Data Science 0 (0) 1
IOS Press

DWAEF: A Deep Weighted Average Ensemble Framework Harnessing Novel Indicators for Sarcasm Detection

Richa Sharma [a,*], Simrat Deol [b], Udit Kaushish [c], Prakher Pandey [d] and Vishal Maurya [e]
[a] *Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, India*
E-mail: richasharma@keshav.du.ac.in; ORCID: https://orcid.org/0000-0002-4472-1681
[b] *Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, India*
E-mail: simrat205711@keshav.du.ac.in
[c] *Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, India*
E-mail: udit205805@keshav.du.ac.in
[d] *Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, India*
E-mail: prakher205723@keshav.du.ac.in
[e] *Department of Computer Science, Keshav Mahavidyalaya, University of Delhi, India*
E-mail: vishal205750@keshav.du.ac.in

Abstract. Sarcasm is a linguistic phenomenon often indicating a disparity between literal and inferred meanings. Due to its complexity, it is typically difficult to discern it within an online text message. Consequently, in recent years sarcasm detection has received considerable attention from both academia and industry. Nevertheless, the majority of current approaches simply model low-level indicators of sarcasm in various machine learning algorithms. This paper aims to present sarcasm in a new light by utilizing novel indicators in a Deep Weighted Average Ensemble-based Framework (DWAEF). The novel indicators pertain to exploiting the presence of simile and metaphor in text and detecting the subtle shift in tone at a sentence's structural level. A Graph Neural Network (GNN) structure is implemented to detect the presence of simile, Bidirectional Encoder Representations from Transformers (BERT) embeddings are exploited to detect metaphorical instances and Fuzzy Logic is employed to account for the shift of tone. To account for the existence of sarcasm, the DWAEF integrates the inputs from the novel indicators. The performance of the framework is evaluated on a self-curated dataset of online text messages. A comparative report between the results acquired using conventional features and those obtained using proposed indicators is provided. The encouraging findings produced after applying DWAEF demonstrate that the proposed method surpasses the outcomes of previous research that made use of primitive features.

Keywords: Sarcasm Detection, Deep Ensemble Learning, Weighted Average Ensemble Model, Graph Neural Networks, BERT, Fuzzy Logic

1. Introduction

Natural languages have evolved gracefully over time all around the globe. Various nuances of a language allow humans to put forth their views on myriad topics with ease and creativity. The use of figurative language by native speakers is one such medium of expressing opinions [1]. Sarcasm, interlaced

*Corresponding author. E-mail: richasharma@keshav.du.ac.in.

with irony and wit, affords both sharpness and subtlety to convey contempt. Automatic detection of sarcasm in the text is one of the critical challenges faced by researchers in the field of sentiment analysis. Sensing the negative connotation in a sentence containing positive words is required to detect sarcasm in an effective manner. Primitive computational models developed for sarcasm detection made use of primitive features such as n-grams, punctuation and intensifiers and exploited machine learning algorithms for classification purposes. To identify sarcasm in text, the research proposes a Deep Weighted Average Ensemble-based Framework (DWAEF). The proposed framework makes use of three indicators to produce competent results. These indications concern utilising the presence of simile and metaphor in text and identifying small shifts in tone between the constituent clauses of a sentence. The framework leverages deep learning components, namely Graph Neural Network (GNN) [2] and Bidirectional Encoder Representations from Transformers (BERT) [3] based embeddings to detect simile and metaphor respectively and Fuzzy Logic [4] to apprehend polarity shifts between the constituent clauses of a sentence. Finally, the outputs of the three components are provided to a Weighted Average Ensemble Model (DWAEF) an ensemble structure comprising Attentive Interpretable Tabular Learning (TabNet), One-Dimensional Convolutional Neural Networks (1-D CNN) and MLP-based learners. The results obtained using the ensemble method are thoroughly compared with results obtained solely using base learners and meta learners classification models. With the accuracy of 92.01% achieved by DWAEF, the proposed ensemble-based approach surpasses the results obtained during earlier studies based on the usage of primitive features only. The main contributions of this study are summarized below:

(1) Leveraging key linguistic features, namely- simile, metaphor and constituent clauses of a sentence for sarcasm detection that, to the best of the authors' knowledge, have not yet been used together for this purpose
(2) Implementing GNN in the framework to detect the presence of simile in a text on the basis of a sentence's dependency tree
(3) Exploiting BERT embeddings to detect the presence of metaphor in a text
(4) Capturing the shift in polarity of a sentence's constituent clauses using fuzzy-logic
(5) Harnessing ensemble structure of various machine learning and deep learning algorithms for facilitating sarcasm classification task.

The rest of the paper is organized as follows. Section 2 discussed the earlier research done in the field of sarcasm detection. Section 3 puts forth the motivation behind the proposed study. Section 4 presents and describes the proposed methodology. Section 5 describes the experiments and gives a detailed analysis of the results obtained. Section 6 concludes the paper.

2. Related Work

Detection of sarcasm is a challenge for humans and for machines, even more so. As a result, it has gained popularity in many NLP applications. An extensive survey of the literature brought to light that researchers have mainly employed the following approaches to detect sarcasm.

- **Machine Learning Methods**: The common form of sarcasm consists of a positive sentiment situation followed by a negative sentiment situation. The study in [5] discussed an algorithm that automatically learns positive and negative sentiment phrases from sarcastic tweets. In [6] authors surveyed several machine learning algorithms to classify the sarcastic tweets and found that a combination of SVM and CNN resulted in higher prediction accuracy. Researchers in [7] applied KNN, RF, SVM, and ME classifiers on the following features- sentiment related, syntactic and semantic, punctuation-related and pattern-related. As per their results, the RF classifier outperformed all applied models with the highest accuracy of 83.1%. In [8], the researchers harvested sarcastic tweets with the help of hashtags such as *#not*, *#sarcasme* and divided them into tweets containing user mentions and tweets that do not. A trained machine learning classifier, Winnow2 [9] was then employed to segregate tweets aimed at specific users from those that were not. In [10], the researchers included extra-linguistic information and employed Binary Logistic Regression with l2 regularization and achieved a gain in accuracy as compared to purely linguistic features in sarcasm detection. In [11], authors used unigrams, bigrams and trigrams to create more general sarcasm indicators which in turn resulted in a 75% precision and 62% recall score for a bootstrapping classifier. Authors of [12] tried identifying sarcastic messages with the help of machine learning algorithms and presented a comparison of the performances of machine learning techniques and human evaluators.

- **Deep Learning and Transformer based Methods**: Researchers in recent studies have employed various neural network techniques such as CNN and LSTM along with different word embeddings viz. Word2Vec, FastText and GloVe on the Reddit Corpus. They accounted for the impact of varying epochs, training size and dropout on the performance [13–15]. In [16], the study implemented BERT, ROBERTa, LSTM, Bi-LSTM and Bi-GRU models for detecting sarcasm in text. They concluded that the transformer-based ensemble performed better than the baseline models scoring 0.43 on F1-score. Authors in [17] used an ensemble of LSTM, GRU and Baseline CNNs to detect sarcasm in online text and concluded using a weighted average ensemble resulted in better results. However, the approach used by the researchers failed to detect sarcastic tweets written in a very polite way. In [18], the researchers used four component methods namely LSTM, CNN-LSTM, SVM and MLP on the Reddit and Twitter datasets resulting in F1-scores of 67% and 73% respectively.
- **Graph Neural Networks based Methods**: Recent studies have made extensive use of word embeddings in deep neural networks for various natural language processing tasks (NLP). However, there is a growing demand for modelling text data as graphs. In comparison to revolutionary neural networks such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), CNN, and BERT, the graphical representation of text allows for more efficient extraction of semantic and structural information. Therefore, numerous researchers have investigated graph-based methods and their application to NLP problems. The first graph attention-based model to identify sarcasm on social media was proposed in [19]. The graph model captured complex relationships between a sarcastic tweet and its conversational context by modelling a user's social and historical context together. Graph Neural Network (GNN), a modern class of networks applied on graph-structured data [20], has found application in the field of sarcasm detection. In [21], a Graph Convolutional Network (GCN) was used to capture global information features in a satirical context and a Bidirectional Long Short-Term Memory (Bi-LSTM) was implemented to capture the sequence features of the comments. The two sets of results were combined and evaluated using a conventional classifier which in turn yielded an accuracy of 73.57%. Later, The authors in [22] proposed an Affective Dependency Graph Convolutional Network framework to detect messages with implied contradictions and incongruity.

Apart from the state-of-the-art technology, many researchers have investigated the use of different features in text data and different methodologies for detecting sarcasm. The article [23] provides a detailed literature survey on sarcasm detection. Additionally, it provided a detailed analysis of the set of features used for sarcasm detection. Subsection 2.1 discusses the types of feature sets used in sarcasm detection and how researchers have employed them in past studies.

2.1. Types of Primitive Feature Sets Used in Sarcasm Detection

Previous works on sarcasm detection made use of low-level features such as n-grams, punctuation, intensifiers and so forth. Some of the primitive features such as punctuation count, count of mixed-case words, count of repeated words and letters, presence of intensifiers and presence of interjections are later used in this research as part of feature set preparation. The primitive features used in sarcasm detection can be broadly classified into:

(1) Lexical Features: This feature set includes text properties such as unigrams, bigrams, n-grams, skip-gram, hashtags, etc. The study in [11] used unigrams, bigrams and trigrams to create more general sarcasm indicators thereby improving the precision and recall of their bootstrapping classifier. Researchers in [10] created binary indicators of lower-cased word unigrams and bigrams along with brown cluster unigrams and bigrams which grouped words used in similar contexts into the same cluster. The authors of [5] extracted every unigram, bigram, and trigram that occurred immediately right after a positive sentiment phrase in a sarcastic tweet.

(2) Pragmatic Features: These are some of the main features used for sarcasm detection in text. They include emoticons, smileys, number of hashtags, replies, and so forth. The study in [7] included the count of positive, negative and sarcastic emoticons. The authors of [24] considered the effect of sentiment contained in hashtags by developing a set of rules around the number of hashtags and their polarity. Researchers in [12] took into account the sentiment of replies to the user.

(3) Hyperbole Features: These features include intensifiers, interjections, quotes, punctuation and so forth. Researchers in [10] created a binary indicator for the presence of 50 intensifiers retrieved from Wikipedia. The authors of the study in [25] opined that writers often use sarcasm-based writing styles to compensate for the lack of visual or verbal cues. The authors in [26] accounted for uppercase and lowercase characters along with the repetition of punctuation marks.

(4) Contextual Features: These features comprise extra components, outside the realm of formal linguistics, used frequently in a sentence, especially in online messages. The researchers in [8] harvested a large number of sarcastic tweets with the help of hashtags such as #not,#sarcasme and divided them into tweets containing user mention and tweets that do not. In [10], the researchers emphasized extra-linguistic information from the context of a tweet in the form of 'Author Features', 'Audience Features', 'Environment Features' and 'Tweet Features' and achieved gains in accuracy compared to purely linguistic features in sarcasm detection.

The previous research and development in the field of sarcasm detection prompted the authors of this paper to take on this problem and address its concerns at a new level. The following section describes the impetus behind the present study.

3. Motivation

This section explains the rationale for delving into the complexities of sarcasm detection using similes, metaphors, and the clausal structure of a sentence. Subsection 3.1 discusses similes in literature and forms the base for the proposed methodology for its computational detection. Subsection 3.2 introduces and explores metaphors in literature thereby building the foundation for its computational detection. Subsection 3.3 deliberates upon a sentence's clausal structure as well as the polarity change from one clause to another. 3.4 lays the motivation for using deep learning methods in an ensemble structure.

Table 1
Examples and constituent components of a simile

Simile	Tenor	Vehicle	Property	Event
Her voice is as smooth as silk.	Her voice	silk	Smoothness	is
A sweet voice carolling like a gold-caged nightingale	sweet voice	gold-caged nightingale	The property here is implicit, left for the reader to infer	carolling
Her grandmother's love story was as old as the hills.	grandmother's love story	hills	old	was
A slow thought that crept like a cold worm through his brain.	slow thought	cold worm	The property here is implicit, left for the reader to infer	crept

3.1. Simile

A simile as mentioned earlier is a figurative device used to draw comparisons between two unlike things. Its presence is always explicitly indicated with the usage of "*like*" or "*as*". A simile consists of the following four key components-*Tenor*, *Vehicle*, *Property*, *Event* and *Comparator* [27]. Table 1 provides examples of similes along with its constituent components. This study proposes the presence of a simile as a potential marker for sarcasm in the text as its presence in a sarcastic remark may accentuate the hidden emotion. For instance, "*Of course they were invited! They are always as welcome as a skunk at a lawn party*" implies that the subject's presence is actually not appreciated. Here the comparison "*as welcome as a skunk at a lawn party*" represents the undesirability and vileness of the subject. Another potential example of a sarcastic remark embedding a simile is "*Asking politicians to give up a source of money is like asking Dracula to forsake blood*" wherein the speaker mocks politicians' flaws by drawing analogies to Dracula. The computational detection of simile relies on the syntactical dependency tree of a sentence, which is described in more detail in section 4.

3.2. Metaphor

A metaphor is a figure of speech that compares two unrelated ideas. At the basic linguistic level, both metaphor and simile involve the juxtaposition of two concepts. However, metaphors lack the usage of "*like*" or "*as*" while drawing the comparison. For example, the two statements, "*Mary is a rock*" and "*Mary is like a rock*" will be inferred by the reader in the same sense about Mary's personality [28]. The only difference between the statements is that the former statement is a metaphor and the latter is a simile. The difference lies in the presence of the comparator "*like*" in one and its absence in the other.

A metaphor also arises when seemingly unrelated properties of one concept are seen in terms of the properties of some other concept. Metaphorical utterances in sarcastic remarks in certain situations are common. For example, "*You are the cream in my coffee*" when used sarcastically implies that the hearer has fallen short of the speaker's affection [29]. Another example of such an utterance is, "*I am not saying that I hate you, what I am saying is that you are literally the Monday of my life.*" wherein the speaker indirectly expresses his hate towards the listener by comparing the latter's presence in the former's life as depressing and unwanted as Monday. Since comparison is drawn between two distinctive entities, computing cosine similarity between the subject and object of comparison forms the bases for its computational detection. This study facilitates the detection of only two types of metaphorical sentences out of the three mentioned by [30]. Table 2 provides a summary of the two types.

Table 2
Types of metaphors addressed in this study and their examples

Metaphor Type	Relationship	Example
Type I	Subject IS A object phrase; (**X** is **Y**)	1. Mary is a rock. 2. He is the sugar in my coffee. 3. That fella is the raspberry seed in my wisdom tooth.
Type II	Verb acting on Noun phrase; (**X** acts on **Y**)	1. My car drinks gasoline. 2. He planted good ideas in their minds. 3. Inflation has eaten up all my savings.

Table 3
Distinctive subtleties between main and subordinate clauses

Sentence	Main Clause	Separator	Sub-ordinate Clause
She had a long career but she is remembered for one early work.	She had a long career	but	-
	She is remembered for one early work		
I first saw her in Paris, where I lived in the early nineties.	I first saw her in Paris	where	(where) I lived in the early nineties
If it looks like rain, a simple shelter can be made out of a plastic sheet.	A simple shelter can be made out of the plastic sheet	if	(if) it looks like rain

3.3. Clauses

Clauses are a group of related words which unlike phrases have a subject and a verb. A clause can be a part of a sentence or be a complete sentence in itself. All sentences have at least one main clause. The main clause is a clause that can stand alone as an independent complete sentence. On the other hand, a subordinate clause is a clause that cannot stand as an independent complete sentence by itself. It is typically introduced with a subordinating conjunction and is dependent on the main clause. Consider the examples taken from an article[1] given in Table 3 elaborating the distinctive subtleties between a main clause and a subordinate clause. Sarcasm, in its prevalent form, exists as the disparity of sentiments. This disparity can further take up two forms [9]:

(1) A shift from positive polarity to negative polarity: In this type, sarcastic sentences contain positive expressions followed by negative expressions. Consider the sarcastic sentence, "*Thank you, officer, now that you have my license I can't drive*" where the main clause "*Thank you officer*" has a positive connotation and the subordinate clause "*now that you have my driving license I can't drive*" has a negative connotation.
(2) A shift from negative polarity to positivity polarity: In this type, sarcastic sentences contain negative expressions followed by positive expressions. For instance, "*I hate my sister because she cooks so well*" wherein the main clause "*I hate my sister*" holds a negative connotation and the subordinate clause "*because she cooks so well*" holds a positive connotation.

To cater to such types of situations, this research proposes to measure the polarity shift from the main clause of a sentence to the subordinate clause of a sentence at various degrees as a potential indicator of

[1]https://www.lexico.com/grammar/clauses

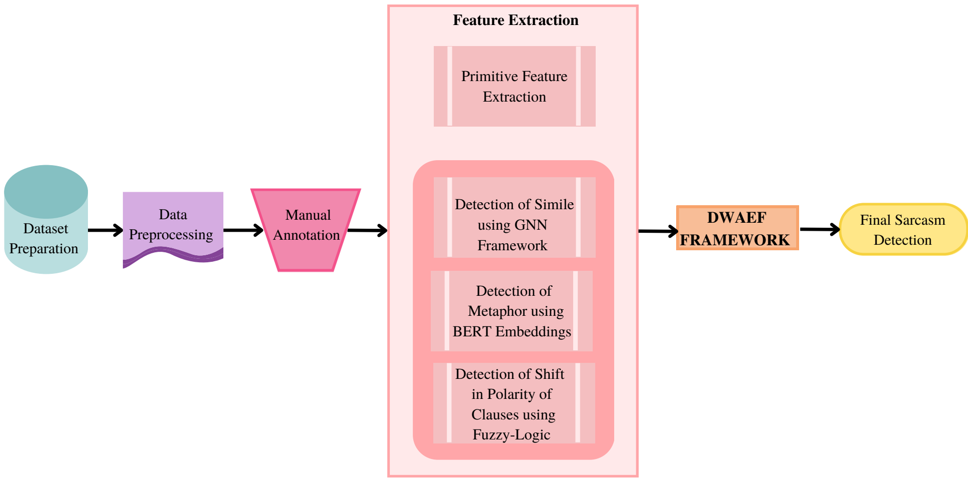


Fig. 1. The Methodology

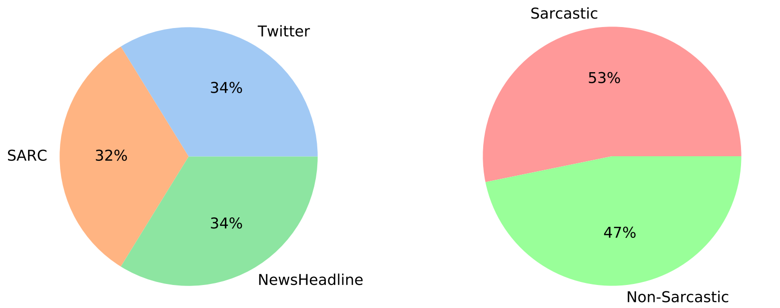
sarcasm. The significance of fuzziness comes into play while dealing with ambiguities surrounding how positive or negative a stand-alone clause can be. Its amalgamation with the computational detection of polarity shift may result in efficient results.

3.4. Motivation behind using a Deep Ensemble Structure

Current cutting-edge research studies utilise geometric deep learning, BERT, and Fuzzy-Logic. This study combines these techniques into a single framework in order to produce competent results. Furthermore, most authors have employed conventional machine learning classifiers for evaluation purposes, whereas ensembles of deep learning algorithms (TabNet, CNN, and MLP) are employed in this research. One of the most significant issues with conventional machine learning techniques is that they frequently fail to capture the underlying characteristics and structure of the data. Consequently, poor performance is observed when these algorithms are applied to datasets that are highly imbalanced, high-dimensional, and noisy. [31]. Therefore, it is essential to construct an efficient model, particularly for complex tasks such as sarcasm detection. Ensemble learning is one of the approaches. Ensemble learning strategies combine multiple machine learning algorithms to produce poor predictive outcomes. These results are then fused together to generate more accurate solutions. Any ensemble framework comprises a collection of base learners and meta learners. Base learners, also known as weak learners, are machine learning classifiers whose predictions are combined with those of other weak learners to compensate for their weaknesses. The meta learner or strong learner is the combined learnt model. The promising results obtained by past researchers with different ensemble structures for sarcasm detection motivated the authors of this work to implement a deep ensemble framework DWAEF. The framework is comprehensively described in the forthcoming section.

4. Methdology

The methodology followed by this research is elaborated in Fig. 1. A detailed description of dataset



(a) composition of the curated dataset

(b) distribution of Sarcastic and non-sarcastic sentences across the dataset

Fig. 2. Description of the curated dataset

preparation is provided in subsection 4.1. Once the data has been collected, it goes through several stages of preprocessing to remove redundancy which is further elaborated in subsection 4.2. The pre-processed data is then annotated by four expert linguists with 100% agreement that the dataset consisted of both sarcastic and non-sarcastic sentences. Following this, the pre-processed and correctly labelled data goes through the feature extraction process wherein along with the indicators proposed by this research, primitive features are also extracted. Followed by feature set preparation, the results are obtained using two ensemble frameworks. Each module portrayed in Fig 1 is explicated in the forthcoming subsections. Subsection 4.3 discusses the primitive feature set preparation. Subsection 4.4 discusses the detection of novel features viz, simile, metaphor and change in the polarity of a sentence's constituent clauses.

4.1. Data Set Preparation

The researchers of this work prepared a dataset of 2891 sentences. Out of these, 1538 were sarcastic and were compiled from various sources- i) 520 sentences were extracted from Twitter with hashtags- #sarcasm, #not, #sarcastic, #irony, #satire; ii) 520 were taken from the NewsHeadline dataset curated by [32]; iii) remaining 498 were taken from the SARC dataset curated by [33]. The 1353 non-sarcastic sentences were compiled from Twitter and the NewsHeadline dataset. Further, four expert linguists independently performed annotation to ensure that 1538 were actually sarcastic and the rest were non-sarcastic. After preprocessing, the dataset was reduced to 2889 sentences. The composition of the dataset and the distribution of sarcastic and non-sarcastic sentences are illustrated in Fig 2

4.2. Data Preprocessing

Since the data on Twitter is full of redundancy due to the rampant usage of slang, hashtags, emoticons, alterations in spelling, loose usage of punctuation, and so forth, the following data pre-processing steps were performed:

(1) Duplicate tweets and re-tweets were also dropped.
(2) Hashtags were completely removed.
(3) Tweets containing URLs were dropped.

(4) Emojis were removed from the text.
(5) The occurrences of the following punctuation marks ['.', '?', '*', '!', ','] were first counted and then the data was freed of irrelevant punctuation marks.

4.3. Primitive Features

The primitive features used by this study include various features explained earlier in section 2. The said feature set consists of punctuation count, count of mixed-case words, count of repeated words and letters, presence of intensifiers and presence of interjections. Each one of the aforementioned features is comprehensively explained below.

(1) Punctuation Count: The punctuation marks are sometimes overdone to indicate sarcasm. For example, to emphasise a point, users use an asterisk ('*'). To represent a pause, an ellipsis ('. . . ') is used and a bunch of exclamatory marks ('!!!') indicate exclamatory utterances [25]. Thus, each of the aforesaid punctuation marks along with some more ['.', '?', '*', '!', ','] were counted as one of the features.
(2) Count of mixed-case words: This feature set includes counting the occurrence of mixed-case words in the text.
(3) Count of repeated words and letters: Users also tend to repeat letters in words to over-emphasize parts of the text. A similar pattern can be observed in the case of words. As a result, the number of repeated letters and repeated words were counted and used as a set of 2 individual features.
(4) Presence of intensifiers: Intensifiers or hyperbolic words are generally adverbs or adjectives which strengthen the evaluative utterance of a sarcastic remark. Consider the utterances were taken from [34], "*'fantastic weather', 'when it rains'*" and "*weather is good when it rains*". Both utterances may literally convey a positive outlook of the speaker. However, sensing the context, the utterance with the word fantastic can easily be identified as sarcastic. For this study, a list of commonly used intensifiers was retrieved from Wikipedia[2] and used to check the presence of intensifiers in the tweets.
(5) Presence of interjections: Interjections are words or phrases primarily used in a sentence to convey emotions. For instance, "*aha*", "*yay*", "*oh*", "*nah*", "*yeah*", "*wow*", and so forth are some of the commonly used interjections. A list of interjections was retrieved from an article[3] and was used to check the presence of interjections in tweets.
(6) Number of times words having opposite polarities come together: This feature captures the contrast between two words having opposite polarities.
(7) Length of the largest sequence of words with polarities unchanged
(8) Count of positive and negative words

4.4. Frameworks for Proposed Features

4.4.1. GNN Framework for Simile Detection

For the purpose of this research, a simile is detected on the basis of its syntactical pattern using GNN Fig. 3 presents dependency trees of two sentences containing similes. The dependency trees were created using Stanford NLP Group's CoreNLP server [35]. A GNN-based text classification model is used to

[2]https://en.wikipedia.org/wiki/Intensifier
[3]https://www.english-grammar-revolution.com/list-of-interjections.html

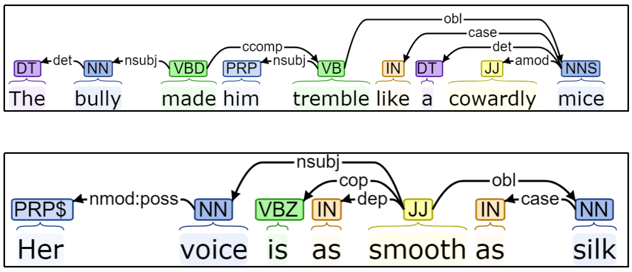


Fig. 3. The dependency trees depicting the syntactic dependency between various components of a simile.

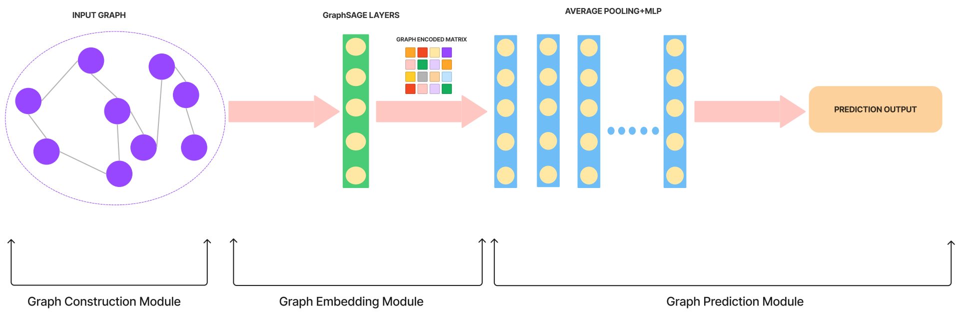


Fig. 4. The GNN framework for simile classification.

learn the dependency structure of similes. The entire set-up for the simile classification model consists of a Graph Construction Module, a Graph Embedding Module and a Prediction Module. Each of the modules is implemented using Graph4NLP library [36]. The modules are elaborated thoroughly below and the entire framework is summarized in Fig. 4.

(1) **The Graph Construction Module**: The graph construction module focuses on building a syntactic dependency tree-based static graph for each of the texts in the dataset. All of the dependency trees are built using Stanford NLP Group's CoreNLP server. Dependency relations from the dependency parsing trees are converted into dependency graphs. 3000 sentences consisting of both similes and non-similes are collected to train the model. Fig. 5 illustrates a series of initialization preprocessing steps that the raw data goes through before being passed to the Graph Embedding Module. The steps are:

- build_topology: This module builds a syntactic dependency text graph for each of the data items in the raw dataset.
- build_vocab: This module is responsible for building vocabulary out of all tokens appearing in the data items.
- vectorization: This module is the lookup step, responsible for converting tokens from ASCII characters to word embeddings

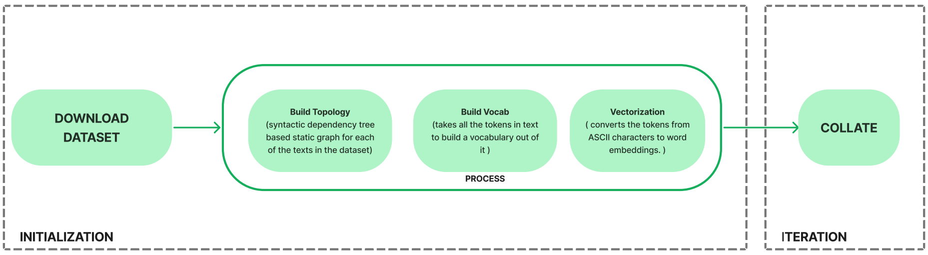


Fig. 5. Dataset preprocessing workflow.

Once the initialization is complete, the data items are collated into the batch data which will be used for runtime iteration over the entire dataset.

(2) **The Graph Embedding Module**: The authors of this research implemented Bi-Fuse GraphSAGE [37], a GNN framework for inductive representation learning of graphs which is used to generate low-dimensional vector representations for nodes. This module implements the message passing and aggregation operations. After the message passing and aggregation of the messages, the embedding of nodes is updated and the final output i.e the encoding matrix of the graph is used as inputs to the prediction module to predict target objects. The mathematical operation of GraphSAGE is given below:

$$h_{N(v)}^{(k+1))} = aggregate(h_v^k, \forall_v \in N(v)) \tag{1}$$

$$h_v^{(k+1)} = \sigma(W^k \cdot concat(h_v^k, h_{N(v)}^{(k+1)} + b)) \tag{2}$$

$$h_v^{(k)} = norm(h_v^k) \tag{3}$$

The embedding generation process takes the entire graph G(V, E) and features for all nodes, $x_i \in V$ as input. In each iteration from k=0 upto k=K where k denotes the current step in the loop and $h_v^{(k)}$ denotes a node's representation at that step, K signifies the number of aggregator functions and W^k denotes set of the weight matrices in each iteration. First, each node $v \in V$ aggregates the representations of the nodes in its immediate neighbourhood, as represented by equation 1, into a single vector $h_{N(i)}^{(l+1)}$. After the aggregation of the neighbouring feature vectors, GraphSAGE concatenates the node's current representation h_v^k, with the aggregated neighbourhood vector $h_{N(v)}^{(k+1)}$, given in equation 2 and this concatenated vector is fed through a fully connected layer with a non-linear activation function represented by σ, following which each current node's representation is normalised as illustrated by equation 3.

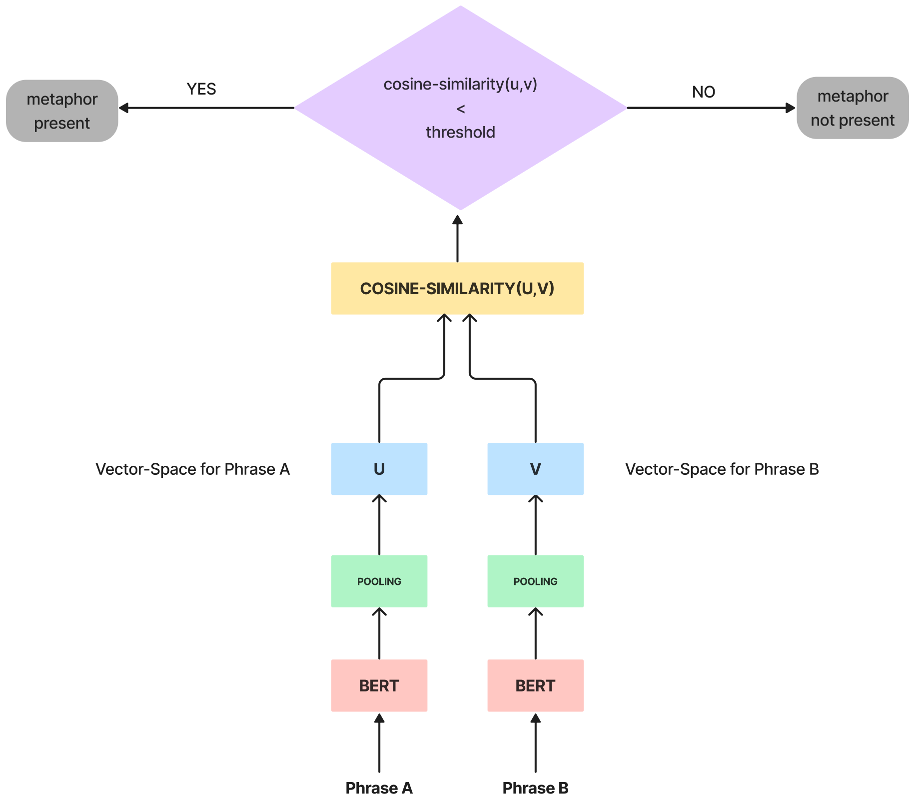


Fig. 6. The framework for metaphor detection.

(3) **The Prediction Module**: The prediction module consists of an average pooling layer with 300 hidden units and an MLP classifier which produces predicted labels. In case of a presence of a simile is detected the framework predicts label ‘1’ and in case of a non-simile, the framework predicts label ‘0’. The said trained framework is saved for predicting the presence of a simile on the sarcasm dataset. The training and validation accuracy and loss curves are discussed in section 5.

4.4.2. Bert-based Structure for Metaphor Detection

The detection of metaphors is achieved by generating BERT embeddings. Fig. 6 illustrates the framework used for detecting the presence of a metaphor and Fig. 7 illustrates the BERT-based network used for generating the embeddings with the hidden layer representations in red. For the BERT base, each encoder layer outputs a set of dense vectors.

Each vector contains 768 values each of which is nothing but contextual word embeddings. Initially, each sentence is split into two halves. For sentences of type I, phrase A consists of the subject and phrase B consists of the object. On the other hand, for sentences of type II, phrase A consists of a verb which acts on a noun phrase represented by phrase B. BERT-based embeddings are generated for both phrases A and B and cosine similarity is calculated. The entire process can be summarized as follows:

Detecting Type-1 Metaphors:

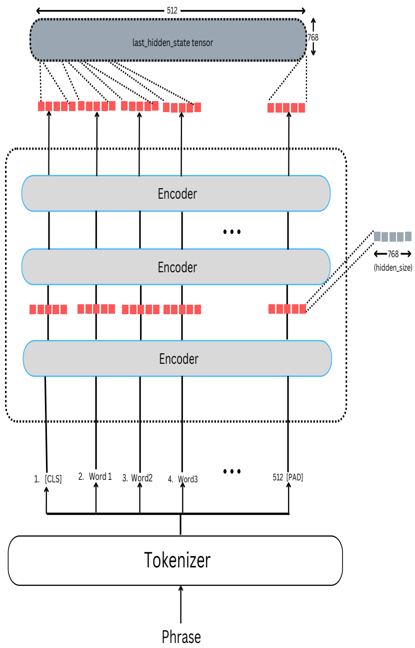


Fig. 7. BERT-based network used for generating embeddings.

(1) All the sentences in the dataset are first tokenized using spaCy.
(2) The sentences are then split into two phases: one containing the subject of comparison and the other containing the object of comparison.
(3) Dense contextual embeddings are constructed for each of the phrases. The last_hidden_state tensor from the BERT model is extracted to quantify textual similarity. This vector is then moulded by a pooling operation that takes the mean of all token embeddings and compresses them into a single vector space representing a single phrase. Furthermore, the cosine similarity between the two vector spaces of the phrases is calculated. Following multiple trials, a threshold value of 0.7 was chosen to determine the presence or absence of a metaphorical instance in a sentence. A cosine similarity larger than 0.7 accurately indicated the lack of a metaphorical statement, whereas one less than 0.7 suggested its presence.

Detecting Type-2 Metaphors:

(1) All the sentences in the dataset are first tokenized using spaCy.
(2) All the sentences are split into two phrases: one containing the personified verb and the other containing the object of comparison.
(3) For each of the phrases, dense contextual embeddings are generated and textual similarity is measured in terms of cosine similarity.

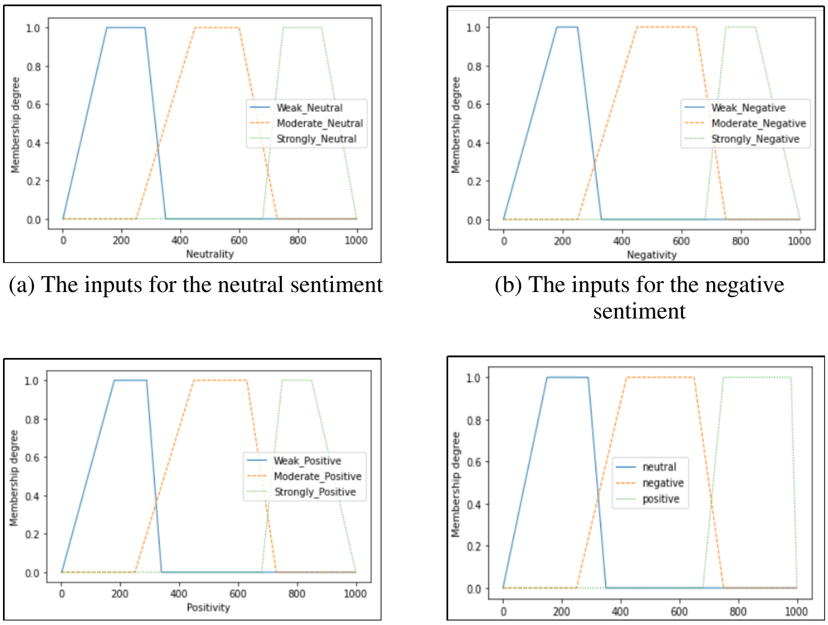
4.4.3. Fuzzy Logic-based Approach for Capturing Polarity Change in Clauses

The following steps were performed to detect sarcasm in the form of disparity of sentiments in clauses:

Table 4
The degree of each sentiment along with the corresponding polarity percentage associated with it

Sentiment	Degree	Range
Positive	Weakly Positive	0-35 %
	Moderately Positive	25-75 %
	Strongly Positive	68-100 %
Negative	Weakly Negative	2-34%
	Moderately Negative	25-73 %
	Strongly Negative	68-100 %
Neutral	Weakly Neutral	1-35%
	Moderately Neutral	25-73 %
	Strongly Neutral	68-100 %

(1) Tokenization: All the pre-processed textual data is first tokenised using spaCy's NLP object.
(2) Separation of clauses: To find the polarity of a sentence's constituent clauses, a sentence is first separated into clauses. This is done in two ways. All the sentences are checked for the presence of separators from the following list ['*after*', '*before*', '*as soon as*', '*while*', '*when*', '*as*', '*because*', '*since*', '*if*', '*provided that*', '*as long as*', '*unless*', '*although*', '*though*', '*even though*', '*then*', '*which*', '*who*', '*that*', '*whose*', '*and*', '*but*', '*&*']. If a sentence does not contain any of the separators mentioned then splitting of the sentence is done on the basis of two markers. The first marker is the subject of the sentence with syntactic dependency "nominal subject (nsubj)". The second marker is the last occurrence of any preposition in a sentence. It is marked with syntactic dependency "preposition (prep)". These markers divide the sentence into three parts, the first part spans from the beginning of the sentence to the first marker ("nsubj'), the second part spans from the first marker("nsubj") to the second marker("prep") and the third part spans from the second marker("prep") to the end of the sentence. For example, the sentence, "*You're everything I want in someone, I don't want anymore.*" splits into "*You're everything I want*" and "*in someone I don't want anymore*". Another example would be, "*Right before I die I am going to swallow a bag of popcorn kernels to make the cremation a bit more interesting.*" splits into "*Right before I die*", "*I am going to swallow a bag of popcorn kernels*" and "*to make the cremation a bit more interesting*".
(3) Computing the polarity of clauses: For finding the polarity of a sentence's constituent clauses, pysentimiento [38] is used. pysentimiento is a python toolkit for sentiment analysis and text classification. It is a transformer-based open-source library. It uses BERTweet [39] as a base model in English. The list of constituent clauses is taken for each sentence and the polarity score corresponding to each clause in the form of positive, negative and neutral proportions is obtained.
(4) Applying Fuzzy Logic to eliminate overlapping sentiment classes: In each sentence, every clause has three sentiment proportions i.e., positive, negative and neutral. To determine the overall polarity of a clause Fuzzy-Logic has been implemented using Simpful [40]. Although pysentimiento provides sentiment proportions for the positivity, negativity and neutrality of a clause, it does not provide any valuable information about the degree (weak, moderate, strong) of each sentiment. This study uses fuzzy logic to determine the overall polarity of the clauses with the help of a set of rules based on the projected degree of each sentiment. The degree of each sentiment viz. Positive, negative and neutral have been devised using the assumed ranges given in Table 4. The trapezoidal membership function is used to define a non-polygonal fuzzy set for each sentiment viz. positive, negative and neutral. Fig. 8 (a,b,c) illustrates various inputs for each sentiment.



(a) The inputs for the neutral sentiment

(b) The inputs for the negative sentiment

(c) The inputs for the positive sentiment

(d) The fuzzified output

Fig. 8. Fuzzy inputs for neutral, negative and positive sentiments

Following is the set of fuzzy rules:

(a) IF (Neutral IS Weak_Neutral) AND (Negative IS Weak_Negative) AND (Positive IS Moderate_Positive) THEN (Output IS positive)
(b) IF (Neutral IS Weak_Neutral) AND (Positive IS Weak_Positive) AND (Negative IS Moderate_Negative) THEN (Output IS negative)
(c) IF (Positive IS Weak_Positive) AND (Negative IS Weak_Negative) AND (Neutral IS Moderate_Neutral) THEN (Output IS neutral)
(d) IF (Positive IS Weak_Positive) AND (Negative IS Moderate_Negative) AND (Neutral IS Moderate_Neutral) THEN (Output IS negative)
(e) IF (Negative IS Weak_Negative) AND (Positive IS Moderate_Positive) AND (Neutral IS Moderate_Neutral) THEN (Output IS positive)
(f) IF (Neutral IS Weak_Neutral) AND (Negative IS Moderate_Negative) AND (Positive IS Moderate_Positive) THEN (Output IS positive)
(g) IF (Neutral IS Strongly_Neutral) AND (Negative IS Weak_Negative) AND (Positive IS Weak_Positive) THEN (Output IS neutral)
(h) IF (Positive IS Strongly_Positive) AND (Negative IS Weak_Negative) AND (Neutral IS Weak_Neutral) THEN (Output IS positive)
(i) IF (Negative IS Strongly_Negative) AND (Neutral IS Weak_Neutral) AND (Positive IS Weak_Positive) THEN (Output IS negative)

The fuzzified output is presented in (d) part of Fig. 8. Defuzzification is then applied to get the final polarity output.

(5) Checking for the disparity of sentiments: The total number of clauses with positive, neutral, or negative sentiment labels are counted and utilised to account for polarity shifts from negative to positive or positive to negative. If such a shift occurs, the function returns '1'; otherwise, it returns '0'.

4.5. The Deep Weighted Average Ensemble Framework

DWAEF, the proposed Deep Weighted Average Ensemble-based Framework, is a three-tiered structure. It is comprised of three base learners: a TabNet [41], a 1-D CNN and a Multi Layer Perceptron. The curated dataset is used to pre-train the three models. During training, each of the base learners receives the outputs of the GNN-based simile detection framework, the BERT-based metaphor detection framework, and the Fuzzy-based polarity shift detection framework. Next, the predictions produced by each module of the ensemble are weighed. Based on the Dirichlet distribution, a weight optimisation search is carried out along with a randomised search on the dataset. The previously trained models, a TabNet [41], a 1-D CNN and an MLP, are added to the Dirichlet Ensemble Object. After the model is fitted using the Dirichlet Markov Ensemble Method, its resulting accuracy is acquired. No meta-learner is used in this ensemble method.

The findings that were achieved through the utilisation of the suggested methodology are reported in section 5.

5. Evaluation, Results and Analysis

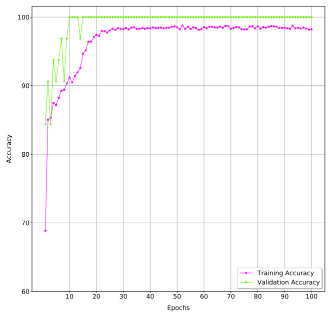
This section examines the results acquired by employing different techniques on the dataset. The purpose of the proposed methodology is to efficiently detect sarcasm in online text using the presence of figurative comparisons, i.e., similes and metaphors and shifts in polarity of the text's constituent clauses. This segment is organised as follows: Subsection 5.1 discusses the accuracy vs epochs and loss vs epoch curves for the GNN framework. The accuracy score obtained during experimentation with different threshold values by the BERT-based Metaphor Detection Framework is discussed in subsection 5.2. The confusion metrics for the fuzzy-based approach are presented in subsection 5.3. Subsection 5.4 discusses the results obtained using DWAEF, the deep weighted average ensemble model.

5.1. Results obtained by the GNN-based Simile Detection Framework

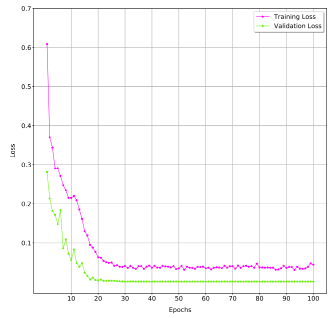
The GNN framework described in section 4.4.1 was pre-trained on a dataset of 3000 sentences, out of which roughly 50% were similes, and the rest 50% were non-similes. This pre-trained framework was then tested on the main collated sarcasm dataset to extract the presence of a simile as one of the features. With a batch size of 32, the model was executed for 100 epochs. The rest of the hyperparameters are given in Table 5. The training and validation curves of the proposed framework are illustrated in Fig. 9. It is evident from both curves that the framework is free from both overfitting and underfitting. The testing accuracy obtained using the proposed GNN framework for simile detection was 99.22% using GloVe word embeddings. The state-of-the-art results ensured accurate detection of the simile for the main sarcasm dataset.

5.2. Results obtained by the BERT-based Metaphor Detection Framework

Before settling on the best threshold value to assess the existence or absence of a metaphorical instance in a sentence, several values were tested. The various values tested and the accompanying accuracy values are listed in Table 6. It is clear that at a threshold value of 0.7, the most accurate predictions were achieved for both type 1 and type 2 metaphors. Thus, a cosine similarity of more than 0.7 accurately indicates the absence of a metaphorical remark, whereas a cosine similarity of less than 0.7 indicates its presence.



(a) Training accuracy (in pink) and validation accuracy (in green) vs number of epochs



(b) Training loss(in pink) and validation loss (in green) vs number of epochs

Fig. 9. Training and validation curves of the GNN Framework for simile detection

Table 5
Hyperparameter settings for the GNN framework

Parameter	Value
Seed	1234
Batch size	32
Epochs	100
Learning rate	0.01
Learning rate patience	2
Learning rate reduce factor	0.5
Hidden layers	300
Drop out	0.3
Graph pooling	avg_pooling
Optimizer	Adam
Loss	Cross Entropy
Activation	ReLU

Table 6
Various Threshold Values and the Corresponding Accuracy Values

Threshold value	Accuracy	
	Type1	Type2
0.3	0.72	0.60
0.4	0.62	0.60
0.5	0.70	0.68
0.6	0.78	0.66
0.7	**0.82**	**0.78**

5.3. Results obtained by the Fuzzy-based Polarity Shift Detection Framework

The confusion matrix shown in Fig. 10 depicts the performance of the Fuzzy-based Framework. In the matrix, there are four distinct combinations of expected and actual values. It is evident from the matrix that the framework properly identified the variations in polarity that actually indicated sarcasm at the clausal structural level of 1309 sentences. Additionally, 125 incorrect sentences were identified

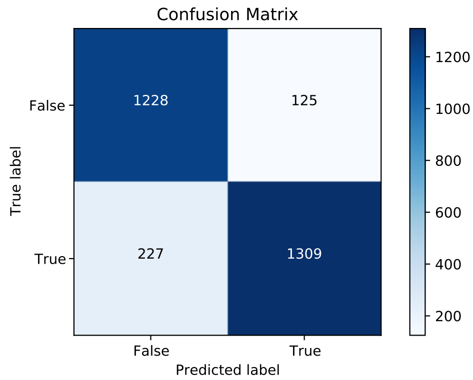


Fig. 10. Confusion matrix for the Fuzzy-based Polarity Shift Detection Framework

as true. On the other hand, it also correctly identified the lack of a tone change in 1228 sentences but failed to recognise a polarity shift in 127 sentences. It is obvious from the matrix that the framework made accurate predictions for 87.81% of the whole dataset. While 91.28% of all predicted true classes were predicted actually true, 85.22% of all real true classes were predicted true by the framework. One of the reasons for the state-of-the-art outcome is the usage of fuzzy rules to cope with uncertainties over whether a solitary sentence is positive or negative.

5.4. *Results obtained by DWAEF*

DWAEF's performance is assessed in two stages. In stage 1, the results are obtained by the DWAEF base learners (TabNet, 1-D CNN and MLP) individually vs the DWAEF, using only the primitive features. In stage 2, the results are obtained by the same models using a combination of both primitive features and proposed features. Table 7 gives the corresponding hyperparameter settings for each of the models used in DWAEF and the results are displayed in Table 8. In each case, combining primitive features with the proposed features yielded superior outcomes. A more detailed comparison report is summarised in Table 9, where the researchers compared the accuracies of the three most powerful and widely used traditional machine learning classifiers, Random Forest (RF), Support Vector Machine (SVM), and AdaBoost (AB), using a combination of the proposed features and primitive features, with the accuracies of the DWAEF base learners and the overall accuracy of DWAEF.

Table 9 summarizes accuracy scores obtained by all models using proposed features in combination with primitive features. It can be inferred that the DWAEF outperforms all the models used in this study in terms of accuracy. Also, the proposed features of this research, viz, presence of figurative comparisons, i.e., simile and metaphor and shift in polarity of a sentence's constituent clauses, successfully aid in better detection of sarcasm in online text messages. The detection of sarcasm has also been boosted by switching to a deep weighted average ensemble framework because the framework assigns each base member's share of the prediction weight based on how well it performed individually during training. Moreover, Researchers in [17] fell short of detecting sarcasm in sentences written in a formal and polite tone. However, including the proposed novel indicators successfully detected sarcasm in such sentences. Table 10 presents some of them.

Table 7

Hyperparameter settings for TabNet, 1-D CNN and MLP used in DWAEF

Model	Hyperparameter Settings
TabNet	Optimizer: Adam Learning Rate: 0.001 step_size:10 Gamma:1.4 Mask_type: entmax
1-D CNN	Seed:1234 Learning rate: 0.0025 Dropout rate: 0.8 Loss: sparse categorical cross entropy Optimizer: SGD
MLP	Activation: ReLU Alpha: 0.00025 hidden_layer_sizes: (200,150,100,50,25) learning_rate: adaptive Solver: Adam max_iter: 200 random_state: 25

Table 8

Accuracy scores for TabNet, 1-D CNN, MLP and DWAEF

Stage	TabNet	1-D CNN	MLP	DWAEF
Stage 1: Primitive features only	76.80	75.03	64.09	78.00
Stage 2: Primitive features + proposed features	**89.80**	**87.07**	**85.21**	**92.01***

Table 9

Summary of accuracy scores of all classifiers

Model	Accuracy Score(%)
RF	81.37
SVM	78.58
AB	81.13
MLP	85.21
TabNet	89.80
1-D CNN	87.07
DWAEF	**92.01**

Table 10

Examples of Correctly Classified Sentences Written in a Polite/Formal Tone

Sentences	Novel Indicator Present	Classification Result(%)
Everyone has a photographic memory, some just don't have film.	Metaphor	1
When it comes to finding a good place to eat you can't doubt her choice, she's a connoisseur of food no wonder why she eats like a pig.	Simile, Clauses	1
No, you're right, we should just put the mentally ill down like dogs if they do something inappropriate.	Simile	1

6. Conclusion and Future Work

Detection of sarcasm poses one of the leading challenges in sentiment analysis, as a single sarcastic remark can influence sentiment analyzers to produce undesirable results. Primitive techniques used in sarcasm detection used low-level features and traditional machine learning algorithms.

The study looked into sarcasm detection with a new perspective. It proposed DWAEF, a deep-weighted ensemble-based framework for sarcasm detection. The framework utilized figurative speech components mainly, the presence of simile, the presence of metaphor and the change in the polarity of a sentence's constituent clauses using deep learning techniques. The predictions done by the above modules were then fed into DWAEF, which comprised a 1-D CNN, a TabNet and an MLP as its base learners.

Based on the results, it can be concluded that combining the proposed indicators with the primitive features achieved better results across all classifiers. It was seen that the proposed ensemble framework performed better as compared to traditional machine learning classifiers. The proposed technique achieved the highest accuracy of 92.01% when proposed indicators were combined with primitive features and evaluated using a weighted average ensemble of deep learning algorithms.

The study employed various state-of-the-art tools and techniques; still, the proposed framework may be made to improve the model's characteristics and efficiency. In future, the authors plan to incorporate more advanced tools in the framework and equip it to perform cross-lingual and multimodal predictions.

References

[1] E.W. Pamungkas, V. Basile and V. Patti, A joint learning approach with knowledge injection for zero-shot cross-lingual hate speech detection, *Inf. Process. Manag.* **58**(4) (2021), 102544. doi:10.1016/j.ipm.2021.102544.

[2] M. Gori, G. Monfardini and F. Scarselli, A new model for learning in graph domains, in: *Proceedings. 2005 IEEE international joint conference on neural networks*, Vol. 2, 2005, pp. 729–734.

[3] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).

[4] L.A. Zadeh, Fuzzy logic, *Computer* **21**(4) (1988), 83–93.

[5] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert and R. Huang, Sarcasm as contrast between a positive sentiment and negative situation, in: *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 704–714.

[6] S.M. Sarsam, H. Al-Samarraie, A.I. Alzahrani and B. Wright, Sarcasm detection using machine learning algorithms in Twitter: A systematic review, *International Journal of Market Research* **62**(5) (2020), 578–598.

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

[8] K. Hallmann, F. Kunneman, C. Liebrecht, A. van den Bosch and M. van Mulken, Sarcastic Soulmates: Intimacy and irony markers in social media messaging, in: *Linguistic Issues in Language Technology, Volume 14, 2016-Modality: Logic, Semantics, Annotation, and Machine Learning*, 2016.

[9] N. Littlestone, Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Machine learning* **2**(4) (1988), 285–318.

[10] D. Bamman and N. Smith, Contextualized sarcasm detection on twitter, in: *proceedings of the international AAAI conference on web and social media*, Vol. 9, 2015, pp. 574–577.

[11] S. Lukin and M. Walker, Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue, *arXiv preprint arXiv:1708.08572* (2017).

[12] S. Muresan, R. Gonzalez-Ibanez, D. Ghosh and N. Wacholder, Identification of nonliteral language in social media: A case study on sarcasm, *Journal of the Association for Information Science and Technology* **67**(11) (2016), 2725–2737.

[13] P. Mehndiratta and D. Soni, Identification of sarcasm using word embeddings and hyperparameters tuning, *Journal of Discrete Mathematical Sciences and Cryptography* **22**(4) (2019), 465–489.

[14] M.S. Razali, A.A. Halin, L. Ye, S. Doraisamy and N.M. Norowi, Sarcasm detection using deep learning with contextual features, *IEEE Access* **9** (2021), 68609–68618.

[15] A. Baruah, K. Das, F. Barbhuiya and K. Dey, Context-aware sarcasm detection using bert, in: *Proceedings of the Second Workshop on Figurative Language Processing*, 2020, pp. 83–87.

[16] M. Abdullah, J. Khrais and S. Swedat, Transformer-Based Deep Learning for Sarcasm Detection with Imbalanced Dataset: Resampling Techniques with Downsampling and Augmentation, in: *2022 13th International Conference on Information and Communication Systems (ICICS)*, IEEE, 2022, pp. 294–300.
[17] P. Goel, R. Jain, A. Nayyar, S. Singhal and M. Srivastava, Sarcasm detection using deep learning and ensemble learning, *Multimedia Tools and Applications* (2022), 1–24.
[18] J. Lemmens, B. Burtenshaw, E. Lotfi, I. Markov and W. Daelemans, Sarcasm detection using an ensemble approach, in: *proceedings of the second workshop on figurative language processing*, 2020, pp. 264–269.
[19] J. Plepi and L. Flek, Perceived and intended sarcasm detection with graph attention networks, *arXiv preprint arXiv:2110.04001* (2021).
[20] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei and B. Long, Graph neural networks for natural language processing: A survey, *arXiv preprint arXiv:2106.06090* (2021).
[21] S. He, F. Guo and S. Qin, Sarcasm Detection Using Graph Convolutional Networks with Bidirectional LSTM, in: *Proceedings of the 2020 3rd International Conference on Big Data Technologies*, 2020, pp. 97–101.
[22] C. Lou, B. Liang, L. Gui, Y. He, Y. Dang and R. Xu, Affective dependency graph for sarcasm detection, in: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1844–1849.
[23] P. Chaudhari and C. Chandankhede, Literature survey of sarcasm detection, in: *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, IEEE, 2017, pp. 2041–2046.
[24] D.G. Maynard and M.A. Greenwood, Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis, in: *Lrec 2014 proceedings*, ELRA, 2014.
[25] A. Rajadesingan, R. Zafarani and H. Liu, Sarcasm detection on twitter: A behavioral modeling approach, in: *Proceedings of the eighth ACM international conference on web search and data mining*, 2015, pp. 97–106.
[26] F. Barbieri, F. Ronzano and H. Saggion, UPF-taln: SemEval 2015 tasks 10 and 11. Sentiment analysis of literal and figurative language in Twitter, in: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015, pp. 704–708.
[27] V. Niculae and C. Danescu-Niculescu-Mizil, Brighter than gold: Figurative language in user generated comparisons, in: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 2008–2018.
[28] J. O'Donoghue, Is a metaphor (like) a simile? Differences in meaning, effect and processing, *UCL Working Papers in Linguistics* **21** (2009), 125–149.
[29] M. Popa-Wyatt, Ironic metaphor: a case for Metaphor's Contribution to Truth-conditions, in: *E. Walaszewska, M Kisielewska-Krysiuk & A. Piskorska (ed.) In the Mind and Across Minds: A Relevance-theoretic Perspective on Communication and Translation*, 2010.
[30] S. Krishnakumaran and X. Zhu, Hunting elusive metaphors using lexical resources., in: *Proceedings of the Workshop on Computational approaches to Figurative Language*, 2007, pp. 13–20.
[31] X. Dong, Z. Yu, W. Cao, Y. Shi and Q. Ma, A survey on ensemble learning, *Frontiers of Computer Science* **14**(2) (2020), 241–258.
[32] R. Misra and P. Arora, Sarcasm detection using hybrid neural network, *arXiv preprint arXiv:1908.07414* (2019).
[33] M. Khodak, N. Saunshi and K. Vodrahalli, A large self-annotated corpus for sarcasm, *arXiv preprint arXiv:1704.05579* (2017).
[34] C. Liebrecht, F. Kunneman and A. van Den Bosch, The perfect solution for detecting sarcasm in tweets# not (2013).
[35] C.D. Manning, M. Surdeanu, J. Bauer, J.R. Finkel, S. Bethard and D. McClosky, The Stanford CoreNLP natural language processing toolkit, in: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.
[36] L. Wu, Y. Chen, H. Ji and B. Liu, Deep learning on graphs for natural language processing, in: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2651–2653.
[37] W. Hamilton, Z. Ying and J. Leskovec, Inductive representation learning on large graphs, *Advances in neural information processing systems* **30** (2017).
[38] J.M. Pérez, J.C. Giudici and F. Luque, pysentimiento: A Python Toolkit for Sentiment Analysis and SocialNLP tasks (2021).
[39] D.Q. Nguyen, T. Vu and A.T. Nguyen, BERTweet: A pre-trained language model for English Tweets, *arXiv preprint arXiv:2005.10200* (2020).
[40] S. Spolaor, C. Fuchs, P. Cazzaniga, U. Kaymak, D. Besozzi and M.S. Nobile, Simpful: a user-friendly Python library for fuzzy logic, *International Journal of Computational Intelligence Systems* **13**(1) (2020), 1687–1698.
[41] S.Ö. Arik and T. Pfister, Tabnet: Attentive interpretable tabular learning, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, 2021, pp. 6679–6687.