ELRA), 2012), pp. 392–398. URL http://www.lrec-conf.org/proceedings/lrec2012/summaries/661.html
10. R. Rakov, A. Rosenberg, in *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, ed. by F. Bimbot, C. Cerisara, C. Fougeron, G. Gravier, L. Lamel, F. Pellegrino, P. Perrier (ISCA, 2013), pp. 842–846. URL http://www.isca-speech.org/archive/interspeech_2013/i13_0842.html
11. P.A. Wang, in *Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation, PACLIC 27, Taipei, Taiwan, November 21-24, 2013* (National Chengchi University, Taiwan, 2013). URL https://aclanthology.org/Y13-1035/
12. A. Onan, in *Software Engineering Methods in Intelligent Algorithms - Proceedings of 8th Computer Science On-line Conference 2019, CSOC 2019, April 2019, Vol. 1*, *Advances in Intelligent Systems and Computing*, vol. 984, ed. by R. Silhavy (Springer, 2019), *Advances in Intelligent Systems and Computing*, vol. 984, pp. 293–304. DOI 10.1007/978-3-030-19807-7_29. URL https://doi.org/10.1007/978-3-030-19807-7_29

13. A. Onan, M.A. Toçoglu, Turkish J. Electr. Eng. Comput. Sci. **28**(2), 1086 (2020). DOI 10.3906/elk-1907-11. URL https://doi.org/10.3906/elk-1907-11
14. E. Riloff, A. Qadir, P. Surve, L.D. Silva, N. Gilbert, R. Huang, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL* (ACL, 2013), pp. 704–714. URL https://aclanthology.org/D13-1066/
15. A. Reyes, P. Rosso, T. Veale, Language resources and evaluation **47**(1), 239 (2013)
16. D. Das, A.J. Clark, in *Proceedings of the International Conference on Multimodal Interaction: Adjunct, ICMI 2018, Boulder, CO, USA, October 16-20, 2018* (ACM, 2018), pp. 3:1–3:5. DOI 10.1145/3281151.3281154. URL https://doi.org/10.1145/3281151.3281154
17. Y.A. Kolchinski, C. Potts, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, ed. by E. Riloff, D. Chiang, J. Hockenmaier, J. Tsujii (Association for Computational Linguistics, 2018), pp. 1115–1121. DOI 10.18653/v1/d18-1140. URL https://doi.org/10.18653/v1/d18-1140
18. A. Kumar, V.T. Narapareddy, V.A. Srikanth, A. Malapati, L.B.M. Neti, IEEE Access **8**, 6388 (2020). DOI 10.1109/ACCESS.2019.2963630. URL https://doi.org/10.1109/ACCESS.2019.2963630
19. R.A. Potamias, G. Siolas, A. Stafylopatis, in *Engineering Applications of Neural Networks - 20th International Conference, EANN 2019, Hersonissos, Crete, Greece, May 24-26, 2019, Proceedings*, *Communications in Computer and Information Science*, vol. 1000, ed. by J. MacIntyre, L.S. Iliadis, I. Maglogiannis, C. Jayne (Springer, 2019), *Communications in Computer and Information Science*, vol. 1000, pp. 164–175. DOI 10.1007/978-3-030-20257-6_14. URL https://doi.org/10.1007/978-3-030-20257-6_14
20. N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria, A.F. Gelbukh, IEEE Intell. Syst. **34**(3), 38 (2019). DOI 10.1109/MIS.2019.2904691. URL https://doi.org/10.1109/MIS.2019.2904691
21. K.T. Sundararajan, A. Palanisamy, Comput. Intell. Neurosci. **2020**, 2860479:1 (2020). DOI 10.1155/2020/2860479. URL https://doi.org/10.1155/2020/2860479
22. R.A. Potamias, G. Siolas, A. Stafylopatis, Neural Comput. Appl. **32**(23), 17309 (2020). DOI 10.1007/s00521-020-05102-3. URL https://doi.org/10.1007/s00521-020-05102-3
23. D.S. Chauhan, D. S R, A. Ekbal, P. Bhattacharyya, in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing* (Association for Computational Linguistics, Suzhou, China, 2020), pp. 281–290. URL https://aclanthology.org/2020.aacl-main.31
24. D.S. Chauhan, D. S R, A. Ekbal, P. Bhattacharyya, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics, Online, 2020), pp. 4351–4360. DOI 10.18653/v1/2020.acl-main.401. URL https://aclanthology.org/2020.acl-main.401
25. A. Onan, M.A. Toçoglu, IEEE Access **9**, 7701 (2021). DOI 10.1109/ACCESS.2021.3049734. URL https://doi.org/10.1109/ACCESS.2021.3049734
26. A. Joshi, A. Kunchukuttan, P. Bhattacharyya, M.J. Carman, in *WISDOM Workshop at KDD* (2015)
27. A. Joshi, D. Kanojia, P. Bhattacharyya, M.J. Carman, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, ed. by S.P. Singh, S. Markovitch (AAAI Press, 2017), pp. 5095–5096. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14179
28. A. Mishra, T. Tater, K. Sankaranarayanan, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), pp. 6144–6154
29. T. Chakrabarty, D. Ghosh, S. Muresan, N. Peng, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 7976–7986
30. A. Ghosh, T. Veale, in *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@NAACL-HLT 2016, June 16, 2016, San Diego, California, USA*, ed. by A. Balahur, E.V. der Goot, P. Vossen, A. Montoyo (The Association for Computer Linguistics, 2016), pp. 161–169. DOI 10.18653/v1/w16-0425. URL https://doi.org/10.18653/v1/w16-0425
31. R. Misra, P. Arora, CoRR **abs/1908.07414** (2019). URL http://arxiv.org/abs/1908.07414
32. O. Rohanian, S. Taslimipoor, R. Evans, R. Mitkov, in *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018*, ed. by M. Apidianaki, S.M. Mohammad, J. May, E. Shutova, S. Bethard, M. Carpuat (Association for Computational Linguistics, 2018), pp. 553–559. DOI 10.18653/v1/s18-1090. URL https://doi.org/10.18653/v1/s18-1090
33. S. Castro, D. Hazarika, V. Pérez-Rosas, R. Zimmermann, R. Mihalcea, S. Poria, in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, ed. by A. Korhonen, D.R. Traum, L. Màrquez (Association for Computational Linguistics, 2019), pp. 4619–4629. DOI 10.18653/v1/p19-1455. URL https://doi.org/10.18653/v1/p19-1455
34. J. Devlin, M. Chang, K. Lee, K. Toutanova, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, ed. by J. Burstein, C. Doran, T. Solorio (Association for Computational Linguistics, 2019), pp. 4171–4186. DOI 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423
35. M. Neethu, R. Rajasree, in *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)* (IEEE, 2013), pp. 1–5
36. M.A.K. Halliday, R. Hasan, *Cohesion in english*. 9 (Routledge, 2014)
37. Q.V. Le, T. Mikolov, in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, *JMLR Workshop and Conference Proceedings*, vol. 32 (JMLR.org, 2014), *JMLR Workshop and Conference Proceedings*, vol. 32, pp. 1188–1196. URL http://proceedings.mlr.press/v32/le14.html
38. Y. Tay, L.A. Tuan, S.C. Hui, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, ed. by S.A. McIlraith, K.Q. Weinberger (AAAI Press, 2018), pp. 5956–5963. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16570
39. C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, CoRR **abs/1811.03378** (2018). URL http://arxiv.org/abs/1811.03378
40. L. Peled, R. Reichart, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, ed. by R. Barzilay, M. Kan (Association for Computational Linguistics, 2017), pp. 1690–1700. DOI 10.18653/v1/P17-1155. URL https://doi.org/10.18653/v1/P17-1155

41. S. Oraby, V. Harrison, L. Reed, E. Hernandez, E. Riloff, M.A. Walker, in *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 13-15 September 2016, Los Angeles, CA, USA* (The Association for Computer Linguistics, 2016), pp. 31–41. DOI 10.18653/v1/w16-3604. URL https://doi.org/10.18653/v1/w16-3604
42. M. Khodak, N. Saunshi, K. Vodrahalli, in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*, ed. by N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, T. Tokunaga (European Language Resources Association (ELRA), 2018). URL http://www.lrec-conf.org/proceedings/lrec2018/summaries/160.html
43. S. Ilic, E. Marrese-Taylor, J.A. Balazs, Y. Matsuo, in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EMNLP 2018, Brussels, Belgium, October 31, 2018*, ed. by A. Balahur, S.M. Mohammad, V. Hoste, R. Klinger (Association for Computational Linguistics, 2018), pp. 2–7. DOI 10.18653/v1/w18-6202. URL https://doi.org/10.18653/v1/w18-6202
44. D. Pelser, H. Murrell, CoRR **abs/1911.07474** (2019). URL http://arxiv.org/abs/1911.07474
45. A. Joshi, V. Tripathi, P. Bhattacharyya, M.J. Carman, in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, ed. by Y. Goldberg, S. Riezler (ACL, 2016), pp. 146–155. DOI 10.18653/v1/k16-1015. URL https://doi.org/10.18653/v1/k16-1015
46. A. Joshi, V. Tripathi, K. Patel, P. Bhattacharyya, M.J. Carman, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, ed. by J. Su, X. Carreras, K. Duh (The Association for Computational Linguistics, 2016), pp. 1006–1011. DOI 10.18653/v1/d16-1104. URL https://doi.org/10.18653/v1/d16-1104
47. S. Amir, B.C. Wallace, H. Lyu, P. Carvalho, M.J. Silva, in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, ed. by Y. Goldberg, S. Riezler (ACL, 2016), pp. 167–177. DOI 10.18653/v1/k16-1017. URL https://doi.org/10.18653/v1/k16-1017
48. T. Bali, N. Singh, in *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, PEOPLES@COLING 2016, Osaka, Japan, December 12, 2016*, ed. by M. Nissim, V. Patti, B. Plank (The COLING 2016 Organizing Committee, 2016), pp. 119–127. URL https://aclanthology.org/W16-4313/
49. S. Suzuki, R. Orihara, Y. Sei, Y. Tahara, A. Ohsuga, in *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017*, ed. by H.J. van den Herik, A.P. Rocha, J. Filipe (SciTePress, 2017), pp. 519–526. DOI 10.5220/0006192805190526. URL https://doi.org/10.5220/0006192805190526
50. E. Filatova, in *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017*, ed. by V. Rus, Z. Markov (AAAI Press, 2017), pp. 264–269. URL https://aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15480
51. A. Ghosh, T. Veale, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, ed. by M. Palmer, R. Hwa, S. Riedel (Association for Computational Linguistics, 2017), pp. 482–491. DOI 10.18653/v1/d17-1050. URL https://doi.org/10.18653/v1/d17-1050
52. D. Ghosh, A.R. Fabbri, S. Muresan, in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, ed. by K. Jokinen, M. Stede, D. DeVault, A. Louis (Association for Computational Linguistics, 2017), pp. 186–196. DOI 10.18653/v1/w17-5523. URL https://doi.org/10.18653/v1/w17-5523
53. D. Hazarika, S. Poria, S. Gorantla, E. Cambria, R. Zimmermann, R. Mihalcea, in *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, ed. by E.M. Bender, L. Derczynski, P. Isabelle (Association for Computational Linguistics, 2018), pp. 1837–1848. URL https://aclanthology.org/C18-1156/
54. D. Ghosh, A.R. Fabbri, S. Muresan, Comput. Linguistics **44**(4) (2018). DOI 10.1162/coli_a_00336. URL https://doi.org/10.1162/coli_a_00336
55. C.V. Hee, E. Lefever, V. Hoste, in *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018*, ed. by M. Apidianaki, S.M. Mohammad, J. May, E. Shutova, S. Bethard, M. Carpuat (Association for Computational Linguistics, 2018), pp. 39–50. DOI 10.18653/v1/s18-1005. URL https://doi.org/10.18653/v1/s18-1005
56. C. Wu, F. Wu, S. Wu, J. Liu, Z. Yuan, Y. Huang, in *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018*, ed. by M. Apidianaki, S.M. Mohammad, J. May, E. Shutova, S. Bethard, M. Carpuat (Association for Computational Linguistics, 2018), pp. 51–56. DOI 10.18653/v1/s18-1006. URL https://doi.org/10.18653/v1/s18-1006
57. J. Patro, S. Bansal, A. Mukherjee, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, ed. by K. Inui, J. Jiang, V. Ng, X. Wan (Association for Computational Linguistics, 2019), pp. 6335–6341. DOI 10.18653/v1/D19-1663. URL https://doi.org/10.18653/v1/D19-1663
58. F.H. Calderon, P.C. Kuo, H. Yen-Hao, Y.S. Chen, in *WISDOM Workshop at KDD* (2019)
59. S. Oprea, W. Magdy, in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, ed. by A. Korhonen, D.R. Traum, L. Màrquez (Association for Computational Linguistics, 2019), pp. 2854–2859. DOI 10.18653/v1/p19-1275. URL https://doi.org/10.18653/v1/p19-1275
60. S.A. Hayati, A. Chaudhary, N. Otani, A.W. Black, in *Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019, Hong Kong, China, November 4, 2019*, ed. by W. Xu, A. Ritter, T. Baldwin, A. Rahimi (Association for Computational Linguistics, 2019), pp. 212–216. DOI 10.18653/v1/D19-5527. URL https://doi.org/10.18653/v1/D19-5527
61. J. Lemmens, B. Burtenshaw, E. Lotfi, I. Markov, W. Daelemans, in *Proceedings of the Second Workshop on Figurative Language Processing, Fig-Lang@ACL 2020, Online, July 9, 2020*, ed. by B.B. Klebanov, E. Shutova, P. Lichtenstein, S. Muresan, C.W. Leong, A. Feldman, D. Ghosh (Association for Computational Linguistics, 2020), pp. 264–269. DOI 10.18653/v1/2020.figlang-1.36. URL https://doi.org/10.18653/v1/2020.figlang-1.36
62. R. Akula, I. Garibay, in *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EACL 2021, Online, April 19, 2021*, ed. by O.D. Clercq, A. Balahur, J. Sedoc, V. Barrière, S. Tafreshi, S. Buechel, V. Hoste (Association for Computational Linguistics, 2021), pp. 34–39. URL https://aclanthology.org/2021.wassa-1.4/
63. S. Oprea, W. Magdy, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 1279–1289
64. R. Akula, I. Garibay, Entropy **23**(4), 394 (2021). DOI 10.3390/e23040394. URL https://doi.org/10.3390/e23040394